

HOME COMPUTERTM magazine

FOCUSING EXCLUSIVELY ON ● APPLE ● COMMODORE ● IBM ● TEXAS INSTRUMENTS

Vol. 5 No. 4

\$3.50 in USA
\$4.50 in Canada

Time Management

Computer-Assisted Efficiency Comes Home

Featuring A Ready-To-Use Time Manager for:

- Creating Efficient Schedules
- Customizing Calendars
- Maintaining A Telephone Database
- Printing A Personal Appointment Book

Special Apple, Commodore, IBM, & TI Software:

- Honing Trigonometry Skills

um Mine
mulation
odroid Action
Adventuring

—Speeding Up
A BASIC Program

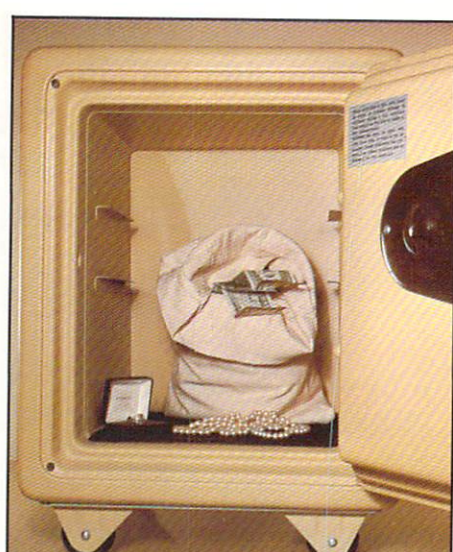
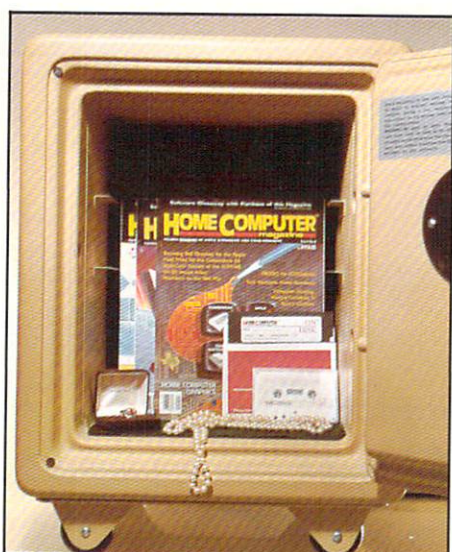
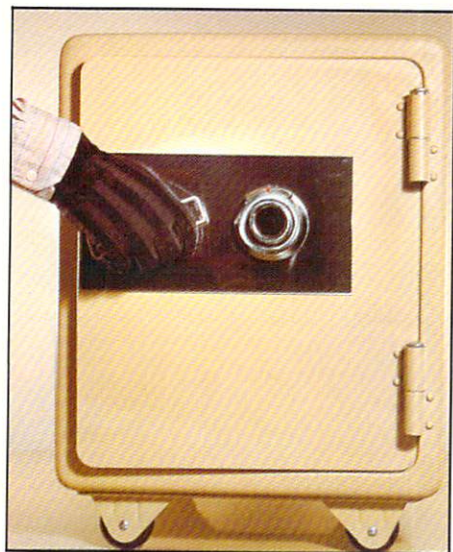
—Reviews Galore:

- ★ Desk Tools & Communication Sidekicks
- ★ Loading & Display Utilities
- ★ Automotive & Adventure Simulations
- ★ Icon Magic & Memory Expansion

Contains
37
Type-In
Programs!



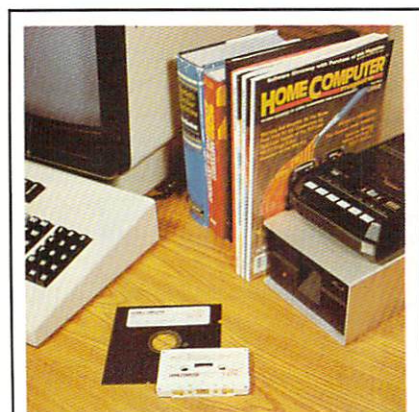
TANDY 1000 USERS:
RUN our PC/PCjr Programs.
See Page 130



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.

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



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



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

VERSIONS SUPPORTED:

Machine

APPLE II Family (A)
COMMODORE 64 (C)
IBM PC/PCjr (I)
TI-99/4A (T)

Media

ON DISK™
ON DISK™/ON TAPE™
ON DISK™
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: Please note that some HCM programs may not RUN on your machines, because of differences in hardware and/or BASIC interpreters.



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



Issue 4.4:

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



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 enhancement 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

This space reserved for Issue 5.4

This space reserved for Issue 5.5

HCM BACK ISSUES

FOR NEW READERS



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 4 Hot Home Brands

We are 4 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 3 of the 4 systems you may not even have. You'll learn more about your Apple, 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 (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 and Mexico add \$11 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 and Mexico. 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.

Each separate contribution to this June 1985 issue and the issue as a collective work is Copyright © 1985 by Emerald Valley Publishing Co. All rights reserved. Copying done for other than personal or internal reference use without the permission of Emerald Valley Publishing Co. is prohibited. Requests for special permission or bulk orders should be addressed to the publisher.

Limited License for use of programs in Home Computer Magazine. Emerald Valley Publishing Co. (EVP) is the owner of all rights to the computer programs and software published in this magazine. To allow for use of the software by the purchaser of the magazine, EVP grants to such purchaser only, the limited license to enter these programs into the purchaser's computer, and to place such programs on a diskette or cassette for the purchaser's personal use.

Any other use, distribution, sale, or copying of these computer programs without the written consent of EVP is expressly prohibited and in violation of this limited license and the copyright laws.

Home Computer Magazine, HCM, and Home Computer Digest are trademarks of Emerald Valley Publishing Co.

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, G.R. Michaels,
Steven P. Nelson, Patricia Swift,
Randy Thompson

User Group Editor Judy Campbell
Assistant Editor Dana M. Campbell

Program Translators
Stephen A. Cordon, Robert Paschelke,
Nancy Vendelin

Asst. to the Publisher Rhea J. Grundy
Production Manager Norman Winney, Jr.
Creative Director Gel-Lei Gom

Photography
Nelson Stevens, K.D. Wainsworth
Production Assistant Rachel Knight
Customer Service Tel. (503) 341-1029
Dealer Sales & Distribution Tel. (503) 341-1036
Main Switchboard Tel. (503) 485-8796



Outside HCM

Once again, that amazing machine called the computer becomes many machines in one. Need an alarm clock that tells you why it's gone off? Need a calendar that highlights special days and appointments? How about a schedule that prints out in many forms, including a neat little appointment book? It might take a lot of time to gather all those things together—if you couldn't use a home computer. Your computer's adaptability is one reason why we at **Home Computer Magazine** keep coming up with so many good ways to use it.

Inside HCM

Time... is there ever enough? In this day and age, time may seem as precious as gold—which is another way of saying "time is money." One thing is sure: time is *relative* to the situation at hand. A bored young child on a long summer's day may think time will never pass. But to adults immersed in their workaday world, time is all too short—a day's activity just doesn't seem to fit into one earth revolution.

One solution is to manage our time before it manages us. But what is the best tool for taming time? Clipboard and stopwatch? Napkin notes and wake-up calls? Such a helter-skelter approach can create a very tedious mess. But computers—those masters of *tedium*—may also be the masters of time.

Inside this issue, we provide a program that can manage your time simply by dividing it into a workable schedule. Take your appointments for a day, week, or month and run them through *Run-Day-View*. Then call your schedule up on screen, or print out a pocket-sized appointment book, a weekly calendar, even a handy phone list. Sound simple? It is.

Always *in date*, and at the core of every computer, is the time-saving art of mathematics. Trigonometry, in particular, conserves both time and effort by using the triangle as an indirect measuring tool—letting us find, for example, the height of a mountain without going to the mountain. So, take the time to learn some *Trig-Trix*, a program (not a breakfast cereal) that complements last issue's *Geometrix* program as a practical exercise in another fundamental area of math.

Instant feedback from a program like *Trig-Trix* is a time-honored educational

technique. But mimicry—or nowadays, computer *simulation*—is another excellent teacher. This issue's *Mine Over Matter* program simulates a huge uranium mining operation (the first step in a fuel cycle leading to the nuclear power plant in last issue's *Over-Reaction*).

Digging into a less serious vein, *Archeodroid* plays out the future excavation of Planet Earth. And to complete our software digs, we continue to provide unique machine-specific applications in each of our regular columns. Dive in as *Apple Seedlings* bakes a pie chart; *Commodore Hornblower* sifts through SID's filters; *IBMpressions* looks in at windowing; *Razzle Dazzle* records sound-on-sound with the 99/4A; and the new *MAC-ROs* column draws pixel-by-pixel on the Macintosh.

There's a time for our software, and there's a time for our user-friendly reviews. This issue, we dig for the truth about a variety of products, all vying for your hard-won dollars. From a learn-it-yourself computer model of an *Injured Engine*, to build-it-yourself programs like *Adventure Master* and *Adventure Construction Set*; from the PC and PCjr's famous *Sidekick*, to Apple's friend *Jane*; and from a new expansion system for the PCjr, to a bevy of utilities for the C-64 and TI-99/4A—we look for the best, and help you avoid the worst.

To make your computer time even more valuable, we offer a wealth of regular features and tutorials, both old and new. Programmer's can trim runtime with Part 2 of *Speeding Up a BASIC Program*. And the final installment of *Build a LOGO Adventure* will add hours of, well... *adventure*.

Is there ever enough time? Only time itself will tell. But, at **Home Computer Magazine**, our time is yours...

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

HCM

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

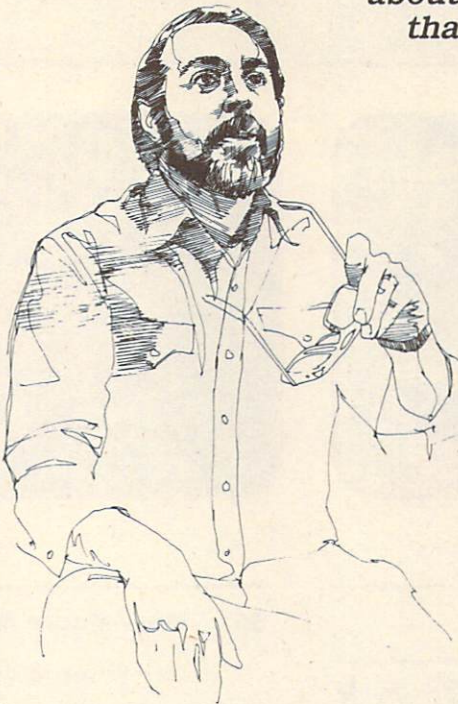
A few months back, we began an ambitious project to fine-tune this magazine—tailoring it more precisely to the specific needs of our changing readership. When I asked for your written input, many of you responded magnificently. We received suggestions from virtually all 50 states, all provinces of Canada, and quite a few foreign countries. The age of respondents ranged from 10 to 85. An enlightening experience, indeed...

So, what is the result of this project? Well, as they say—"the proof of the pudding is in the eatin'." Examine this issue and you'll notice the new *Programmer's Window* pages, plus the *edge-tab markings* and the new *Listings Contents* page—making it easier to find the 3 separate-but-related program sections (i.e., Software Instructions, Programmer's Windows, and Program Listings).

Look for the *HCM Glossary Terms* line at the end of articles; it flags important words, and allows you easy "look-up" in our specially prepared *HCM Glossary* in the rear of the issue. As for our product reviews, many of them will now carry a *Counterpoint* box for added balance. And finally, the column *Algorithm-A-Tricks* brings a new level of understanding to readers who want to know the "tricks of the trade."

Each issue in the next several months will introduce more enhancements, suggested by your ideas. One forthcoming feature is particularly exciting: *Problems in Productivity* will center on real-world problems and tasks to which we'll apply our own *HCM* productivity programs—software such as *Snap-Calc*, *The Organizer*, and *It Figures!* So if you don't have these back issues (see back-issue pages inside the front cover), now's the time to get them along with their corresponding *ON DISK* or *ON TAPE* media.

I want to personally thank each one of you who has taken the time to submit your "On Screen Feedback." The feedback that *HCM* is still receiving is so valuable that it would be unfair to select a winner of the free trip to Eugene, Oregon at this time. There have been, however, many of you who intended to write but "just never got around to it..." So, please continue sending your one-page letters with suggestions and constructive criticism to: On Screen Feedback, Home Computer Magazine, P.O. Box 70288, Eugene, OR 97401. (For guidelines, see this column in Vol. 5, No. 2.) And just so you have no excuses this time, I've included an essential memory aid, shown here:



As I write these words, my thoughts wander to last week's Consumer Electronics Show in Chicago. As expected, Big Blue and Big Red didn't exhibit at the show. The Commodore floor area featured its 128K machine, originally unveiled in January, while the new Atari Corporation again demonstrated its XE and ST series. Many software publishers were conspicuously absent. The mood at the computer portion of the show was rather quiet and subdued—mirroring the present plateau-state of the industry.

"From media pundits we hear about an outbreak of 'closeting' that is running rampant..."

Although the show reflected a natural period of maturation and consolidation within an industry—any industry—it's unfortunate that this greatly shrunken exhibition has already started to fuel more articles in the popular press about the "death" of the home computer market. From media pundits continually looking for that "big story," we hear about an outbreak of "closeting" that is running rampant, decimating the ranks of home computer users everywhere. We're told of millions of home machines that are now "old technology" and therefore "obsolete."

What we *don't* hear—because it doesn't make a good "story"—is that most of these machines have *not* as yet been pushed to their limits. It's just *not* fashionable anymore to report on the millions of smart consumers who continue to benefit from being "early innovators." And with a new generation of software and peripherals just now starting to appear, even greater utility, learning, and enjoyment is "in the cards" for present computer users. Furthermore, as the "newer technology" machines (such as the Macintosh, Atari ST series, and Commodore Amiga) become available and more affordable, we can expect to see more of the new features—such as, "iconoclastic" mouse environments, CD ROM mass-storage devices, and inexpensive hard-disk drives—filter down to our older machines.

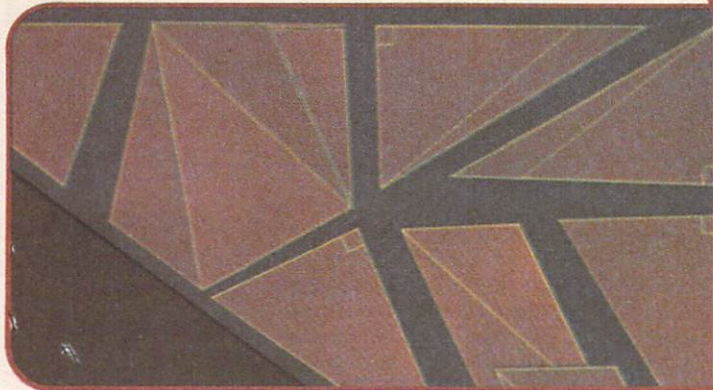
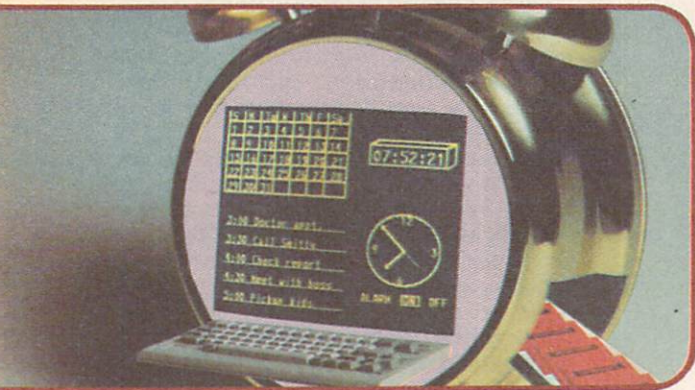
Although time is on our side, we cannot afford to just sit and wait for the industry to eventually recover its past vitality. It's up to us—active computer users—to keep as many machines out of America's closets as possible. For if people "give up" on computers, it hurts all of us who remain; industry growth and innovation slows as a result.

But it's no good just harping on the problem; we need a well-defined plan of attack if we're going to pull this industry out of its present doldrums. We here at *HCM* can offer you the ammunition: *our magazine*, the best "closet fighter" we know. *HCM* gives home-computer owners a reason to be active users; it provides inexpensive software, activities, and a learning environment that keeps those wonderful machines out of the dreaded closets.

So, if you believe in getting as much as you can from this important industry, we ask you now to become a *Home Computer Evangelist*. Go forth and show your favorite magazine at user groups, offices, friend's homes, scout troop meetings, PTA get-togethers, etc. Let 'em know that home computing with *HCM* is a worthwhile and enjoyable pursuit. And above all, stay with us—there are definitely interesting times ahead...

HOME COMPUTER™

magazine



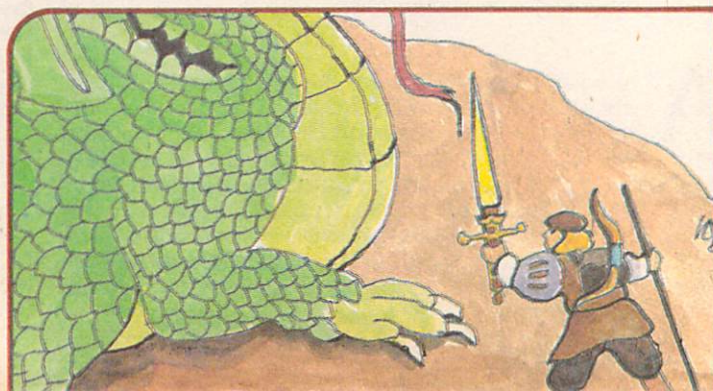
FEATURES

- 16 Run-Day-View™**
 Make your schedule, then view or print.
 by Randy Thompson
 HCM Staff
- 19 Trig-Trix™**
 Using the triangle for indirect measurement.
 by Roger Wood
 HCM Staff
- 22 Archeodroid™**
 From the future, dig the past.
 by B.J. Bruns
 and the HCM Staff
- 24 Mine Over Matter™**
 Sell that uranium—but remember to replant!
 by William K. Balthrop
 HCM Staff
- 28 MAC-ROS™**
 Mac-drawing, one pixel at a time.
 by William K. Balthrop
 HCM Staff
- 30 IBMpressions™**
 Looking through computer windows.
 by Scott Williams
- 32 Razzle Dazzle™**
 Record in three-part harmony.
 by William K. Balthrop
 HCM Staff
- 34 Apple Seedlings™**
 How about an Apple pie . . . chart?
 by Roger Wood
 HCM Staff

- 36 Commodore Hornblower™**
 Get in and change SID's filters.
 by Randy Thompson
 HCM Staff
- 58 Speeding Up A BASIC Program**
 Part 2 puts the pedal to the metal.
 by John P. Russo
 and the HCM Staff
- 62 Algorithm-A-Tricks™**
 Spotlighting this issue's best software procedure.
 by the HCM Staff
- 63 Build A LOGO Adventure**
 Part 4 provides the entire kit.
 by Andrew Kelth
 and the HCM Staff

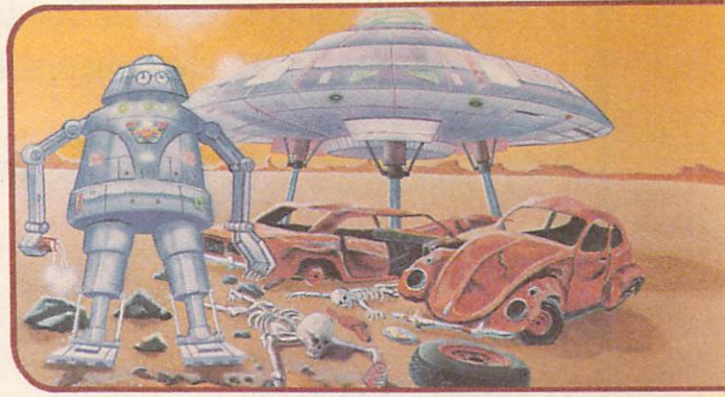
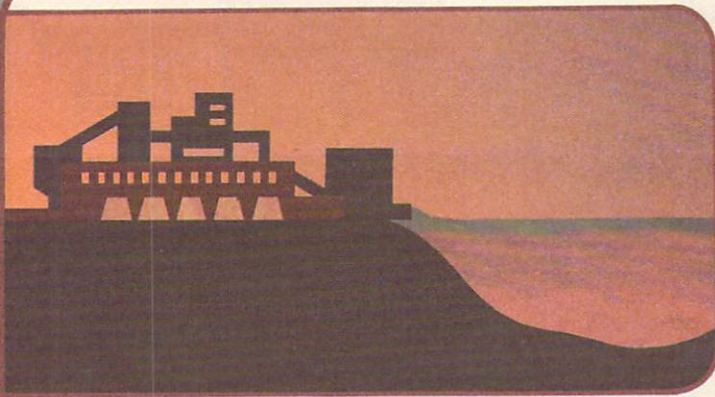
PRODUCT REVIEWS

- 39 Injured Engine**
 Be a greaseless grease-monkey.
 A Review



CONTENTS

VOLUME 5 NUMBER 4



40 Sidekick

On the side, software with a real kick.



A Review

42 Jane

An Iconoclastic view of an Icon-driven program.



A Review

44 Worlds In Creation: Adventure Construction Set & Adventure Master

Of the two, which can create a better world?



A Review

46 Romancing the PCjr: The Quadjr. Expansion Chassis

Expansive, but tricky.



A Review

49 The Display Enhancement Package

Make that 28-column screen bigger than Texas.



A Review

50 SkiWriter II

Word-processing over the wire.



A Review

51 Alien Addition

An edu-game shows its age.



A Review

52 A Day At The Races: Kwik-Load! vs. Mach 5

Which utility breaks the tape?



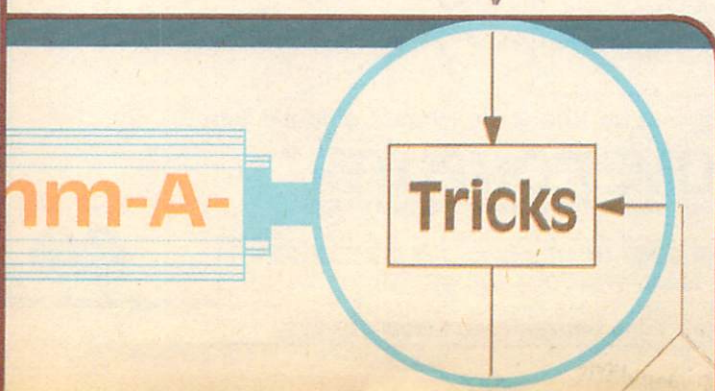
A Review

DEPARTMENTS

- 5 Welcome to HCM
- 6 Inside/Outside HCM
- 7 On Screen
- 11 Letters to the Editor
- 27 HCM One Liners
- 38 HCM Review Criteria
- 60 Home Computer Industry Journal™
- 65 HCM Product News
- 69 HCM Glossary
- 70 Program Typing Guide
- 71 Program Listing Contents
- 71 Programmer's Window Contents
- 130 DeBugs on Display
- Home Computer Tech Notes:
 - 54 Apple
 - 55 Commodore
 - 56 IBM
 - 57 TI

— ATTENTION TANDY 1000 OWNERS —

See the special instructions on page 130 that allow you to RUN the program listings of this issue.



HISTORICAL NOTE
99'er Magazine (founded in December, 1980) was
the forerunner of Home Computer Magazine.

THE BEST

**SUPER
CLOSE-OUT
SPECIAL
for
TI-99/4A
USERS**

A Giant Home Computer Compendium™
for the Texas Instruments 99/4A

OF **99'er**™

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.
- A selection of sensational game software featuring full-color graphics, animation, and sound effects.
- 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.
- 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 \$19.95 + \$3.00
(Book alone) shipping

Best Of 99'er \$35.00 + \$2.50
(Tape Set alone) shipping

USE BIND-IN
CARD AT
CENTER OF
MAGAZINE

**SPECIAL
OFFER**

Buy the Tape Set for ONLY \$35.

And Get the Book **FREE** + FREE SHIPPING

**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!



**FREE
BONUS
WHILE
SUPPLIES
LAST**

PLUS

2
Programming Techniques and Languages



4
LOGO



**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.

TO ORDER—USE BIND-IN CARD
AT CENTER OF MAGAZINE

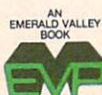
**THE BEST
OF
99'er**



People Who Know the Home Computer Best™

HOME COMPUTER™
magazine

IF CENTER BIND-IN ORDER CARD IS MISSING—
MAIL WRITTEN ORDER TO:
P.O. Box 70288 Eugene, OR 97401
FOR CREDIT CARD ORDERS CALL 1-(800)-828-2212



Letters

to the Editor

Tandy is Dandy

Dear Sir:

I thought that I should write and let you know that the program listings in HCM for the IBM PC and IBM PCjr will run on the Tandy 1000 home computer. I have been buying your magazine for 5 months now and have copied several of the programs listed with hardly any problems at all.

The MICROSOFT GWBASIC used by Tandy is virtually identical to IBM's BASICA. The only difference that I have been able to find is that you must put a space before and after the equals sign. Also, until Tandy releases the updated version of GWBASIC, you cannot use the tiling option in the PAINT statement.

I have enjoyed HCM very much and look forward to each issue. I hope that more Tandy 1000 owners discover the treasure of quality program listings in your magazine.

Robert Eastman
Chandler, AZ 85224

Well, Robert, as you can see from the cover, great minds think alike. We also discovered that the Tandy 1000 will run most of the PC/PCjr software that we publish. And, indeed, we've noticed the problem that you mentioned about the tiling option in the PAINT statement. In fact, we have made a change to one of the programs this month to accommodate this, so that it will run without any problems. Tandy 1000 owners will also notice that there is a section of the DeBugs page that talks about the differences between the listings as printed for IBM, and the changes necessary to make them function with the Tandy 1000.

Music to Our Ears

Dear Sir:

I am looking for music software. I own a Commodore 64 computer with a Datasette, which I run through an ordinary television. I do not own a disk drive or printer and do not know when I can afford one. All music software that I've seen requires one or both.

I've managed to program a crude synthesizer and music editor. What I'd like to be able to program is a synthesizer that will allow me to play something and print the notes to the screen (my synthesizer program does this crudely) and allow me to save it, play it back, edit it, and do it with all three voices at once.

My crude knowledge of programming allows me to adapt programs, somewhat. What I'd like to know is which of your back issues contained music programs, as I may want to order those back issues or tapes.

I appreciate any help you can offer.

Marino Mangini
Naples, NY 14512

Marino, your timing is a little bit off, but close! Starting with Home Computer Magazine, Vol. 5, No. 2, there was a two-part article titled "The Music of Sound" which talked about software that will run on your C-64—strictly musical, of course. For some HCM software, check out our "Commodore Hornblower" column which, starting in Vol. 5, No. 2, began the complete description of a full-up synthesizer making use of all of the capabilities in the SID chip. Once this software

is completed in a future issue, it will be offered on Commodore cassette tape and will allow you to store and save your music via your Datasette recorder. By the way, we have had such a favorable response to our music-related articles that the publisher of Home Computer Magazine (Emerald Valley Publishing Co.) has announced a new magazine devoted to this area. The magazine, called Music & Electronics, will promote "creative discovery through sound technology," and is targeted to readers of all ages and backgrounds—with or without home computers—interested in learning more about musical composition, performance and education through the magic of electronics.

Figuring on the IIc

Dear Sir:

Well, you finally did it. After four straight monthly newsstand purchases with two back issues ordered, I subscribed. Your number-crunching program, "It Figures!" (Vol. 5, No. 2), is what really did it. If someone is interested in physics and mathematics, how could they resist? Already I am planning on adding separate data bases for the hundreds of physical variables and equations.

By coincidence, this same issue evaluated a Microsoft version of CP/M. The only fault of my Apple IIc is that it does not run CP/M. Now a new product promises to turn the IIc into a CP/M machine plus add more RAM memory. Applied Engineering of Carrollton, Texas has a new Z-RAM for the IIc (yes, the IIc). Up to 512K plus CP/M is available for a machine which was not manufactured as "expandable." Could you evaluate this system?

Of the many interesting CP/M programs, "MU-Math" by Microsoft is the one I want most. The program solves equations the same way an algebra student does. It works on symbols, not just numbers, to factor, differentiate, integrate, etc. The final answer does not have to be a number. The program runs on IBM and, until recently, on CP/M. I said "until recently" because Microsoft has withdrawn the CP/M version.

Harold A. Lamkin
Mt. Clemens, MI 48043

We're glad you appreciated our efforts in "It Figures!," Harold. The CP/M add-on product for the IIc that you mention sounds interesting and we'll try to schedule it for review in a future issue. Just a note to all readers: it's a great help when you do let us know what products you would like to see reviewed in Home Computer Magazine. Due to our finite editorial space and the fact that we do reviews in great depth, it's important for us to select products that our readers are interested in seeing reviewed—not just ones which receive a lot of hype.

Using RAM Disk on the IBM PC

Dear Sir:

I have accidentally discovered a method to greatly increase the execution speed of many (protected) disk-based programs. One drawback is that not all protected programs will work this way. As an example, I use the "IBM Writing Assistant" for long reports, and the spell-check speed is greatly increased, as is the access time of switching modes.

My PC has 640K in this example:

1. On boot-up, I auto-configure a 360K RAM disk called drive C. The reason I use 360K, leaving 278K free, is that I can always copy a full disk, but this large RAM disk is not always needed.

2. Insert your working program disk in drive A. In this case it would be "IBM Writing Assistant." (Be sure to remove the write-protect tab.)

3. Enter

`COPY CON:G.BAT (PRESS ENTER)`

`COPYA:*.* C: (PRESS ENTER)`

`C: (PRESS ENTER)`

`WRITE (PRESS ENTER)`

Press F6. This will copy a file called G.BAT on your work disk.

*IMPORTANT - DO NOT TRY TO USE DISKCOPY. For some unknown reason, it does not work on some protected disks.

4. From now on, all you must do is put your program disk in drive A and press G.

At this time, the programs will automatically copy one by one onto your RAM disk. Then it will load and run the program. You may remove your work disk if you wish, as it will not be used. Be sure to place a formatted disk in either drive to save any file you may create.

The speed and time saved is just amazing, and it's faster than using a hard disk.

Robert T. Collina
Lake Hopotcong, NJ 07849

Yes, indeed, Robert, that is a great way to run the "IBM Writing Assistant" with the spelling checker. In fact, if you look at the Quadri Expansion System review in this issue, you will find that the same technique works with an expanded PCjr (with only some problems).

Play TI Again, Sam

Dear Sir:

I have a TI-99/4A, and an Apple IIe, and a Korg-Ply-800 synthesizer. I have seen MIDI interfaces in other magazines for the Apple, but I was wondering if they had any for my TI or if any people who have TI's know how to make them. If not a MIDI interface, how about some software or plug-in keyboard that will turn my TI into a synthesizer. It would also be nice if there was a TI Hornblower.

George Tsihlas
Norristown, PA 19403

The Roland Company makes a device known as the MPU401 which contains MIDI interfaces, allowing up to four devices—such as keyboard synthesizers—to be connected to a computer. The interface to the MPU401 is through an 8-bit parallel port. This device looks like a good interface for the TI-99/4A equipped with a parallel port, although at this time we are uncertain as to whether a special cable adapter will be required between the two units, or whether software exists that will run on the TI machine to support the MPU401. By the way, take a look at this month's "Razzle Dazzle" column for a great TI sound utility.

Likes TI Tech Notes

Dear Sir:

I was wondering if the Apple I is compatible with the Apple programs in your issues? I really enjoy William K. Balthrop's Tech Notes on

Letters

to the Editor CONTINUED

"Doing Without Extended BASIC."

One more thing. I received an ON TAPE issue (Vol. 4, No. 4) with "Stadium Jumping" on it. What is the game based on and how do you play?

Jeff Noble
Evanston, IL 60202

Jeff, in order to run Apple programs that are supplied to you ON DISK from HCM, you need to have at least 64K of memory and have Applesoft BASIC resident in Read Only Memory—precluding the somewhat-antique Apple I. A note to readers buying ON TAPE and ON DISK products: Make sure that you also have (or order) the corresponding volume and issue number of Home Computer Magazine. The magazine contains all the information about every program contained on your magnetic media. It's impractical for us—a waste of both computer memory and magazine paper—to imbed all the program instructions into the listings. All of the information you seek for "Stadium Jumping" is contained in the Vol. 4, No. 4 issue of HCM.

No-Advertising Policy Hailed!

Dear Editor:

Being involved in marketing of computer software programs for 3 years, I have become disgusted with the procedures of many computer magazines.

I have found that many magazines will not review a particular computer software program unless the company who published it is actively advertising in the magazine. The reason they resist reviewing programs from nonadvertisers is because of the pressure they get from companies who do advertise. I can empathize with those companies—but, as an end-user, I want to know which programs are the best, and what their outstanding features are. I'm not interested in a manufacturers' advertising program—I just want the best program available!

It is refreshing to find that Home Computer Magazine is interested in reviewing computer software programs based on the program's merits. The result is that readers of Home Computer Magazine benefit in the fact that reviews are not biased.

Wade E. Gefre
Marketing Manager
Solidus International Corp.
Bellingham, WA 98225

Thank you, Wade, for your kind comments. The response to our no-advertising policy has been very positive. We're grateful that a good number of hardware and software producers like yourself (Sysres for the C-64 reviewed in Vol. 5, No. 3) also see the wisdom of this policy.

Switch Printer

Dear Sir:

I have an Apple IIc computer and a TI-99/4A system with a TI impact printer (Epson MX-80) with serial and parallel ports. How can I use the TI printer with my Apple IIc? What modifications must be made to the printer? Also, can I run the TI printer with the Apple IIc and the TI-99/4A with an external switch?

Alan Gellerstein
Spring Valley, NY 10977

Basically, the answer to your question, Alan, is yes. There are boxes that will allow serial devices to be switched, but you may not want to spend the money required for such a box (they are fairly expensive). You can manually plug and unplug the two cables at the back of your common printer. The big problem involves finding out which internal printer DIP switch settings will match both the IIc and the TI. For instance, if you have a setup of 9600 baud, 8 bits with no parity, and an automatic line feed, will this produce the same results from either machine? In addition, you may have to spend a little time with the "RS232C" port statement used with your TI machine, making sure that it produces the same results as the PR# statement in your Applesoft program, but certainly it can be done. Good luck and have fun.

Accessing MouseText

Dear Sir:

I am 14 years old and own an Apple IIc (as well as a TI-99/4A and a VIC-20), but that's not why I'm writing. I was working on a bar graph program over the past week. I got the program to make some nice graphs (3-D and all that), but I wanted to make a nice title page with orderly menus. Let me get off the track a second. A while back I was turning off the 80-column card and I accidentally hit [ESC] [CONTROL] 4 (I found out later that you don't have to hit control and that [CONTROL] 8 turns off the 80-column card). This neat little solid-white cursor appeared. The menu options were printed in 40-column MouseText.

After awhile I found that it could only be turned on (in 40 columns) from outside a BASIC program (disappointment and anger—does this have too many brackets or are those braces?). At any rate, I found a way to turn on MouseText and I wondered if you could help me do it from inside a program.

Congratulations! Your magazine is the best.

Don Scott
Buffalo, NY 14207

MouseText is really fairly easy, Don. Essentially, all you have to do is first turn on 80-column mode with a PR#3 command, do an INVERSE command, and then print the ESCape character, CHR\$(27) in your program. Now the regular capital letters will all be MouseText instead. To get back to regular text, just print a CHR\$(24) and do a NORMAL command. Here's a 4-line program that PRINTs the whole set:

```
100 PRINT CHR$(4); "PR#3"
110 INVERSE
120 PRINT CHR$(27)
130 FOR CH=64 TO 95:PRINT
CHR$(CH); " ";:NEXT
140 PRINT CHR$(24);:NORMAL
```

Of course this program only works on an Apple IIc—but it should provide you with the means needed to put MouseText into your program. For a more complete discussion of the subject, see Home Computer Magazine Vol. 4, No. 5, page 63.

Responds to HCM Review

Dear Sir:

Legacy Technologies was aware of your product review in issue 5.2 of HCM.

Our general reaction was favorable towards

the review. We felt that Mr. Brader did in fact use the product, and give accurate information regarding its operation and design.

There is a lot of power packed into the Legacy product and we apologize for not allowing better clearance on the memory cards, although to date no one has caused any damage to any L-Bus card. The sound he refers to signifies to our staff—a well-built U.S. product.

In fact, the review could have been improved if we had provided another memory card and our CPS card, and I apologize for that.

Since the announcement from IBM regarding the PCjr, we have witnessed a change in dealer support for the PCjr: hence, Legacy now will offer our product directly to the consumer at a discount.

Legacy will also be offering a 256K RAM memory card that should be available this summer. The card will be a 256/512K RAM memory card, allowing the Legacy owner to expand to a system total of 640K all on one memory card (presently you must purchase 2 EXP 256 cards).

There is one simple rule of thumb for utilizing software on the Legacy/PCjr—Is the program written through DOS and BIOS constraints? Legacy changes neither of these, so a PC program written within these will run on an expanded PCjr with a Legacy.

Thank you for considering the Legacy II for the PCjr as a product for your magazine's review.

Greg Brehm
Vice President of Marketing
Legacy Technologies
Lincoln, NE 68504

Thank you, Greg, for your update on the Legacy product line.

FORTH On the TI Computer

Dear Sir:

I am writing this in reference to the continuing saga of the "TI Forth" vs. "Wycove Forth" controversy.

I have been an owner of the Wycove Forth system for just about a year now and I am thoroughly satisfied with its operation. I have found it to be a much easier version to program in and much more versatile in application.

As you know, the TI version of Forth was released in an "unfinished" state due to the untimely pullout of Texas Instruments from the home computer market. Several of the screens have errors in them as they were published, and the system does not utilize the capabilities of the 99/4A as the system was initially developed. In addition, the TI system is a "hybrid" Forth in that for most (in fact, almost all) of the options to be loaded, the assembly-language screens must be loaded first. The result is a trade-off between speed (our benchmarks gave the TI version a slight edge in speed) and ease of programming.

I have found the Wycove version to be a much easier system to develop and program, and it is much more of a true Forth language. I very much prefer the ease of programming to the very slight loss of speed.

The Wycove system contains full speech and sound capabilities as well as a "nifty" little clock screen. The standard features delivered with the

Wycove system make the TI version pale by comparison.

Incidentally, Wycove makes Forth available to cassette-based TI-99/4A systems and it runs on Extended BASIC, Mini-Memory, or Editor/Assembler. In my view, it is one of the best software packages offered for the TI-99/4A.

Robert Carmany
Greensboro, NC 27407

Your comments about TI Forth and its shortcomings, Robert, lead to a question for our readers: Since Texas Instruments put TI Forth in the public domain, has any individual or group made significant improvements to the language? Has anyone solved the problems that Robert addresses? And does anyone have a dynamite Forth application to share in our pages?

More Jr Expansion

Dear Sir:

I have just finished reading your article in Vol. 5, No. 2 on the Legacy II expansion unit for the PCjr and I am delighted that I chose the Quadram Expansion Chassis instead. The Quadram Expansion Chassis was a breeze to install, even for an electronic idiot like me and it looks a whole lot better sitting on my junior than the pictures of the Legacy II in your article. And, with the Quadram, I not only got the second disk drive, but also an internal battery clock and a printer port for a graphics printer.

I have seen the Quadram Expansion Chassis advertised for \$540, and the additional memory board with 128K for \$215. Additional memory can be added in 64K increments at about \$25 per chip. Clearly, more expansion for the money!

I get a lot of computer publications, but you're the best! Keep up the excellent work. How about an article on the Quadram Expansion Chassis in a future issue?

Royce R. Logan
North East, MD 21901

We were certainly happy to see that you are satisfied with your Quadram Expansion Chassis, the Quadjr, and yes—we thought we should review this device. As you can see, our review appears in this issue.

Commodore Plus 4 with ON DISK

Dear Sir:

I have recently "upgraded" from a TI-99/4A to a Commodore Plus 4. My question is: will ON DISK programs for the Commodore 64 run on the Plus 4? If so, I plan to order several of the back issues.

I am having trouble typing in some of the programs and making them work properly. I got "Loan Calc" to work okay, but thus far I have been unsuccessful with "Quiz-Make." Of course, this could be a bug of my own making.

I am a long-time subscriber and I have never failed to make a program work on the TI (including "Quiz-Make"). If there are slight differences in the Commodore 64 and the Plus 4, how about publishing the differences for us nonprogrammers?

SAVE FOR REFERENCE

Franklin Owners Rejoice!

Dear Sir:

First, good news for owners of Apple clones like Robert Hose (whose letter appeared in Vol. 5, No. 3 of HCM) and me. I give the credit for finding this patch, which lets clones run ProDOS, to someone known only as "The Shadow" (draw your own conclusions).

The problem is that the ProDOS loader looks for specific values in ROM which are different on the Franklin. If none are found, the computer freezes up. The following steps will make your ProDOS disk bootable:

1. Boot ProDOS. When the system hangs up, press RESET.
2. Type 2647:EA EA(ProDOS 1.0) or 265B:EA EA (all other versions).
3. Type 2000G (no space between the last zero and the G).

Congratulations! But the change is not permanent; to make it permanent, you must modify the ProDOS file (Hope you haven't erased that disk yet, Mr. Hose!). The following BASIC program will do just that.

```
10 ONERR GOTO
20 DS = CHR$(4):REM
30 INPUT "INSERT DISK TO BE
PATCHED AND PRESS RETURN":RS
40 PRINT DS:
"BLOADPRODOS.TSYS,AS$2000"
50 POKE 9819,234:POKE 9820,234
60 PRINT DS:"BSAVE
PRODOS.TSYS,AS$2000":END
70 ER = PEEK(222):IF ER < > 6 AND
ER < > 8 THEN PRINT "ERROR":
ER:END
80 IF ER = 8 THEN PRINT
"I/O ERROR":END
90 PRINT CHR$(7):"PRODOS FILE
ISN'T ON THIS DISK!": GOTO 30
For ProDOS 1.0 change line 50 to:
50 POKE 9799,234:POKE 9800,234
Keep up the good work!
```

Steve Sobol
Beachwood, OH 44122

Steve, that is indeed great news to potential ON DISK subscribers who are Franklin owners. Thanks for the information.

I enjoy your magazine immensely and look forward to each issue.

William Moon
Booneville, AR 72927

No, William, in most cases our ON DISK programs designed for the Commodore 64 will not operate without alteration on the Commodore Plus 4. Because "Loan Calc" is a text-only program, without sound or graphics, it is a notable exception. Many hardware and software differences exist between the two machines. Commodore 64 BASIC programs rely heavily upon POKES for simple functions, such as changing screen color, where the Plus 4 has many built-in BASIC commands to handle the same functions. Thus, a careful study of the 2 BASICs is required before translations could be accomplished.

Writing Assistant Does Long Letters

Dear Sir:

I am a brand-new subscriber to your magazine who checked out almost every computer magazine I could find before subscribing to Home Computer. It is undoubtedly the best!

I am writing in regard to a letter from James McCloskey on page 9 of Vol. 5, No. 1.

I believe you have neglected to tell him the easiest way to write long documents using the "IBM Writing Assistant" (which I am using at this time). The best way to write these long documents is to write them in pieces of about four pages, and then use the "JOIN" command, which is described in the "Writing Assistant" manual, to join them together for printing. This has solved the problem for me and for my boys who must write term papers and other long documents for school. They have written 20-page long documents using this method.

I use an IBM PCjr, and now that IBM has seen fit to discontinue manufacturing it, I

hope that Home Computer Magazine will not abandon it too.

Stuart A. Sylvester
Wantagh, NY 11793

Thanks for the tip, Stuart, on the JOIN command in "IBM Writing Assistant." This certainly will allow you to write longer documents on your printer, but it still does not provide the complete flexibility one would hope for in being able to move text around within a large document. On your concern, don't worry—HCM will keep supporting PCjr. It is a fine machine with many capabilities. And now that its price has been officially cut on remaining inventory, the PCjr is an excellent buy. The rate of new subscriptions coming to HCM from the PCjr user base is very much on the increase as other folks discover just what a great value Home Computer Magazine really is.

C-64 MERGE Program Used with BBS

Dear Sir:

Although I have owned a home computer for several years now, I only recently decided to expand with a disk drive and a modem, so I am very new to the areas of disk file handling and telecommunications. When trying to download files from one of the local BBSs, I became very frustrated because the terminal program I have will only transmit and receive sequential files. That meant that I could download a program and store it on disk, but since it was stored as a sequential file and I didn't know how to convert it back to a program file, I could not use it.

Well, HCM came to my rescue with the "Merge" program for the C-64 in the Tech Notes section of Vol. 5, No. 1. I simply loaded and ran the "Merge" program and entered the name of the file I wanted to convert, and I had a usable program that I could run to my heart's content. I don't know if you

Letters

to the Editor

CONTINUED

had this application in mind when you published the program, but it works beautifully.

Thanks. You have a great magazine. Keep up the good work.

Mike Poole
Topeka, KS 66603

How about that, Mike. You found a use that our staff hadn't considered. We wonder how many other people have discovered that the MERGE program can be used for this purpose. Of course, another more popular use is to generate your BASIC program with a word processor, save it as a sequential file, and then use the MERGE program to "load" it into the BASIC interpreter. Often using a good word processor will make it easier to find statements in your program, do block moves, etc.

MIDI and the TI

Dear Sir:

First, thank you for your very excellent magazine. Do you know of any MIDI (Musical Instruments Digital Interface) and software for the TI-99/4A? I would appreciate it very much if you would let me know of any distributor who sells things like that.

Walid El-Azem
Holbark, Denmark

Walid, we are currently searching for such devices ourselves to be covered in Home Computer Magazine and our new sister publication Music & Electronics magazine. We are even considering producing a do-it-yourself TI-to-MIDI kit to give the TI-99/4A some of the same capabilities as the new Yamaha CX5M MSX music computer. As we learn more about products in this area, we will be sure to pass word along to you through these pages.

Double-Spaced Printer Listings & More

Dear Sir:

A really great magazine (in its new format). I'm into my second copy purchased for cash at the local Walden's and B. Dalton stores (Vol. 5, Nos. 1 and 2). However, I'm probably one of the very last VIC-20 owners! I'll reserve my decision on a subscription for a while yet.

An addendum to Jack Ryan's letter in Vol. 5, No. 1, concerning double-spaced listings: Using the Cardco interface you can get double-spaced listings by using a file number of 128 or greater, i.e.,

OPEN 128,4,(SA):CMD 128:LIST

Use whatever secondary address (SA) you need for a particular print option. This should work on most Commodore-emulate interfaces, or you can do like I did and cut a small opening into the cover-case of the interface to expose the DIP switches and make any changes at will.

Another tip: On Commodore 64 and VIC-20 units, the following one-liner will make an excellent screen dump when used at a high number GOSUB:

```
63000 OPEN4,4,(SA):OPEN3,3,(SA):
PRINT"(HOME)":FOR I=0TO999:
GET#3,AS:PRINT#4,AS:NEXT:
CLOSE3:CLOSE4:RETURN
```

(Use 505 vs 999 on a VIC and abbreviate all key words to get it all on one line.)

Automatic Typo-Finder Wanted

Dear Sir:

I just recently discovered Home Computer Magazine, and I am very impressed. In fact, I have already sent in my subscription application, along with an order for some back issues. I would like to make one suggestion. It would be very nice for your readers who type in your programs themselves to be able to validate their typing. While I realize that this would take up valuable space, I think that the time saved by avoiding typing errors would be worth it. My idea is to publish short programs with each issue which could be used to validate the typing of each of your programs. The reader could type in the short checksum program (it should not be more than about 20 lines) and then use it to validate the typing of larger programs. The following is a very simple program to do the checksum calculation for the IBM PC. To use the program, type in the program to be checked, and save it as an ASCII file (save "file",A). Then run the checksum program on the ASCII file. Here is the program (with checksums):

```
10=00J4 10 DEF FNMD(A,B)=INT(A-INT(A/B)*B)
20=0U1Z 20 INPUT "FILE NAME TO CHECK: ",FILES
30=0T8P 30 OPEN "I",1,FILES
40=0O3K 40 IF EOF(1) THEN END
50=0H#8 50 LINE INPUT #1,LN$
60=0L@A 60 CODE=0
70=0M13 70 FOR I=1 TO LEN(LN$)
80=0Wft 80 CODE=CODE+(I*ASC(MID$(LN$,I,1)))
90=00jt 90 NEXT I
100=0RPr 100 DG$=""
110=0QPC 110 FOR I=1 TO 4
120=0MYJ 120 convert to base 64
130=0UnZ 130 DG=FNMD(CODE,64)
140=1PUn 140 get one digit
150=0SG2 150 CODE=INT(CODE/64)
160=0W3i 160 strip off one digit
170=0Pza 170 DG$=MID$("0123456789ABCDEFGHIJKL
LMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz@#",DG+1,1)+DG$
NEXT I
PRINT USING "#####" ; VAL(LN$);D
G$," print checksum
GOTO 40
keep going til EOF
```

Be assured, Robert, that we are working on methods that can be used to ease the key-in error frustration that some of our readers experience. Certainly checksum-type programs are high on our list. If other readers have other ideas on solving this problem, we'd like to hear from them, too.

The program simply computes a checksum for each line of the program, and prints the line number and checksum for each line. While there are probably many better methods of computing checksums, the method used here should suffice. It will result in a 24-bit integer which is then printed out in base 64. Using this somewhat-cumbersome method allows the line to be up to 256 characters long, and any character from 0 to 255 can be used. The maximum checksum would occur for a line of 256 characters, each having a value of 255. The checksum would be:

$$1*255 + 2*255 + 3*255 + \dots + 256*255 = 8388480$$

Note that this number can be expressed as a 24-bit binary number, which means that all possible checksums can be represented by a single 4-place base-64 number. Please feel free to use this program or some adaptation of it if you decide to implement some sort of checksum with your programs.

Robert R. Sloane
Lawrence, KS 66045

Finally, I think I'll try to transpose "The Organizer" to run on my VIC—then perhaps a subscription. Keep up the good work and maybe some other mags will get the idea!

Jim Rudd
Miami, FL 33186

We thank you very much, Jim, for your compact screen dump. We found, however, that it did not accurately duplicate our screen. Perhaps one of our readers has an idea of how to get this routine to do an undistorted screen dump. Considering the VIC-20's memory limitations, it will be a neat trick if you really can transpose "The Organizer" Program (which first appeared in Vol. 5 No. 1, and continued in Vol. 5 No. 2) to that machine.

Memory Full in One Line?

Dear Sir:

I typed your one line arcade game on my TI-99/4A in Extended BASIC. This was from Vol. 4, No. 5.

When I try to run it, I get the message "Memory Full in 1." What can I do to use this program?

Kristin Chotzinoff
Denver, CO 80218

We have tried to duplicate your problem, Kristin, but each time we key-in that one-liner it works just fine. Perhaps when keying it in, you struck a wrong key, making an error in the one-line program. As a possible aid to you, and for current readers who may not have that back

issue, we have repeated it below in a slightly different type font.

```
1 N=28 :: FOR X=4 TO N ::  
CALL SPRITE(#X,60+X,X/2,N,N,X,M) ::  
FOR Y=5 TO X ::  
CALL COINC(#Y,#4,N+M,C) ::  
M=M-C :: DISPLAY AT(4,9):M ::  
CALL JOYST(1,E,F) ::  
CALL MOTION(#4,-2*F,2*E) ::  
NEXT Y :: NEXT X
```

A Convincing Quiz

Dear Sir:

Congratulations for publishing a truly useful magazine. I own a C-64 and I am tired of magazines that focus on programming utilities and games. I have so many programming utilities that I don't even remember what half of them are for. And, if I wanted a game machine, I could have bought an Atari 2600. Give me useful software!

Don't get me wrong, utilities and games are important (I own many games, also). But it seems that most magazines have forgotten why computers are here—to make our lives easier. You folks hit the spot with "Quiz Construction Set" (that was my first issue, by the way). Finally—an educational program that utilizes a printer. I am studying electronics and this makes it easy to create hard-copy tests for myself. And my wife, who works with the retarded population, can create simple worksheets for her clients. She can't take our C-64 to work, so up till now educational programs were of no use to her.

Way to go! You've got my subscription to both magazine and disk with this letter. Keep it up!

Dan Braasch
Rolling Meadows, IL 60008

Dan, as you see with each of the issues that we produce, we try to balance coverage. Where as you appreciate productivity-type educational software, others still appreciate gaming software, and others are still looking for the ultimate software tool in a utility. We have great fun trying to balance every issue to give everybody something that they really want. Our goal is to make every issue of Home Computer Magazine so valuable that our readers save them as reference works.

IBM CONFIG.SYS vs HCM Dual Disk

Dear Sir:

I was very pleased with the recent article on adding a second disk drive to PCjr, and am happily computing with dual disks now. I have several hints I would like to share and some questions you may be able to answer. First the hints.

There was a slight problem utilizing the setup software published in the article when I installed my Microsoft Junior Booster with mouse and 128K memory expansion using DOS 2.1. The only way to install the extra memory is by adding the Microsoft memory device drive "MEMORY.SYS" to a "CONFIG.SYS" file. However, for reasons unclear to me, if a "CONFIG.SYS" file exists prior to the execution of the "AUTOEXEC.BAT" file listed in the article, the added memory on the Junior Booster sidecar is not installed. This problem

was solved by initially naming the "CONFIG.SYS" file "SWITCH.BAT" and replacing the "OFF.BAT" program-placekeeper file with another "dummy" file, the "CONFIG.SYS" file. Then, on initial power-up the system sees no "CONFIG.SYS" file until the "SWITCH.BAT" file is renamed "CONFIG.SYS". When "BOOT.BAT" is executed, "CONFIG.SYS" is now executed, and it does not interfere with the installation of the second drive. The resultant working "AUTOEXEC.BAT" file is below:

```
IF EXIST switch.bat GOTO first  
GOTO last  
:first  
RENAME switch.bat config.sys  
BOOT  
:last  
RENAME config.sys switch.bat
```

The "SWITCH.BAT" file, which is really the configuration file in disguise, then contains the following to load the new memory and clock-calendar:

```
DEVICE = MEMORY.SYS S:/nn V:/nn  
DEVICE = CLOCK.SYS  
(and any other configuration statement desired).
```

Charles R. Garcia
Sepulveda, CA 91343

Well, Charles, we really have to hand it to you for figuring out how to configure a system when using the dual-disk software described in "One for the Money, Two for the Slow" published in Vol. 4, No. 4 of Home Computer Magazine. Here's why your solution works. A CONFIG.SYS file is a batch-type file that installs certain machine-language driver programs. These programs allow for the recognition of extra memory, etc. When initially booting the system, CONFIG.SYS is always searched for first, so the drivers are installed before the AUTOEXEC.BAT file is run. The problem occurs when the dual-disk software from that article reboots the system after the CONFIG.SYS has done its installations. The MODBOOT.BAT routine uses its own hardware addresses for rebooting, which very likely overwrite some of the machine language installed by the CONFIG.SYS file. Therefore, this reboot could easily undo the work done by CONFIG.SYS the first time by overwriting the driver software. Again, congratulations on a great solution to a tricky problem.

Apple-to-IBM Data Moves

Dear Sir:

I previously owned an Apple IIe and have a large number of data files which were created on that machine. I sold the "II" and purchased an IBM PC/XT onto which I wish to transfer these same data files.

Please inform me if there is any hardware available that I can install in my PC/XT which will enable it to read from my Apple floppies and transfer the data to my Winchester.

If the above is not possible, please tell me how to go about transferring data directly from computer to computer (Apple II to IBM), since I have access to a colleague's Apple, and tell me exactly what equipment I would need to carry out such a transfer. I wish to accomplish

this transfer directly without using telephone/modem, if possible, since the telephone service in this part of the world is not exactly reliable.

I look forward to hearing from you.

S. P. Crow
Bombay, India

What you are going to require is a serial interface cable that will connect the IBM and Apple II computers. This cable will require an RS232 interface for each computer (even if you don't use a modem). The IBM will probably have one built-in, but the Apple (unless it is a IIc) will require some sort of serial interface card (such as the Super Serial card from Apple). Then you'll need to have a terminal emulator software package that operates on the IBM (such as "Personal Communications Manager" from IBM) and a terminal emulator software package that runs on the Apple (such as "Apple Access II" from Apple). With this combination, you can transfer any data files that are in normal ASCII code (text-type files). If any readers have done this sort of Apple-to-IBM transfer, let us know. Also, look to future "Home Computer Tech Notes" for more information.

Announcing . . .

Music & Electronics™
The Magazine of Creative Discovery
Through Sound Technology

Emerald Valley Publishing Co., publisher of Home Computer Magazine, is introducing a new magazine devoted to "the musician in all of us." Each bimonthly issue of Music & Electronics explores the many facets of personal music composition, performance, and education through the magic of computers and electronics. Editorial content spans a full range of subjects—from introductory music theory, to audio and digital recording techniques, to interfacing with home computers and home entertainment systems. The magazine offers abundant reviews on the latest electronic music products, as well as a bound-in demonstration record that allows listeners to actually hear the reviewed products and instructional examples. Music & Electronics is written for all ages and levels of musical expertise—from the novice to the professional.

Note: HCM readers may obtain subscription information by mailing a postcard to:

Music & Electronics
Emerald Valley Publishing Co.
1500 Valley River Drive, Suite 250
Eugene, OR 97401

by Randy Thompson
HCM Staff

*Working parents, students, executives,
and active citizens: Get a **View** of how
your **Day** will **Run**, and let the computer
help you manage your precious time.*

When you first **RUN** Run-Day-View, you will be presented with a main menu containing these 6 options:

- 1) Edit appointments
- 2) Edit phone numbers
- 3) Print routines
- 4) Load appointment file
- 5) Save appointment file
- 6) Exit program

This option allows you to create a new appointment file, or edit one that is already in memory. When you choose this option, you will get another menu. It consists of these 3 choices:

- 1) Set date and time
- 2) Edit appointment book
- 3) Return to main menu

You must choose this option before you can create an appointment file. Here, you are asked to input the month, year, and starting time for your date book. The starting time will determine the time at which each day of appointments will begin. In the sample printout in

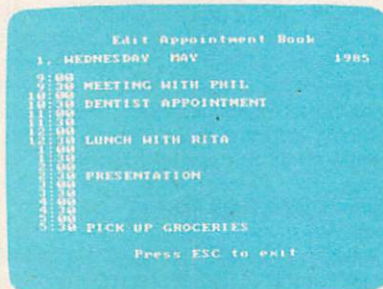
If you attempt to set the date more than once, you will be given the option to erase your current appointment data. This feature was included because appointments for one month will not usually coincide with appointments for another. If you answer **No** while in this mode, you will be returned to the Edit Appointments menu.

Here's where you'll spend most of your time. This option is for actually entering data. The editing screen is easy to use. It is set up just like a page from an appointment book, with the date at the top and the time displayed in half-hour increments along the side. From this screen you may enter appointments, input phone numbers, turn pages, and set markers (markers are explained next). Have a PTA meeting on the 7th? Put it in. Afraid you'll forget to visit Barb at the hospital on Sunday? Just type it in and make the computer remember for you. Turning the page will bring you to a new day in the book. The current day on which you are working will always be displayed at the top of the screen.

One of the most useful features of this program is its ability to set markers. Markers are used for accenting certain appointments. When a marker is set, a little flag appears next to that appointment when you list it on the computer screen. Markers also determine which appointments are included in a weekly summary (see the Print Routines menu). For instance, to help make sure that you don't forget that meeting with your department

Each machine will have its own method of accessing these different functions. Refer to the proper Control Capsule for your computer.

This option does just what it says.



This shows the Edit Appointment Book screen from the IBM version.



"You may even print out a folding appointment book small enough to fit in your pocket."

Edit Phone Numbers

Choosing this option will bring up the phone-list screen. Here you can enter and edit up to 18 phone numbers. This is a good place to put the phone numbers of people with whom you have appointments. For instance, if you're supposed to pick up some contact lenses on Tuesday, you would probably want to keep a record of the optometrist's phone number. Phone number lists are automatically sorted alphabetically.

Print Routines

Once you choose this option, another menu will appear. It is the heart of the program. With this menu you can create a variety of printouts that will display and organize your appointments in a variety of formats. The menu options are:

- 1) Print one page of appointments
- 2) Print the whole appointment book
- 3) Print a weekly summary
- 4) Print the current list of phone numbers
- 5) Return to the main menu.

1) Print page of appointments

This is an alternative to printing the whole appointment book. With this option you can print out just two days of appointments at a time (see Figure 1). If you make any changes to your appointments after printing the whole book, you may use this option to create a replacement page.

2) Print appointment book

Here you can print your own custom date books. An appointment book consists of one month of appointments. Each sheet of printer paper will have two days of appointments on it—thus two pages of the appointment book. (See Figure 3 for an explanation on converting your printout into a convenient pocket-sized book.)

3) Print weekly summary

A weekly summary is simply a calendar showing only 7 days (see Figure 2). You can begin this calendar on any day of the week you wish. If an appointment has its marker set (see Edit appointment book), it will appear as an asterisk followed by a number. To discover

Figure 1

APPOINTMENTS FOR: MAY
1, WEDNESDAY

8:00 Call Dr Bill for Jake
8:30 Drop off laundry

9:00 Work
9:30

10:00
10:30

11:00
11:30

12:00 Lunch w/disk salesperson
12:30

1:00
1:30 Meet w/ User Group Pres

2:00
2:30 Jake to Dr Bill

3:00 Grocery shopping
3:30

4:00 Pick up Jake
4:30 Pick up laundry

APPOINTMENTS FOR: MAY
2, THURSDAY

8:00 Chamber Commerce Brkfst
8:30

9:00 Work
9:30

10:00 Meet w/ General Manager
10:30

11:00 Meet w/ department heads
11:30

12:00 Lunch with Josephine
12:30

1:00 Appt w/ optometrist
1:30

2:00 Pick up cleaning/shoes
2:30 Shop for bday-Ian

3:00 Pick up Jake at school
3:30 Drop Jake at pool

4:00
4:30 Pick up Jake/groc store

Figure 2

WEEKLY SUMMARY STARTING
1, MAY 1985

WEDNESDAY 1

* 1 * 2

THURSDAY 2

* 3 * 4
* 5 * 6

FRIDAY 3

* 7 * 8
* 9

SATURDAY 4

* 10 * 11
* 12

SUNDAY 5

* 13 * 14
* 15

MONDAY 6

* 16 * 17

TUESDAY 7

* 18 * 19
* 20

*APPOINTMENTS FOR WEEK STARTING
1, MAY 1985

* 1 Meet w/ User Group Pres
* 2 Jake to Dr Bill
* 3 Chamber Commerce Brkfst
* 4 Meet w/ General Manager
* 5 Meet w/ department heads
* 6 Appt w/ optometrist
* 7 PTA Committee meeting
* 8 Sky diving lesson
* 9 Pick up Shari at school
* 10 User Group meeting
* 11 Pick up beer/wine
* 12 Picnic at Skinner's
* 13 Ian's first communion
* 14 First communion lunch
* 15 Visit Barb at hospital
* 16 Sky diving lesson
* 17 Tanning session
* 18 Data Processing meeting
* 19 Meet w/ Hilton Sales Mgr
* 20 PTA committee meeting

the appointment to which a particular number refers, look at the cross-reference list that is always provided below the weekly summary.

4) Print phone numbers

To get a printout of the phone-number list, choose this option. Each month's phone numbers are printed alphabetically, with the corresponding month listed at the top. You can carry this list with you, or simply set it by your phone.

Continued on next page

5) Return to the main menu

If you don't want to print anything, here's your panic button. This option will return you safely to the main menu.

Load Appointment File

Choose this option to load a previously created appointment file. Files consist of one month of appointments and phone numbers.

CONTROL CAPSULE Run-Day-View



KEY	FUNCTION
<i>Edit Modes:</i>	
—	Exit editing screen.
SHIFT INSERT	Insert character.
DEL	Delete (backspace).
Cursor down or Return	Next item.
Cursor up	Previous item.
Cursor left	Cursor left.
Cursor right	Cursor right.
<i>Edit Appointments Mode:</i>	
F1	Turn forward a page—increment screen by a day.
F2	Turn back a page—decrement screen by a day.
F3	Set markers.
F5	Input phone number.
F7	Jump a page.

CONTROL CAPSULE Run-Day-View



KEY	FUNCTION
<i>Edit Modes:</i>	
Esc	Exit editing screen.
Backspace	Backspace.
Del	Delete character at cursor.
↑	Previous item.
↓	Next item.
←	Cursor left.
→	Cursor right.
<i>Edit Appointments Mode:</i>	
F1	Turn back a page—decrement screen by a day.
F2	Turn forward a page—increment screen by a day.
F3	Jump a page.
F4	Set markers.
F5	Input a phone number.

CONTROL CAPSULE Run-Day-View



KEY	FUNCTION
<i>Edit Appointments Mode:</i>	
1	Jump a page.
2	Set markers.
3	Enter an appointment.
4	Input a phone number.
5	Exit editing screen.
<i>Edit Phone Numbers Mode:</i>	
1	Input a phone number.
2	Exit editing screen.

Save Appointment File

This allows you to save a month of appointments. You must have a file in memory before choosing this option.

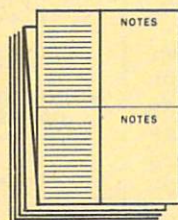
HCM

CONTROL CAPSULE Run-Day-View

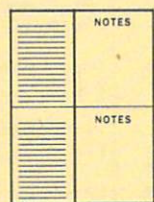


KEY	FUNCTION
<i>Edit Modes:</i>	
Esc	Exit editing screen.
↑	Previous item.*
↓	Next item.*
←	Cursor left.*
→	Cursor right.*
<i>Edit Appointments Mode:</i>	
Control D	Turn back a page—decrement screen by a day.
Control F	Turn forward a page—increment screen by a day.
Control P	Jump a page.
Control Q	Set markers.
Control Z	Input phone number.
<i>*SPECIAL FOR II+ USERS:</i>	
Control J	Previous item.
Control K	Next item.
Control H	Cursor left.
Control U	Cursor right.

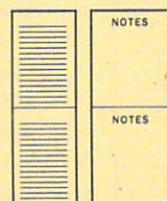
Figure 3
Converting The Printout
Into A Book



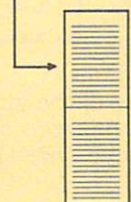
Neatly fold the paper back into a stack (along the horizontal perforations).



Cut along the vertical line down the center of the paper.



Keep excess paper for notes.



Fold top portion back (away from you) on horizontal line.

Fold forward (toward you) along fanfold crease.



Continue folding each page until a small (approximately 3" x 5") book is formed. Bind at the top with a clip or staples.



TRIG-TRIX

Trigonometry can help you calculate the unreachable heights and complex forces in nature. This program will help you fathom the depths of this eternal triangle of mathematics

by **Roger Wood**
HCM Staff

Have you ever wondered how surveyors figure out the height of a mountain peak? Do they climb up to the top, drill a hole to the bottom, and drop a tape down the hole? How silly, you say. But by what mysterious calculations do they arrive at a true figure without direct measurement? *Trig-Trix* will help make this measuring process no longer seem so mysterious. This program exercises the fundamentals of trigonometry, a form of mathematics that uses the triangle as a basis for measuring distances—either in the abstract, or in the physical world.

A major cornerstone in trigonometry is the relationship of a right triangle's sides to its angles. These boil down to three basic relationships: Sine, Cosine, and Tangent. *Trig-Trix* is designed to administer problems dealing with these basic line and angle relationships by addressing right triangles, the Law of Sines, and the Law of Cosines. When you run the program, you are presented with this menu:

- 1.) RIGHT TRIANGLES
- 2.) LAW OF SINES
- 3.) LAW OF COSINES
- 4.) END PROGRAM

1) RIGHT TRIANGLES

When you select option 1, you will see this submenu:

1. DETERMINE SIDES
2. DETERMINE ANGLES
3. RETURN TO MAIN MENU

The triangle in Figure 1 is for reference when using either of the Right Triangle options.

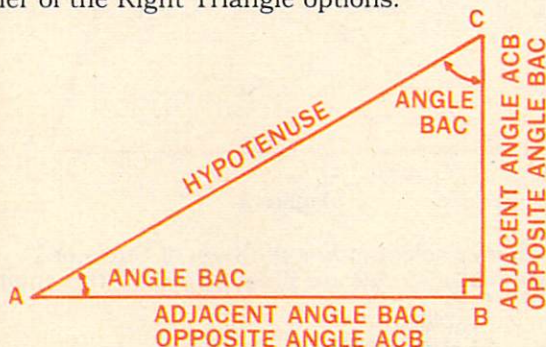


Figure 1

The Sine (SIN) of an angle can be defined as the ratio between the side opposite an angle and the hypotenuse of the right triangle. (The hypotenuse is the angle opposite the right angle—line segment AC in Figure 1). For example, the Sine of the angle defined by points BAC

is the length of line segment BC divided by the length of line segment AC.

Likewise, the Cosine (COS) of an angle is the relationship between the side adjacent to the angle and the hypotenuse. The Cosine of angle BAC is AB divided by AC. Finally, the Tangent (TAN) of an angle is the opposite side over its adjacent side. BC divided by AB is the Tangent of angle BAC. Here is a complete list of these relationships for the triangle in Figure 1:

SIN of angle BAC = BC/AC	SIN of angle ACB = AB/AC
COS of angle BAC = AB/AC	COS of angle ACB = BC/AC
TAN of angle BAC = BC/AB	TAN of angle ACB = AB/BC

Whether you choose to Determine Sides or Determine Angles, the program randomly selects one of the two non-right angles, and two of the sides. It then randomly selects appropriate values for the two "known" parts, and prompts the user for the unknown. For example: to determine a side, AC, you may be given another side, BC, and angle BAC. A typical formula for solving this problem would be:

$$AC = (BC/\sin(\text{angle BAC}))$$

If BC = 17, and angle BAC = 5 degrees, the proper expression would be:

$$AC = (17/\sin(5)) \text{ or } AC = 195.05$$

Enter Answer As Numbers Or Expressions

All options give you chances to enter the value that you think is the answer (rounded to two decimal places), or an expression which would result in the correct value. For example, if the answer is 20, you could enter 20, 10*2, or SQRT(400).

The point here is not to test your arithmetic, but to exercise your ability to logically think through a problem. Once you have arrived at a logical expression, you may even use a hand-held calculator to determine a value—but you can also use the computer as a calculator

by entering the expression instead of a final answer. At that time, the computer will follow your expression, arrive at a figure, and check it against the correct value.

The algorithm used by the program to evaluate your input is a modified version of the one employed in *It Figures!*, which appeared in *HCM* Vol. 5, No. 2. To fit this format, you must enter each expression using the following guides: Your expression can include any numeric value within the range of your machine's BASIC, including decimal numbers—but you cannot use scientific-notation format (e.g., 10000 is OK, but 10 E3 will provoke a SYNTAX ERROR message). You can include any of five different operators: +, -, *, /, or ^ ([|] on the Commodore 64). You may also use parentheses. The expression you enter can include several functions:

- (1) The three trig functions: SIN, COS, and TAN;
- (2) The three corresponding inverse functions: ASN (Arcsine), ACS (Arcosine), and ATN (Arctangent); or
- (3) The SQR (square root) function.

Each of these functions will operate on the value immediately following the parenthesis. Attempts to type-in any other letters to an expression, or to use a value that is illegal in a function (such as trying to get the square root of a negative number), will result in an error message.

One major difference between the expressions you enter in this program, and those of a similar BASIC language expression is that there is no "precedence of operators" as there is in BASIC—the value is determined in a left-to-right fashion. The program lets you use parentheses, however, so you can force the precedence of operators. That is, if you want to add 6 plus 3 and multiply the sum by 4, any of these would work:

$(6 + 3) * 4$, or $4 * (6 + 3)$, or $6 + 3 * 4$

In a BASIC program, only the first two would give the same answer that *Trig-Trix* would give. The third would result in an answer of 72 if entered in a BASIC program, instead of 36, which would result from our program.

Other Options

If you enter the correct answer to a problem, the computer will display the final numeric answer. If you are unable to give the correct answer in three tries, the computer will then display the correct answer. You will then be given these choices:

1. DO A PROBLEM OF THIS TYPE
2. ENTER VALUES
3. EXIT

If you select 1, you will be given another problem like the one you just finished. For example, if you just completed a problem determining angles, you would be given another determining angles. It may, however, be a different angle, and the sides you are given as "knowns" may be different, thus forcing you to employ a different function. In any case, the values will be a different set of random choices.

If you choose 2, Enter Values, then you will be presented with a problem identical to the one you just did, only now you can ask the computer to figure it out for you. By entering several different values in a given problem and noting the answers, you can gain insight into how the various functions operate. After each of these "Enter Value" problems, you will be returned to the same menu to either do another problem of the same

type, take another opportunity to Enter Values on this problem, or Exit. This menu is always presented after you have completed a problem on any level.

One more note on entering values: Because each exercise uses a specific figure, there will be a certain range of legal values. Entering an illegal value—say, one that would produce a triangle that couldn't be drawn on paper—will cause illogical results, error messages, or have other unexpected effects on the program. Part of the exercise entails figuring out what the legal range should be in advance of actually entering a value.

Higher Levels—The Two Laws

After you have honed your skills using the lowest level, you can work on two other important laws in trigonometry: the Law of Sines and the Law of Cosines. These two laws greatly increase the power of trigonometry to overcome the problems of physical measurement—whether on land, at sea, in the air, or

in outer space. See the sidebar "*Trig-Trix in the Real World*" for some ideas on how such problems can yield to a trigonometric solution.

2) LAW OF SINES

The Law of Sines works for any triangle, not just right triangles. Stated simply, the Law of Sines decrees that the Sine of an angle in a triangle is to its opposite side, as the Sine of any other angle is to its opposite side. This allows for a much more indirect method of finding unknown values. For example, in Figure 2 the following relationship holds:

$$\frac{\text{SIN angle ABC}}{\text{AC}} = \frac{\text{SIN angle BAC}}{\text{BC}} = \frac{\text{SIN angle ACB}}{\text{AB}}$$

If angle ABC = 60, and sides AC = 10 and BC = 5, then it is a simple matter of using the following equation to determine angle BAC.

$$\text{BAC} = \text{ASN}(5 * \text{SIN}(60)/10)$$

$$\text{BAC} = 25.66 \text{ degrees}$$

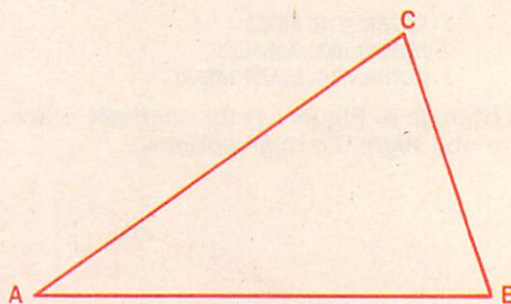


Figure 2

When you select either the Law of Sines or Law of Cosines options, you are presented with this menu:

1. DRILL
2. CHALLENGE
3. RETURN TO MAIN MENU

If you select 1, you will be presented with a problem similar to the one shown above, and will be given three chances to solve it, just as in the earlier option. You will also be asked whether you wish to continue with another problem, or Enter Values when the problem is solved.

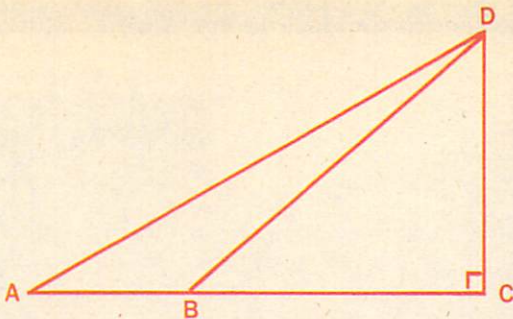


Figure 3

When you select 2, Challenge, you will be given a problem that will tax your ingenuity in using the Law of Sines. This problem centers on Figure 3, which appears as a double right-triangle, one inside the other. Your challenge is to find the common height (CD) of both triangles, given AB, angle BAD, and angle CBD. To do this, you must put together what you've learned from the lower-level problems and enter the correct expression or final answer.

3) LAW OF COSINES

The Law of Cosines option works very similarly to the Law of Sines option. You will once again be asked whether you wish to Drill or do the Challenge. (Drill uses a diagram like Figure 2.) The major difference between the two laws is that the Law of Cosines allows you to determine a side if you know the other two sides and their included angle. For example, if you know AB, AC, and angle BAC, you can find BC. Here's the Law of Cosines formula as applied to this problem:

$$BC = \sqrt{(AB^2) + (AC^2) - (2 * AB * AC * \cos(\text{angle BAC}))}$$

For example, if you are presented with: AB = 10, AC = 5, and angle BAC = 30 degrees, then you can get the correct answer by entering:

$$\sqrt{(10^2) + (5^2) - (2 * 10 * 5 * \cos(30))}$$

which equals 6.20 when rounded to two decimal places. You may also Enter Values to quiz the computer on any problem you wish.

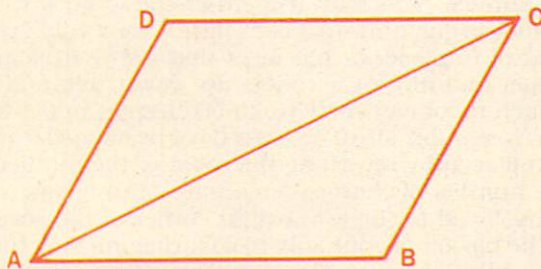


Figure 4

In addition, the Challenge option is available if you wish to try a more difficult problem. In this option, the problem presented centers on Figure 4, a parallelogram bisected by a diagonal line. You must find the length of this diagonal (AC), given AD, AB, and angle ABC. Remember that, in a parallelogram, opposite sides are equal in length.

HCM

HCM Glossary terms: expression (mathematical), vector (physics), hypotenuse, algorithm.

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

TRIG-TRIX IN THE REAL WORLD

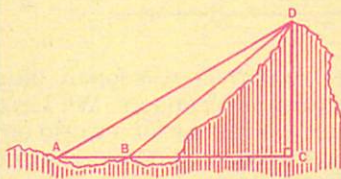
Trigonometry has a very practical purpose in the real world. Abstract calculations based on the triangle can help us measure many things that are difficult or impossible to gauge directly. Here we show you just a few examples of everyday trig tricks:

1. How high is that tree?



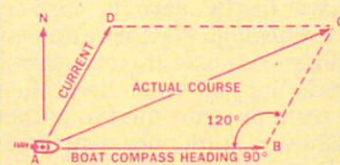
To find the height of any tree, simply regard the tree as the vertical line of a right triangle—line BC in this drawing. Now pick a spot well out from the tree base and measure that distance (try to make it in a whole number of feet). Take a simple protractor, hold it on a level plane on the spot you've picked (point A) and use a straight-edge to sight a line from the center of the protractor to the top of the tree. Measure the angle along the straight edge with the protractor. You now have your two givens, line AB and angle BAC. To find BC, use a formula based on the right triangle problems in the Trig-Trix program, such as: $AB = BC / \sin(\text{angle BAC})$

2. How high is that mountain?



If you worked through the Challenge option in the Law of Sines portion of Trig-Trix, you will probably recognize the drawing shown here. Finding the height of a mountain peak is an excellent example of an abstract formulation leading to a practical solution. In this scenario, we need only know two angles, BAD and CBD, and the distance AB. We won't give you a formula in this case, because it would spoil all the fun of the Challenge. However, once you master this program, you will uncover the mystery of how a surveyor measures the mountain without even climbing it. You could even do it yourself!

3. With my ship pointed one direction at a certain speed, and a current dragging it off its heading, what's my actual speed and course direction?



This is a common problem on the water. If you know the speed and direction of the current, as well as the heading and speed, you can calculate the actual speed and course using the Law of Cosines. The solution to this problem is an application of what is known in physics as vectors. The current's direction (angle NAD) and speed (AD in miles per hour) form one vector component, and the ship's heading (angle NAB) and speed (AB in miles per hour) form the other. The actual speed is found by determining the length of the diagonal of the parallelogram (AC) formed by these two components. The actual direction of travel is determined by the angle NAC.

This problem is very similar to the Challenge for the Law of Cosines in Trig-Trix; by stretching the logic of that exercise a little further, you should be able to navigate this problem as well.

HCM



This game is based on an old favorite—*Lost Ruins*—originally published in HCM's forerunner, *99'er Magazine*. Enhanced by added sound and colorful graphics, even our long-time fans will greatly enjoy this much-improved version.

Archeodroid

by **B. J. Bruns**
and the HCM Staff

The year is 9999. Interstellar travel is now commonplace. Civilizations have risen and fallen through the millenia as human beings have ventured into the distant galaxies. But what of mankind's past?

What is this planet, Earth? To this lonely blue pearl we now return as strangers. We have literally searched the heavens for a world only our distant ancestors have seen. And now this same Earth hides the very key to unlocking our past: *garbage!*

Looking Back

Despite fantastic technological achievements, the human race has lost track of its origins. Even the location of the home planet, referred to in ancient space legends as "Earth," has been lost to antiquity.

Archeological expeditions are currently combing the heavens for the lost planet. Fame and fortune await the lucky explorer who brings back convincing evidence that the location of mankind's birthplace has been found.

You might be that lucky explorer—as commander of a computer-guided spaceship sent to explore a distant part of the galaxy. After a long, uneventful search, you have finally landed on a planet that fits the description of the ancient legends and songs. Could this be the planet of your ancestors? The answer lies beneath the surface.

With you are three huge archeological-exploration androids, known as Archeodroids, who do the dangerous and difficult excavation work; but you must direct their search. Your robots have been programmed to gather up the twisted remains of Chevies, junk piles, and fossilized skeletal remains, but they must return them to the ship, or all of the work will be in vain.

The lure of unearthing just a few more treasures keeps urging you onward. Suddenly, one of your three robots becomes trapped in a cave-in as your lucky streak abruptly ends. You immediately send down another of your precious Archeodroids to salvage what it can—only this time you make a solemn space-vow to be more careful with your two remaining links to fame and fortune.

At The Site

The ruins that your robots will be rummaging through are a tangled mess of debris bordered by a hard-rock containment wall, which was built around the site by

fanatical natives who feared technology. You have a straight-on, cut-away view of the site that allows you to see where the artifacts are positioned. You must maneuver your Archeodroids toward deposits of artifacts by blasting tunnels through the ruins. The deeper you excavate, the more difficult it is to climb back out, due to the constant cave-ins brought on by your blasting. Thus, you must make your robots position their blast charges very carefully, or the artifacts that you seek will be damaged during the digging.

Starting To Dig

As the game begins, you have just touched down on the planet and dispatched your three Archeodroids to the surface. You may use either the keyboard or the joystick to guide one robot at a time. Check your system's Control Capsule for the keys that fire the excavating blaster and move the robots up, down, left, and right.

Each robot can carry up to 10 charges in the blaster at a time. After all 10 charges have been used, they can be replaced by returning the robot to the mother ship. The number of charges remaining at any point in time is displayed in the lower-right corner of the screen.

The blaster is your only tool for digging new tunnels. Each blaster charge clears five squares of earth in the direction that the robot was moving when you gave the order to set off a charge. As your Archeodroid digs deeper and deeper into the planet's surface, the danger from cave-ins becomes more serious. Each charge you detonate causes the layers of rock and garbage above you to shift. It's easy to become trapped, so watch your escape route as you tunnel.

The blaster will not detonate if the charge would destroy an edge-piece of the excavation area. You must move out of range of the wall. Edge-pieces correspond to the hard-rock containment wall that was built around the site in ancient times. The only exceptions to this rule are the five edge-squares in the top-center of the screen; these are your only entry and exit points from the tunnels you create.



Charging Up

The charges are not only used for blasting tunnels—they also provide the robots with power for movement and the means for gathering artifacts. When a robot runs out of charges, a small amount of residual energy enables it to return to the ship. If, however, it does not return fast enough, the robot's internal circuitry decays, and it ceases to function. Also, the more blasting the robots do, the greater their danger of being buried by a cave-in. On the loss of an Archeodroid, its last known position is marked with a cross, and any artifacts it has collected remain below. They can only be retrieved by another robot that reaches the location of the cross and brings the artifacts to the mother ship.

The Archeodroids can move freely through any tunnels already excavated, or through areas where artifacts are buried. Artifacts may be picked up from the ground or from inoperative robots by moving an active robot into the squares containing the items.

When an Archeodroid returns to the ship, you will receive points for what it has collected. If part of an object is destroyed by an explosion, you will still get partial credit for successfully retrieving what is left of it.

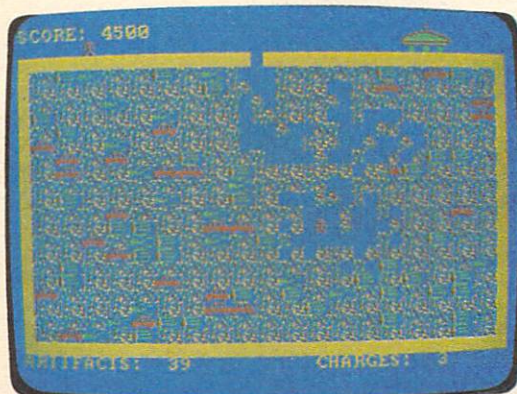
The values of the three types of artifacts depend on their archeological significance:

TYPE	VALUE
Skeleton	10
Car	6
Junk Pile	2

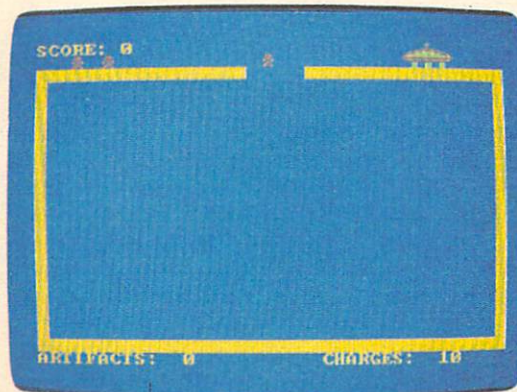
A running total of these values is displayed in the lower-left corner of the screen.

The more charges remaining in the robot, the more points you receive for its artifacts. You can return an Archeodroid to the mother ship at any time and deposit artifacts—but you can only get a new set of charges when your Archeodroid has used all of its previous charges. So, as you run low on charges, be sure to plan an escape route to prevent a cave-in from trapping your robot below ground.

When you've made a clean sweep of one area (one screen), you are transported to another unexplored region and challenged to unearth more evidence that you have found mankind's ancient birthplace.



This photo from the IBM version shows an Archeodroid in the center of the site, and gives you an indication of how much excavation is necessary to retrieve the artifacts.



This photo shows the site after all of the debris and artifacts have been removed. At the top of the screen you can see the 5 squares that are your only entry and exit points. Around the perimeter are the indestructible edge-pieces that form the site's containment wall.

"With you are three huge archeological-exploration androids, known as Archeodroids, who do the dangerous and difficult excavation work; but you must direct their search."



CONTROL CAPSULE ARCHEODROID

KEY	FUNCTION
I	up
J	left
K	right
M	down
space bar	blaster



CONTROL CAPSULE ARCHEODROID

KEY	FUNCTION
↑	up
←	left
→	right
↓	down
space bar	blaster



CONTROL CAPSULE ARCHEODROID

KEY	FUNCTION
V	up
G	left
H	right
B	down
K	blaster



CONTROL CAPSULE ARCHEODROID

KEY	FUNCTION
E	up
S	left
D	right
X	down
Y	blaster

MINE OVER MATTER

by William K. Balthrop
and the HCM Staff

Modern mining has left behind the sieves and picks that were the trade of 19th-century gold miners. Today, good business skills, timely reclamation, and megabucks are the tools for success.

Most folks would call the area desolate, barren—a few dry shrubs here, a jackrabbit there, some foothills in the distance. But you thought the area was perfect, and after doing some survey work, found that you were right. Now your new mill is constantly chug-chugging away—the only sound to be heard for miles except for those coming from the adjoining deep pit being mined of its modern-day treasure: uranium. You knew when you started that you couldn't lose—the ore was high quality and close to the surface, and the environmental impact of mining the area was so little that the overall cost was too low not to invest. So you did.

Mine Over Matter simulates uranium mining operations for up to 4 people. Each player can have up to 5 uranium mines at one time in the survey, construction, and production phases. If you close one or more of the five, you may then open others. The object of the game is to make money from your mines, beat your opponents, and keep from contaminating the land.

This program's playing time can range from 20 years (a turn represents 1 year) to 140 years. As in real life, your mine will have a life cycle of 4 phases: (1) a survey of the land, (2) construction of the mill to process the ore, (3) ore production (the longest phase of a mine's life), and (4) closing the mine and reclaiming the land. No further mining operation may ever take place on a site after this last phase is complete. Each phase takes a minimum of 1 year.

Surveying Phase

You start off with a cash reserve of \$2,500,000. A survey's cost depends on how deep the drill sample has to go. The maximum depth for any mine is 299 feet, and it will cost \$35,385—the minimum survey cost is \$1000, even if the uranium is at the surface. You will not know the depth until the survey has been taken. In addition to the depth of the uranium, a survey will tell you the purity of the ore, and the environmental impact of mining at the selected location.

The ore purity ranges from 0 percent (no ore found) to 99 percent. The purer the ore, the easier (cheaper) it will be to extract the uranium.

The environmental impact of setting up a mining operation is crucial to your mine's success. The environmental factors also range from 0 to 99—the higher the factor, the worse a mine would affect the environmental situation. Due to mining and environmental laws, higher factors mean higher reclamation costs.

The Main Menu:

1. SURVEY
2. PRODUCTION
3. REPORT
4. NEXT TURN

To do surveys of the area on the screen, select option 1 from the main menu. Your current CASH balance, the survey COST, ore QUALITY and DEPTH, and the ENVIRONMENTAL impact factor will be displayed in addition to brief instructions on how to use this section. Move a cursor around the screen to select a location for your survey. Once in position, press the spacebar to do a survey. If you do not wish to set up a mining operation on this site, then simply move the cursor off the location. A flag will be left behind indicating that a survey has been taken there. For a \$1000 fee, you may resurvey an area that has already been surveyed.

If you would like to stake a claim after surveying, press [ENTER] or [RETURN]. The program will then return to the main menu, and a marker identifying the mine as yours will be placed on the screen.

Mill Construction

Before you can produce ore, you must construct a refining mill to convert the raw ore from the mine into "Yellow Cake." You may only select a mine for construction if the mine either has a *completed* survey, is in the middle of or has just completed construction, or is in production. You may not start construction on a mine which either has a survey in progress, is in the middle of its reclamation phase, is out of ore, is already closed, or is closed due to contamination. Therefore, you may only start construction at least one year after the survey.

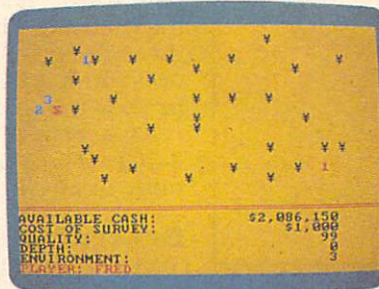
To select this option, first choose PRODUCTION from the main menu. The Production menu asks you to select one of the following:

1. SET PRODUCTION RATE
2. START RECLAMATION
3. EXPAND MILL
4. EXIT

Special thanks for technical advice goes to E. Frank Schnitzer, environmental specialist for the Oregon Department of Geology and Mineral Industries Mine Reclamation Program.



The survey screen from the IBM PCjr version of *Mine Over Matter*. The flags mark spots where previous surveys have been taken, and the numbered boxes represent mining sites, which are color-coded by player. The statistics for a survey's site are listed below the map when one is chosen.



Production Phase

On the turn after you've constructed or expanded a mine, you should set the mine's production rate. To get a mine into production, or to change the mine's current production, select the **PRODUCTION** option from the main menu; and then **SET PRODUCTION RATE**, option 1, from the production menu.

You will be asked to select a mine. It must be in production already, or have completed construction of the mill. A mine cannot produce while it's being reclaimed or closed, or after it has been closed for contamination.

After selecting a mine for production, a list of the mine's vital statistics will be displayed:

CASH:
MILL SIZE:
MAX. PROD.:
QUALITY:
DEPTH:
ENVIRONMENT:
INCREASE CONSTRUCTION:

PRODUCTION FOR:
CURRENT PRODUCTION:
BARRELS/YEAR:
VALUE/BARREL:
GROSS:
NET:
MAXIMUM PRODUCTION:
NEW PRODUCTION:

These figures will help you in determining how much money to spend on construction. The size of your mill (shown in dollar values) will determine how much ore can be processed. If your mine creates more ore than the mill can safely handle, you may end up with a contamination problem—resulting in the shutdown of your mine and the loss of your reclamation bond.

You can come back to this section at any time to increase the size of your mill, even after the mill is in production. However, *every time you do construction on a mine, you will not be able to produce any ore there for the rest of the year (until your next turn).*

You can find out the status of any of your mines by choosing the **REPORT** option from the main menu. It will list the numbers of your mines and their operating status: **NO MORE PRESENT**, **SURVEY IN PROGRESS**, **SURVEY COMPLETE**, **UNDER CONSTRUCTION**, **CONSTRUCTION COMPLETE**, **IN PRODUCTION**, **OUT OF ORE**, **RECLAMATION STARTED**, **RECLAMATION DONE**, **MINE CLOSED** or **CLOSED FOR CONTAMINATION**. Choosing one of your active mines will take you to this checklist:

MILL:
BOND:
BARRELS PRODUCED:
GROSS:
NET:
QUALITY:
DEPTH:
ENVIRONMENT:

When you start construction, the state government will assess the estimated cost of reclaiming the land (the cost of returning the area to its original condition). The mining company (the player) will have to put up a bond for this value. If the mining company should fail to reclaim the land, then the state will use the money from the bond to do so. If the mining company does reclaim the land, then the company (player) will get its bond money back as soon as the reclamation is complete (on the next turn for the purpose of the game.) However, in reality (depending on the state) the mining company probably would not get its bond back until vegetation returned to the area.

You will be asked to indicate how much money you want to spend (your operating overhead) on production. The more you spend on production, the more ore you will take from the ground, up to the limit established by the size of your mill. It takes a lot of trial-and-error to determine the best production rate. You should pay close attention to the *maximum production* rate, for if you *match or exceed* this value with the amount of ore you are extracting, you could easily end up with a contamination problem. (In addition, the program may generate a contamination problem at any mine at random, just to keep the game from becoming too controlled and/or predictable.)

The ratio of dollars to barrels of "Yellow Cake" per year is based on the depth of the mine, the environmental impact on the area, the ore quality, and the quantity of ore remaining in the ground. The quantity of ore left in the mine will offset production because as the quantity gets lower, it will cost you more money to get it out of the ground (the mine may be deeper, or the ore may be more widely dispersed).

Reclamation And Closed Phase

If a mine has run out of ore, it will automatically shut down. The mine's owner has 3 years (3 turns) in which to return the mined land back to its original state. Heavy metals and the radioactive sludge pond from the mill will be buried, and grass and trees will be planted. This phase takes one year (one player turn) to complete. Once this phase is complete, the mill will be closed forever, and the player will get back his or her reclamation bond from the state.

Place a mine into the reclamation phase by first selecting the **PRODUCTION** option from the main menu. Then select **START RECLAMATION** from the production menu. You will be asked to select a mine. After choosing a mine, you will be notified of its reclamation cost based on the environmental impact factor. If you do not have enough money for the reclamation, the mine will not start this phase.

MINE OVER MATTER SCORECARD

Player:	Starting Cash:
Year:	Ending Cash:
	Total Profit (Loss):
Mine #	
Mine Cost: \$	Quality:
Bond: \$	Depth:
Bar. Prod.:	Environment:
Gross: \$	Bar/Year:
Net: \$	Value/Bar.: \$
Max. Prod.:	Curr. Prod.:

CONTROL CAPSULE
Mine Over Matter

KEY	FUNCTION
I or up	Move survey cursor up.
J or left	Move survey cursor left.
K or right	Move survey cursor right.
M or down	Move survey cursor down.
SPACEBAR	Select location for a survey.
RETURN	Lay a claim on the location.
ESC	Exit from current operation.

CONTROL CAPSULE
Mine Over Matter

KEY	FUNCTION
E	Move survey cursor up.
S	Move survey cursor left.
D	Move survey cursor right.
X	Move survey cursor down.
SPACEBAR	Select location for a survey.
RETURN	Lay a claim on the location.
—	Exit from current operation.

CONTROL CAPSULE
Mine Over Matter

KEY	FUNCTION
UP	Move survey cursor up.
LEFT	Move survey cursor left.
RIGHT	Move survey cursor right.
DOWN	Move survey cursor down.
SPACEBAR	Select a location for a survey.
RETURN	Lay a claim on the location.
ESC	Exit from current operation.

CONTROL CAPSULE
Mine Over Matter

KEY	FUNCTION
E	Move survey cursor up.
S	Move survey cursor left.
D	Move survey cursor right.
X	Move survey cursor down.
SPACEBAR	Select location for a survey.
ENTER	Lay a claim on the location.
FCTN 9	Exit from current operation.

The reclamation cost will be only about 40 percent of the cost you put up for the state reclamation bond. After one turn in reclamation has passed, you will get your bond back—so, there is a great benefit in reclaiming the land. The amount of money you get back from the state on your bond can run anywhere from \$50,000 up to double the amount of money you have invested in your mill (potentially millions).

If you believe a mine is not worth keeping open, even if there is ore remaining, you may reclaim and close the mine at any time using the above procedure.

General Operating Tips

There is one factor involved that the survey does not tell you; how much uranium is really down there. You may locate an ideal mining site, build a mill, and start production to find that you have exhausted all of the ore in the first 2 or 3 years of operation. A typical mine may hold out for 20 to 40 years, depending on how much ore is in the mine, and how fast you take it out.

In determining a location for a mine, you will want to choose an area where you will get the most for the least amount of money. Choosing a location with a high quality of ore and a shallow depth is very critical. Just as important is the environmental impact factor—it could make a major difference in your cash reserves at the beginning of a mine's life, during construction, and near the end during reclamation. It also affects the cost of mining the ore (the production overhead).

To assist you in keeping track of all of the factors involved in your mining business, you may want to write down or photocopy the above scorecard so that you have enough for each player's mines. Use a pencil—you'll need to change your cash and other figures with every turn. You might also consider using a new scorecard for every mine for every turn. Of course, each player will not need to keep the information on the top half of the card on every card.

And if you're really engrossed in a long-playing, competitive game but you've got to leave for awhile, you can stop and save the game and return to exactly the same spot later. (Sorry, the TI-99/4A has too little memory to accommodate this option.) Between player turns, simply press the exit key appropriate for your system. Before the program leaves, it will first ask whether you want to save the game, and then whether you want to exit the program. If you say NO to the latter, it will ask whether you want to continue the current game or start a new one. If you say YES to exiting, the program will stop.

HCM

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

**MOVING?**
Don't Miss Out On
Any Issues Of**HOME COMPUTER**
magazine

Send us a Change-of-Address Card
(available at any Post Office)
6-8 weeks prior to the move.

Be sure to include both the old & new address, plus the alphanumeric code above your name on the mailing label.

Please send this
information to:

Home Computer Magazine
P.O. Box 70288
Eugene, OR 97401

Here they are . . . the best of the one-line programs that we have received since printing the third "HCM One-Liners" column in *Home Computer Magazine* Vol. 5, No. 3. Although many interesting programs were submitted, we have selected what we felt were the best four (one for each brand of computer covered in our magazine) of those that arrived prior to this issue's press date. 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 \$50 check for sharing their ideas with our readers.

A Colorful Bar Graph

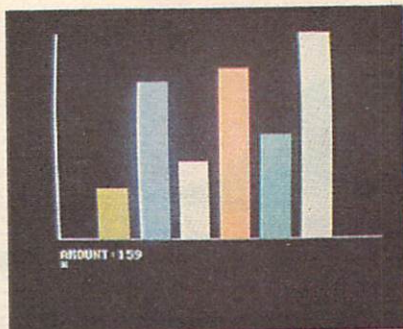
[Applesoft BASIC on the Apple IIe, IIc]
Dear HCM:

This is my HCM One-Liner. It makes multi-colored bar graphs using only one line of code. The maximum number that can be entered is 159. The program displays a prompt to signify that it is finished. An "R" or "r" in response to this prompt will erase the current graph and start another. Any other character entered will end the program. The graph can be BSAVED by entering:

BSAVE [GRAPH.NAME],A8192,E16383.
The program should be entered without spaces.

Donald W. Scott, Jr.
Buffalo, NY 14207

```
1:HOME:HGRO:HCOLOR=3:HPLO:
159:FOR TO 1:159:TO 6:READ
UC:HCOLOR=C:VTAB 22:INP
UT:AMOUNT:":N(I):FORT=1
*35 TO 1*35+25:HPLOTT,
160-N(I)TO 1,160:NEXTT,
I:DATA 1,2,3,5,6,7:RES
TORE:GET Y$:IFY$="R"OR
Y$="r"THEN 1
```



An Eight-Car Race

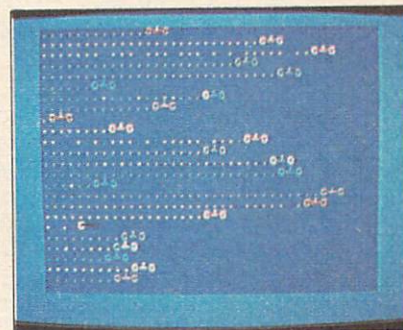
[Commodore 64 BASIC on the C-64]
Dear Sir:

This program races eight cars across the screen. The one traveling the greatest distance wins. Press the (RETURN) key to race again. Because the C-64 accepts only two program lines, there wasn't enough room for handling the scores for the winning cars. You'll have to keep track of the scores yourself. [NOTE: The graphic symbols are in the

order of their appearance: Shift W,
Commodore E, Shift W, Cursor Up.]

Joseph Potter
Leicester, MA 01524

```
1:AS$="W"SHIFTO:WTR:CMDRE
SHIFTO:WTR:CMDRE
ORT=1TORND(1)*35:CHRS(
X+148)AS$3SHIFTCRSRL
FTT":N:SHIFTEW:?:N:SH
IFTT":W:SHIFTA:197,1:
R:SHIFTO:WTR:CMDRE
```



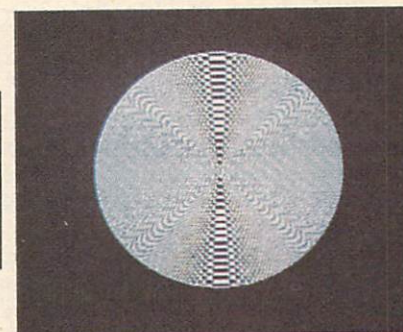
A Cosmic Egg

[BASICA on the IBM PC,
Cartridge BASIC on the IBM PCjr]
Dear Sirs:

This one-line program draws a circular design in screen 2 on the IBM PC, then BSAVES the image to a file in drive B named "DESIGN.PIC." It continues to clear the screen, BLOAD the file, and wait for a key to be pressed. Once a key is pressed, it goes back to screen 0 (text mode), clears the screen, and sets the width to 80.

Frank Swenton
Columbus, OH 43220

```
1:KEY=OFFTO:SCREEN=2:CLS:
FOR N=1 TO 200:C=-1*(IN
T(N/2))=(N/2):CIRCLE(31
9,99),N,C:NEXT:DEF SEG=
&HB800:B:BSAVE B:DESIGN.
PIC,0,&H4000:CLS:BLOAD
B:DESIGN.PIC,0:WHILE
INKEYS=:WEND:SCREEN
0:WIDTH 80:COLOR 7,0,0:
CLS
```



[NOTE: Because of built-in line length limitations in TI BASIC, we are now accepting "Ten-Liners" as entries for this column. —Ed.]

A Math Game Brain Teaser

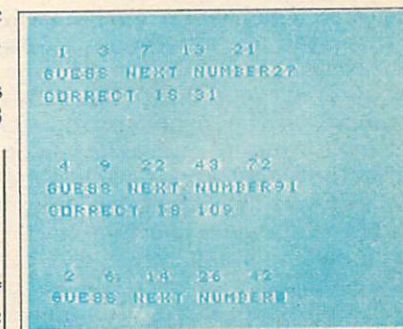
[TI Extended BASIC on the TI-99/4A]
Dear Sir:

This program plays a little mathematical game. It displays a sequence of five numbers, which are generated by an equation using random number coefficients. You are supposed to calculate what the next number in this series should be, and type it in. The computer

will tell you if you're right—and if you're not, it will display the correct answer, and another series will be displayed.

Edward Andrelos
Lefrak City, NY 11368

```
1:RANDOMIZE A=INT(RND*7)
::B=INT(RND*9+1)::FOR
X=1 TO 5:Y=A*X^2-B*
X+B::PRINT Y::NEXT P
RINT:NUMBER:INPUT GUESS
NEXT THEN PRINT "RIGHT"
Y:GOTO 1 ELSE PRINT "C
ORRECT IS":Y:GOTO 1
```



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!



MAC-ROs

Picture Maker

by William K. Balthrop
HCM Staff

With this easy-to-use programming utility, you can create your own character set or custom graphic shapes for use in other programs.

Creating graphic shapes for animation is an essential part of the development of many computer programs. Although the Macintosh is an extremely powerful graphics computer, and Microsoft BASIC is a powerful language, neither has a built-in utility for creating these shapes.

Our program, *Picture Maker*, will allow you to create and save to disk your own custom graphic shapes for use in other BASIC programs.

Making Pictures

When you RUN this program, you will be asked to enter the size of the shape you wish to construct. The maximum size available is a 40-pixel width by 30-pixel height.

After you enter the size, the screen will clear. On the left side of the screen will be a box that will contain a blown-up picture of the shape you will create. To the right of this large box will be a smaller box, where the picture on which you're working will be displayed at its normal size. As you turn pixels on and off in the large box, the corresponding pixels in the small box will be turned on and off.

Using the mouse, move the cursor inside the large box and click the mouse. A small black box will appear where the cursor is pointing. In the small box, a single pixel will be turned on. You may also hold down the button while sliding the mouse to draw.

Buttons

In the upper right-hand side of the screen are five option "buttons." Moving the cursor to one of these

buttons and clicking the mouse will activate the corresponding option:

LOAD
SAVE
CLEAR
PRINT
QUIT

LOAD

When you select this option, the familiar dialogue box for opening a new file will be displayed in the center of the screen. From here you can double-click a file displayed in the leftmost window, cancel the operation by clicking the cancel button, eject the disk in the current drive, or change drives if you have a second drive attached to your system. When you load a file, not only will the picture itself load, but also the size originally specified for that picture—replacing the dimensions of the picture previously on the screen.

SAVE

When you select this option, the familiar Save File dialog box will be displayed in the center of the screen. You can enter the file name of your file in this window. After entering the name, press RETURN or click the Save button. If you want to exit this operation, then click the Cancel button.

CLEAR

Selecting this option will refresh the screen as though you had reRUN the program. If you are working on a shape at the time you select this, it will be erased from the screen and memory.

PRINT

If you have a graphics printer, such as the Apple ImageWriter, you can get a hardcopy of the screen by selecting this option.

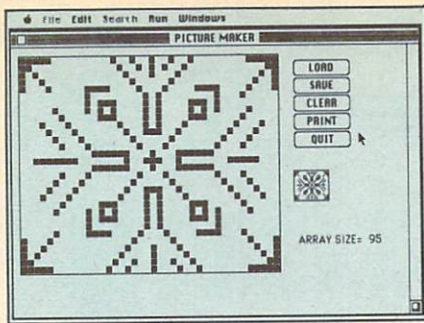
EXIT

Pressing this button will return you back to the BASIC interpreter. If you are unfamiliar with how to get back to the system's desktop, then move the cursor over to the pull-down window labeled File. Holding down the mouse button, slide the cursor down the list of options and select Quit. This will exit you from BASIC and return you to the system's desktop.

The Program

The first statement in the program is CLEAR. This reduces the amount of memory available to BASIC's data segment and the stack.

You may be asking: Why on Earth would you want to reduce these areas? Normally you would want to increase them. The reason is because memory not used by BASIC's data segment and the stack is placed into the heap; this is where Microsoft's BASIC interpreter keeps its transient code segments. Transient code segments are little parts of the interpreter that won't all fit into memory at the same time. If a code segment is needed and it is not yet in memory, it is loaded from the disk. This is why you will notice massive delays in the execution of your BASIC programs, and why the system is always accessing the disk, even when you didn't tell it to.



By reducing the BASIC data segment and stack, we make the heap larger so that more of Microsoft's interpreter can reside in memory. This greatly increases execution time. If you have the new 512 Macintosh, you need not concern yourself with this unless you have a very large program, or you are using one of the other features of Microsoft BASIC which fill up the heap—such as the PICTURE and SOUND commands.

Using Your Shapes:

To use the shapes that have been created with this program, insert the following two routines into your own applications. Use the GETSHAPE routine to load the shape into the SHAPE() array. If you use this routine for more than one shape, then you will need to add another dimension to the SHAPE array. The first shape loaded should be the largest, so that the array is dimensioned with enough elements to load the other shapes. You will need to place a test in the routine to keep the SHAPE() array from trying to dimension more than once. If the shapes are of vastly different sizes, you may want to use a different array for each shape to conserve memory. If you want to replace the shape in the SHAPE array by calling this routine again, then you must ERASE the SHAPE array before reDIMensioning it again. Use the PUTSHAPE routine to place the shape on the screen.

```
GETSHAPE:
OPEN F$ FOR INPUT AS 1
INPUT #1,W,H
E=(4+((H+1)*2*INT(W+16)/16))*5
DIM SHAPE%(E)
FOR EL=1 TO E
INPUT #1,SHAPE%(EL)
NEXT:CLOSE #1:RETURN
```

```
PUTSHAPE:
PUT (X,Y),SHAPE%:RETURN
```

HCM

HCM Glossary terms: dialogue box, data segment, stack, heap, transient code segment.

```
REM *****
REM * PICTURE MAKER *
REM *****
REM COPYRIGHT 1985
REM EMERALD VALLEY PUBLISHING CO.
REM BY WILLIAM K. BALTHROP
REM HOME COMPUTER MAGAZINE
REM VERSION 5.4.1
REM MACINTOSH with
REM MICROSOFT BASIC VERSION 2.0
CLEAR ,15000,3000
DEFINT A-Z:DIM FIXSCREEN(3870)
ON DIALOG GOSUB TESTBUTTON:DIALOG ON
CLS
WHILE WIDE<1 OR WIDE>40:LOCATE 1,1
INPUT "WIDTH OF PICTURE (1-40):";W$
WIDE=VAL(W$)
WEND
WHILE HIGH<1 OR HIGH>30:LOCATE 2,1
INPUT "HEIGHT OF THE PICTURE (1-30):";H$
HIGH=VAL(H$)
WEND
ELEM=(4+((HIGH+1)*2*INT((WIDE+16)/16)))*5
DIM PIC(WIDE,HIGH),SHAPE(ELEM):GOSUB DRAW
GETMOUSE:
BS=MOUSE(0):IF BS=0 THEN PPX=0:PPY=0:GOTO GETMOUSE
CPX=INT((MOUSE(1)-3)/8):CPY=INT((MOUSE(2)-3)/8)
IF (PPX=CPX AND PPY=CPY) THEN GOTO GETMOUSE
IF CPY<1 OR CPY>HIGH OR CPX<1 OR CPX>WIDE THEN GOTO GETMOUSE
CS=PIC(CPX,CPY):PPX=CPX:PPY=CPY
IF CS=0 THEN GOSUB TURNON ELSE GOSUB TURNOFF
GOTO GETMOUSE
TURNON:
PIC(CPX,CPY)=1:LINE (CPX*8+3,CPY*8+3)-STEP(6,6),1,BF
PSET (350+CPX,140+CPY),1:RETURN
TURNOFF:
PIC(CPX,CPY)=0:LINE (CPX*8+3,CPY*8+3)-STEP(6,6),0,BF
PSET (350+CPX,140+CPY),0:RETURN
TESTBUTTON:
DIALOG STOP:IF DIALOG(0)<>1 THEN RETURN
ON DIALOG(1) GOSUB LOADPIC,SAVEPIC,CLEARPIC,PRINTPIC,EXITPIC
IF DONE THEN RETURN FINAL
DIALOG ON:RETURN
SAVEPIC:
GET (50,14)-(420,174),FIXSCREEN
F$=FILES$(0,"SAVE PICTURE TO:")
IF LEN(F$)=0 THEN PUT (50,14),FIXSCREEN,PSET:RETURN
OPEN F$ FOR OUTPUT AS 1:WRITE #1,WIDE,HIGH
FOR EL=1 TO ELEM:WRITE #1,SHAPE(EL):NEXT
FOR CPX=1 TO WIDE:FOR CPY=1 TO HIGH
WRITE #1,PIC(CPX,CPY):NEXT:NEXT
CLOSE #1:PUT (50,14),FIXSCREEN,PSET:RETURN
LOADPIC:
GET (50,14)-(420,174),FIXSCREEN
F$=FILES$(1,"TEXT")
IF LEN(F$)=0 THEN PUT (50,14),FIXSCREEN,PSET:RETURN
OPEN F$ FOR INPUT AS 1
INPUT #1,WIDE,HIGH:ERASE PIC,SHAPE
ELEM=(4+((HIGH+1)*2*INT((WIDE+16)/16)))*5
DIM PIC(WIDE,HIGH),SHAPE(ELEM)
FOR EL=1 TO ELEM:INPUT #1,SHAPE(EL):NEXT
FOR CPX=1 TO WIDE:FOR CPY=1 TO HIGH
INPUT #1,PIC(CPX,CPY):NEXT:NEXT
CLOSE #1:PUT (50,14),FIXSCREEN,PSET
GOSUB DRAW:GOSUB REDRAW:RETURN
CLEARPIC:
CLS:FOR Z=1 TO ELEM:SHAPE(Z)=0:NEXT
FOR Z1=1 TO WIDE:FOR Z2=1 TO HIGH
PIC(Z1,Z2)=0:NEXT:NEXT:GOSUB DRAW
RETURN
PRINTPIC:LCOPY:RETURN
EXITPIC:DONE=-1:RETURN
DRAW:CLS
BUTTON 1,1,"LOAD",(350,16)-(420,34)
BUTTON 2,1,"SAVE",(350,36)-(420,54)
BUTTON 3,1,"CLEAR",(350,56)-(420,74)
BUTTON 4,1,"PRINT",(350,76)-(420,94)
BUTTON 5,1,"QUIT",(350,96)-(420,114)
SHOWPEN:LINE (9,9)-STEP(WIDE*8+2,HIGH*8+2),,B
LINE (349,139)-STEP(WIDE+3,HIGH+3),,B
LOCATE 14,45:PRINT "ARRAY SIZE=";ELEM:RETURN
REDRAW:
FOR CPX=1 TO WIDE:FOR CPY=1 TO HIGH
IF PIC(CPX,CPY)=0 THEN GOSUB TURNOFF ELSE GOSUB TURNON
NEXT:NEXT:RETURN
FINAL:END
```




IBMPRESSIONS

WINDOWING

by Scott Williams

Why pay hundreds of dollars for sophisticated software when you can write similar programs yourself? Create windows and learn a few handy graphics tricks with this entertaining routine.

Suddenly a program seems to destroy a section of the screen by writing over the top of existing text or graphics, then suddenly disappears—returning the screen to its original state. How was the computer able to remember what was on the screen before it was disrupted?

The program listed on the opposite page demonstrates two cases when a graphics screen may become disrupted and then restored back to its original condition. (We call this technique "windowing.") In addition, we have included a graphics routine to provide something to experiment on. This graphics routine may be of interest in itself because it draws a screen full of three-dimensional Easter eggs. We will discuss all three routines in the order in which they appear as you run the program.

Eggs

At the heart of this routine are the **SIN** and **COS** functions. Together, these two functions are the basis by which we are able to draw circles on the computer. In this routine we will create spheres—or eggs, to be more exact. By drawing circles with the **CIRCLE** command at the correct locations and with the right diameter, we can create a three-dimensional drawing of an egg. To help make the eggs more visible, we will draw each egg using two different colors.

The **COS** function determines the location of the center of each circle. The offset created with this function is multiplied by an X and Y coordinate factor—**XO** and **YO**—which determine the direction in which the center of the circles will move. The **SIN** function then determines the radius of the circles.

Windows

The **GET** and **PUT** commands are perfect for temporarily saving the contents of a graphics screen so that a window can be overlaid to display a message, or other graphics. When the window is no longer needed, the saved portion of the screen can be replaced, returning the screen to its original state. The **A1()** array is used to hold the screen graphics.

The size of the array depends on how much of the screen you need to save, and the number of bits used to define the color of each pixel on the screen. The chart below can be used to determine the number of bits used for each pixel.

No. of Colors	Bits	Screen Modes
2 (B/W)	1	2
4	2	1,4*,6*
16	4	3,5*
16(text)	N/A	0 (no graphics)

* IBM PCjr or Tandy 1000 only.

After establishing the number of bits used in your display, use the following formula to determine the size of the array:

$$4 + \text{INT}((\text{hpxels} * \text{bits} + 7) / 8) * \text{vpxels}$$

"hpxels" is the width of the area in pixels that you wish to GET, and "vpxels" is the height of the area in pixels. "bits" is the number of bits per pixel, as determined by the chart above.

This program saves an area of the screen in the **A1()** array in line 380. It then clears that area of the screen using the **LINE** command with the "Box-Fill" option. A text message is then printed inside the window. After pressing a key, the previously saved graphics return to the screen via the **PUT** command. It's important to note here that the program uses the **PSET** option of the **PUT** command. This causes the image stored in the array to be placed back on the screen exactly as it was lifted. The default mode for the **PUT** command is the **XOR** option, explained later.

Cut And Paste

Have you used any of the sophisticated drawing programs available that allow you to work with graphics as a word processor works with text? How would you like to be able to design your own graphics drawing program? This next routine may give you some insight into how such a program could allow you to do just that.

Two arrays can be used to store two different areas of the screen. These areas can overlap or be in totally separate locations on the screen. Once the two areas have been saved (arrays A1() and A2()), you can actually slide one area across the screen without upsetting the rest of the screen's graphics. To do this, you need to use the default option of the PUT command (XOR). This option inverses the image of any pixel on the screen that comes in contact with a pixel in the image in the array. By PUTting the array at the same location twice, we effectively erase the array's image from the screen, inverting the screen pixels again—this time returning them to their original state.

Once the shape has been moved to its destination (the area occupied by the shape contained in the second array A2()), it can be permanently placed down on the screen, erasing any graphics that may have been there already. The PSET option of the PUT command accomplishes this.

Oops! Suppose you didn't really want to move to that area of the screen. That's perfectly OK—nothing is lost. You saved the graphics that previously occupied that area of the screen. It's a simple matter to replace it.

Try experimenting with these techniques. Come up with your own solutions to problems using the methods described here. You might try copying a section of graphics rather than doing a block move—or try using some of the other options available with the PUT command to see their effects.

PCjr and Tandy 1000

If you have an IBM PCjr or Tandy 1000, you may wish to make the following modifications to the program listed here to optimally use your system. The program as listed will display only 4 colors (PC screen mode 1.) With the following modifications, you can get 16 colors, as shown in the two accompanying photos.

Change these two lines to read as follows:

```
240 CLS: CLEAR ..., 32768: SCREEN 5: KEY OFF: RANDOMIZE TIMER
340 C(0) = INT(RND*16): C(1) = INT(RND*16): IF C(0) = C(1) THEN 340
```

HCM Glossary terms: pixel, radian, window.

HCM



Photo 1. This is the Eggs program after the modification for the PCjr has been added. This version has 16 colors to work with; the PC version is capable of using 4 colors.



Photo 2. The graphics displaced by the window have been saved in an integer array. When the window is removed, the graphics can be restored to their original condition.

How would you like to be able to design your own graphics drawing program?

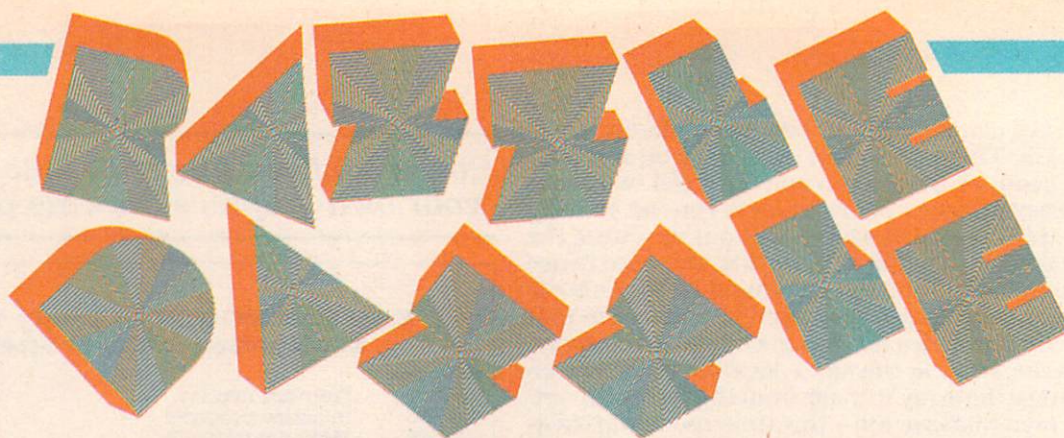
Eggs (IBM PC & PCjr; Tandy 1000) Explanation of the Program

Line Nos.	Explanation
100-230	Program header.
240-250	Initialize program.
260-290	Main control loop.
300-350	Draw ten eggs on the screen.
360-400	Create a window on the screen and place text there.
410-460	Grab a section of the graphics and drag it across the screen.

Variables

A1(), A2()	Arrays containing graphics image.
AS	Input variable.
B, TD, Z	Loop counters.
C()	Contains two colors for egg.
COL	Index to C() array for egg color.
SIZE	Size of the egg.
X, Y	Screen coordinate for start of egg.
XO, YO	Direction of egg's axis.

```
100 *****
110 * EGGS *
120 *****
130 COPYRIGHT 1985
140 EMERALD VALLEY PUBLISHING CO.
150 BY WILLIAM K. BALTHROP
160 HOME COMPUTER MAGAZINE
170 VERSION 5.4.1
180 DOS 2.1 AND EITHER
190 IBM PCjr WITH CARTRIDGE BASIC or
200 IBM PC WITH BASICA and
210 COLOR/GRAPHICS ADAPTER and
220 COLOR MONITOR
230
240 CLS: SCREEN 1: KEY OFF: RANDOMIZE TIME
250 R
260 DEFINT A: DIM A1(737), A2(737)
270
280 CONTROL LOOP
290
300 GOSUB 330: FOR Z=1 TO 2: ON Z GOSUB 3
310 90, 450: NEXT: GOTO 290
320
330 DRAW TEN EGGS
340
350 FOR BB=1 TO 10
360 C(0)=INT(RND*4): C(1)=INT(RND*4): IF
370 C(0)=C(1) THEN 340
380
390 X=INT(RND*319): Y=INT(RND*199): SIZE=
400 INT(RND*30)+10: XO=INT(RND*3)-1: YO=I
410 NT(RND*3)-1: FOR B=0 TO 3.14 STEP .0
420 2: CIRCLE (X+COS(B)*SIZE*XO+.4, Y+COS
430 (B)*SIZE*YO+.4), SIN(B)*SIZE, C(COL):
440 COL=ABS(COL-1): NEXT: NEXT: RETURN
450
460 USE GET & PUT FOR WINDOWS
470
480 GET (100, 50)-(174, 124), A1: LINE (100
490 , 50)-(174, 124), 0, BF: LOCATE 8, 14: PRI
500 NT "PRESS": LOCATE 9, 14: PRINT "A KEY
510 ": LOCATE 10, 14: PRINT "TO GET": LOCAT
520 E 11, 14: PRINT "IT": LOCATE 12, 14: PRI
530 NT "BACK"
540 AS=INKEY$: IF AS="" THEN 400 ELSE PU
550 T (100, 50), A1, PSET: RETURN
560
570 USE GET & PUT TO DO CUT
580 AND PASTE WITH SCREEN RESTORE
590
600 GET (150, 25)-(224, 99), A1: GET (110, 6
610 5)-(184, 139), A2: PUT (150, 25), A1: FOR
620 X=0 TO 39 STEP 2: PUT (150-X, 25+X),
630 A1: FOR TD=1 TO 50: NEXT: PUT (150-X, 2
640 5+X), A1: NEXT: PUT (110, 65), A1, PSET
650 FOR TD=1 TO 5000: NEXT: PUT (110, 65),
660 A1: PUT (110, 65), A2, PSET: FOR X=0 TO
670 39 STEP 2: PUT (110+X, 65-X), A1: FOR T
680 D=1 TO 50: NEXT: PUT (110+X, 65-X), A1:
690 NEXT: PUT (150, 25), A1, PSET: RETURN
```

SOUND-ON-SOUND

by William K. Balthrop
HCM Staff

Frustrated musicians rejoice!

Create musical compositions with the press of a button. Explore the principles of sound-on-sound recording with this simple 3-track recorder program.

Music is both a powerful and an everyday part of our lives. It's hard to imagine what the world would be like without it. Perhaps you have always had a secret fantasy that you could compose great musical works, letting all of that bottled-up creativity flow out into a musical instrument.

Someone once said that the best things come in small packages, and so it may be true with the short program listed here. With *Composer* you can compose and record music, play it back, and save it to disk or tape.

Three voices are available on the TI-99/4A. This means that the computer can produce up to three different tones at the same time. This ability allows you to produce some amazing musical results.

Getting Started

When you run this program, you will be presented with a menu containing five options:

- 1) RECORD
- 2) PLAY BACK
- 3) LOAD FILE
- 4) SAVE FILE
- 5) EXIT

Record

This option will allow you to record up to 200 notes on each of the three voices, one voice at a time. After selecting this option you will be asked to enter the voice number with which you wish to work. Enter a number from 1 to 3.

You will then be given two choices for the method of recording: **PLAY**, and **STEP**. If you select **PLAY** then you will be able to play along with the other two voices (providing there is something recorded on them) as you record this voice. If at any time you wish to terminate music entry to this voice, press the period key; or, if you have a 99/4, press the [ENTER] key.

In **STEP** mode you can get picky about which notes go where. In this mode each beat is single-stepped—when you press a new note, that note is played along with the other two voices (if they contain notes). The program will wait for you to press another note before continuing.

Two octaves of notes are available, starting with middle C. Figure 1 gives you the details for each key, the note it represents, and the frequency produced.

FIGURE 1

Key	Note	Frequency
1	C	262
2	D	294
3	E	330
4	F	349
5	G	392
6	A	440
7	B	494
Q	C	523
W	D	587
E	E	659
R	F	698
T	G	784
Y	A	880
U	B	988

. (period) Press to exit before entering 200 notes.

Note: All keys which do not produce a note will insert a blank—a pause in the music.

Play Back

When you select this option, all that you need to do is sit back and listen. If you have a tune in memory, it will now play back for you all three voices. A counter at the bottom of the screen tells you the number of notes that have played. The sequence will stop when the counter reaches 200.

Load And Save Files

These two options work identically, except that option 3 will **LOAD** a previously saved music file into memory for more work, or simply to be played back. Option 4 will **SAVE** your work so that you may continue later, or so that your friends can enjoy your compositions with you at a later time. If you are using a tape drive to save the music, enter CS1 at the prompt for the file name. Otherwise, if you are saving or loading to a disk, enter the device name and file name; e.g. DSK1.MUSIC.

Exit

When you attempt to exit, or if you press option 5 from the main menu by accident, you will be asked to indicate that you're sure you want to exit. If exiting was an acci-



Create A Pie Chart

by Roger Wood
HCM Staff

Pie charts are a helpful visual aid for presenting data. This program will show you how to combine a pie chart with character graphics on the hi-res screen.

Have you ever wished that you could put a pie chart into your programs, but were just too confused about SIN, COS, radian measure, and aspect ratios to do it? Or, maybe drawing the pie chart didn't bother you, but the difficulty of placing characters on the high-resolution (hi-res) screen made labeling your chart more than you wanted to tackle. Well, by using last issue's "Apple Seedlings" programs entitled *Character Graphics*, we have combined these elements in a short program called *Pie Chart* to help demonstrate how easy it can be.

Because this program uses the Character Graphics System, it relocates itself above the hi-res screen and loads the machine-language routine where BASIC text is usually loaded. This is accomplished in lines 190-210. It is essential that you **SAVE** this program *before* you **RUN** it. If you fail to do this, you will at least get a **FILE** or **PATH NOT FOUND** error when you try to **RUN** it, as it must reload itself from disk; at most, you could lose the program you've typed-in.

Using The Program

When you **RUN** the *Pie Chart* program, input 4 values (at least one non-zero value is required). These values are identified simply as A, B, C, and D. You will then need to indicate the radius of the chart in pixels. The prompt asks for a value between 30 and 85 pixels (picture elements). If you try to draw a pie chart with a radius smaller than 30, the screen resolution will not be fine enough to discern the different categories. The picture shown at right is an example of a radius of 80. If you try to draw a radius of more than 85, you will get an **ILLEGAL QUANTITY** error. This is because, with this large a radius, the program would attempt to plot a point that doesn't exist on the hi-res screen. To keep the program as small as possible, we do not include extensive error-checking, so keep your responses within the ranges indicated.

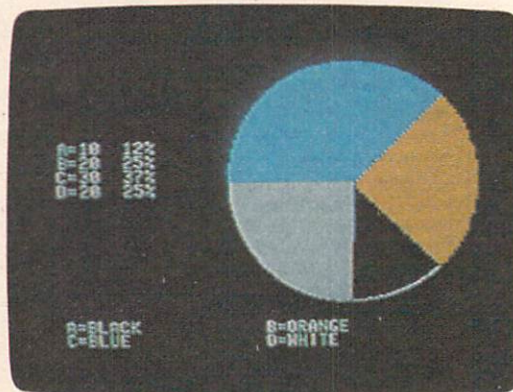
After you have entered these values, the program asks you to indicate the color set that you wish to use. Apple graphics allows for a total of 6 colors: white, black, green, purple, blue, and orange. These colors are divided into 2 color sets (both sets include black and white). Color set 0 adds green and purple, and color set 1 gives you blue and orange. Due to the columnar nature of Apple graphics, you cannot readily display colors from dif-

ferent color sets adjacent to each other. We included only 4 inputs so that only 4 colors will be used at any time, thus avoiding this potential difficulty.

Character Graphics To The Rescue

Before the program draws the pie chart, it calculates the percentages of each of the 4 inputs, and displays the results. This is where the Character Graphics System comes in. Normally the split hi-res/text screen display only allows 4 rows of text at the bottom of the screen, but the Character Graphics System lets you **PRINT** characters that you've defined using the **CALL CHAR** command directly on the hi-res screen. We have defined several characters using the *Character Editor* and included their codes in lines 390-410. [For details on how to use the Character Graphics System, see "Apple Seedlings" in HCM Vol. 5, No. 3—Ed.]

Line 270 then **CALLS** the **PR** routine, which allows these graphic characters to print to the hi-res screen. Notice that **PR** is actually a variable initialized in line 430 to the address of the Character Graphic System's machine-language routine, which is used to accomplish this special **PRINT** function. Next, the values you input and their respective percentages are **PRINTED** right next to where the pie chart will be drawn on the hi-res screen.



Here is a sample screen of the *Pie Chart* program. Color set 1 was selected, with a radius of 80 pixels. Because percentage values are truncated by the program, the total percentage in this case adds up to only 99 percent.

Finally, a list of which colors represent which letters is printed in the text window at the bottom of the screen.

Lines 300-370 actually draw the pie chart. A FOR-NEXT loop steps from .01 to 6.27 in increments of .01 to generate the values needed in the HPLLOT statements within the loop. Those of you who are familiar with radian measure in trigonometry will recognize that this method will encompass the full circumference of any circle when used in conjunction with the SIN and COS functions available in BASIC. It is exactly how this program draws the circle.

Three adjustments are necessary before the values generated by these functions are plugged into the HPLLOT statements in lines 350 and 360: First, an offset is added to establish the placement of the center of the circle. The vertical and horizontal offsets (VP and HP) are hard-coded in line 260, placing the circle midway between the top and bottom, and on the right third of the screen. You can move the circle around to a limited degree by trying different values here.

Second, the radius is multiplied by the value of the function to correctly position the edge of the circle.

The third adjustment is necessary when drawing the vertical position, because of the aspect ratio of the Apple hi-res screen. The aspect ratio is due to the difference in vertical and horizontal resolution. If this factor were not included, our circle would become an ellipse, with the vertical axis greater than the horizontal. We found a factor of 6 to 7 to be most satisfactory.

Color By Percentage

The only other consideration in drawing a pie chart lies in determining which color goes where. The background of the Apple hi-res screen is normally black—therefore, the black section of the chart is outlined with white. The rest of the colors are created by drawing successive rays from the center to the edge of the circle.

We've used the fact that colors can be chosen by number to select which color is active. Lines 300-330 determine which of the 4 different inputs is being represented at any given moment. We set the current color (HC) by comparing the current value of the control variable (I) to the product of Terminating Value (TV) and the percentage already represented, plus what is currently being drawn. For example, as long as the value of the control variable of the FOR-NEXT loop is less than the percentage amount that A should take up, the HC variable is set to 0, and line 350 is executed. This line simply draws the circumference of the circle in white. When I reaches a value greater than A's share, the HC variable changes to a 1. This variable is then used in conjunction with the Color Set (CS variable) selected to determine the color of the rays drawn in line 360.

HCM

HCM Glossary terms: aspect ratio, pixel, machine language, hard-coded.

Pie Chart (Apple II Family) Explanation of the Program

Line Nos.	Explanation of the Program
100-180	Program header.
190-210	Protect hi-res screen.
220	Initialize character graphics.
230	Truncating function.
240-260	Input variables, and initialize constants.
270-290	Display text.
300-380	Draw pie chart.
390-410	Define characters.
420-520	Initialize Character Graphics System.

```

100 REM *****
110 REM * PIE CHART *
120 REM *****
130 REM COPYRIGHT 1985
140 REM EMERALD VALLEY PUBLISHING CO
150 REM BY ROGER WOOD
160 REM HOME COMPUTER MAGAZINE
170 REM VERSION 5.4.1
180 REM APPLE II FAMILY APPLESOFT
190 IF PEEK (104) = 64 THEN 220
200 POKE 104,64: POKE 103,1: POKE 16384
0
210 PRINT CHR$ (4): RUN "PIE.CHART"
220 GOSUB 420: GOSUB 390
230 DEF FN RD(X) = INT (X * 100)
240 TEXT : HOME : PRINT "VALUES " F
OR A,B,C,D : INPUT "A=" : INPUT "B
=" : B : INPUT "C=" : INPUT "D=" : D
250 T = A + B + C + D : AP = A / T : BP = B
/ T : CP = C / T : DP = D / T
260 INPUT "RADIUS (30-85) : CS : HP = 190 : V
" COLOR SET (0 OR 1) : CS : HP = 190 : V
P = 80 : TV = 6.27 : SV = .01
270 HGR : CALL PR : VTAB 8 : PRINT AS : " =
" : B : FN RD (AP) : " : PRINT BS : " =
" : C : FN RD (BP) : " : PRINT CS : " =
" : D : FN RD (CP) : " : PRINT DS : " =
" : D : FN RD (DP) : " : CALL OFF
280 IF CS = 0 THEN VTAB 21 : PRINT "A=B
LACK" TAB (20) "B=GREEN" : PRINT "C=P
URPLE" TAB (20) "D=WHITE" : GOTO 300
290 VTAB 21 : PRINT "A=BLACK" TAB (20) "B
=ORANGE" : PRINT "C=BLUE" TAB (20) "D
=WHITE" :
300 FOR I = SV TO TV STEP SV
310 IF I < (AP * TV) THEN HC = 0 : GOTO
350
320 IF I < ((AP + BP) * TV) THEN HC = 1
: GOTO 350
330 IF I < ((AP + BP + CP) * TV) THEN H
C = 2 : GOTO 350
340 HC = 3
350 HCOLOR = HC + (CS * 4) : IF HC = 0 OR
HC = 4 THEN HCOLOR = 3 + (CS * 4) :
HPLLOT SIN (I) * RAD + HP, COS (I)
* (RAD * 6 / 7) + VP : GOTO 370
360 HPLLOT HP, VP TO SIN (I) * RAD + HP,
COS (I) * (RAD * 6 / 7) + VP
370 NEXT I
380 END
390 CALL CHAR, 1, "0814223E22222200" : CAL
L CHAR, 2, "1E22221E22221E00" : CALL C
HAR, 3, "0E11010101110E00" : CALL CHAR
4, "0F111111111110F000" : CALL CHAR, 5,
"0626100804323000" : CALL CHAR, 14, "0
0000000000000000"
400 CALL CHAR, 16, "1C22322A26221C" : CALL
CHAR, 17, "080C080808081C" : CALL CHA
R, 18, "1C22201804023E" : CALL CHAR, 19
, "3E20101820221C" : CALL CHAR, 20, "10
1814123E1010" : CALL CHAR, 21, "3E021E
2020221C"
410 CALL CHAR, 22, "3804021E22221C" : CALL
CHAR, 23, "3E201008040404" : CALL CHA
R, 24, "1C22221C22221C" : CALL CHAR, 25
, "01C22223C20100E" : CALL CHAR, 29, "00
001F001F00000000" : CALL CHAR, 0, "0" : R
ETURN
420 IF PEEK (2048) < 76 THEN FOR K
= 2048 TO 2336 : READ P : POKE K, P :
NEXT : READ P : IF P < 9999 THEN
PRINT "DATA ERROR" : END
430 PRNT = 2048 : OFF = PRNT + 3 : CHAR = 0
FF + 3 : HCHAR = CHAR + 3
440 AS = "I" : BS = CHR$ (34) : CS = " " : D
$ = " " : RETURN
450 DATA 76,151,8,76,157,8,76,194,8,32,7
76,231,134,37,32,76,231,134,36,32,7
6,231,134,8,32,76,231,134,6,32,76,2
31,138,41,3,170,189,187,8,141
460 DATA 105,8,165,37,32,34,252,32,60,
8,169,160,32,240,253,198,6,208,244,
96,169,7,133,7,160,0,36,50,48,2,160
,127,132,9,165,8,10,10,10,170
470 DATA 164,36,24,165,40,133,38,165,41
,105,24,133,39,24,165,39,105,4,133,
39,189,25,9,69,9,17,38,145,38,232,1
98,7,16,235,96,141,191,8,142,192
480 DATA 8,140,193,8,201,160,144,11,201
,191,176,7,41,31,133,8,32,60,8,173,
,191,8,174,192,8,172,193,8,76,240,25
3,160,115,169,8,208,4,160,240,169
490 DATA 253,174,9,191,224,76,208,12,17
4,15,191,208,7,140,48,190,141,49,19
0,96,132,54,133,55,76,234,3,145,49,
81,17,0,0,0,32,76,231,134,8,32
500 DATA 190,222,165,8,10,10,10,170,202
,160,255,132,6,165,6,73,255,133,6,2
08,6,232,169,0,157,25,9,32,177,0,20
1,34,240,31,73,48,201,10,144,8
510 DATA 105,152,201,16,144,2,169,0,36,
6,208,4,10,10,10,29,25,9,157,25,
9,76,213,8,232,138,41,7,240,7,169,0
,157,25,9,240,243,76,177
520 DATA 0,0,0,0,0,0,0,0,9999

```




Commodore Hornblower

Sound Filtering

by Randy Thompson
HCM Staff

*Have you changed your filters lately?
Try changing the ones on this hot little
sound engine—Commodore's amazing SID chip.*

Besides frequency, waveform, and envelope, the Commodore SID chip has some other tricks up its sleeve: filters. And, unlike the filters you may find in your car, these don't get dirty. The filters provided by the C-64 are, not surprisingly, *sound* filters that enable you to screen out certain frequency ranges. You can filter any one of SID's three voices, together or separately. Combined with the other features found on the SID chip, sound filters can be used to create some impressive effects.

A Simple Synthesizer

If you've been following this column, which has been illustrating how to build a synthesizer step-by-step, then the *Filters* program presented here should look fairly familiar. To play a note, just press its corresponding key—e.g., press a C to play the note C. Shifting the key will sharpen the note—bringing it up a half-step. You can also play an octave higher or lower with the + and - keys. To quit, press the [-] key. Your options are displayed on the screen at all times (see the photo). There's nothing fancy about this simple synthesizer, but it's a good way to experiment with the sound chip.

Filtered Frequencies

In addition to the note-playing capabilities of this program, you are also given the ability to *filter* each sound. The filter settings are controlled by the F1, F3, F5, and F7 keys. F1 actually turns on the filters. If this filter enable is not turned on, all other filter controls are ineffective. The last three keys—F3, F5, and F7—control the low-pass, band-pass, and high-pass filters, respectively. As their names imply, these filters only allow certain frequencies to "pass" through them. The low-pass filter lets low frequencies through; band pass allows only midrange frequencies; and high pass lets the high frequencies pass. You can combine these filters in any combination to create a variety of effects. By turning

on the low- and high-pass filters for example, and leaving the band pass off, you get an effect that is referred to as a *notch* filter (see Figure 1). By blocking the middle frequencies from passing, a notch filter produces a somewhat hollow sound.

The Fine Tuning

It is a pretty general statement to say that the high-pass filter lets high frequencies through. What exactly constitutes a "high" frequency? This is where Cut-Off comes in. The Cut-Off frequency is the frequency at which the filter becomes active. With a value range of 0-2047, Cut-Off can be set to represent a frequency range of 30 to 11902 hertz. Anything above the Cut-Off frequency is considered high, and anything below it is low. The notes that are directly around the Cut-Off frequency are considered the midrange, or band-pass frequencies. Figure 1 is a visual representation of Cut-Off's effect on the three different filters. The last diagram displays a notch filter that is created by combin-

ing both the low and high filters. I'm sure you'll see the great effect that Cut-Off can have on the performance of each filter.

Resonance is like a volume control for Cut-Off. By peaking the volume of those frequencies closest to Cut-Off, Resonance gives the filters a sharper sound. The higher the Resonance value, the greater effect Cut-Off has on the filtered sounds. Resonance can range in value from 0 to 15—with 0 being the lowest Resonance.

To set Cut-Off with the *Filters* program, press the (CSRDOWN) key. Resonance is set with the (CSRRIGHT) key. Once pressed, the cursor will appear, and you can enter the value of your choice. When you are satisfied, press (RETURN), and the setting will be changed. A Cut-Off frequency reading in hertz is given anytime the Cut-Off value is changed. When changing any of these values, it is often a good idea to play a note with the Sustain on. This way, you hear the tonal changes immediately.

```
READY
PRESS LETTER TO PLAY THAT NOTE
PRESS SHIFT TO SHARP NOTE 1/2 STEP
PRESS + TO GO UP AN OCTAVE
PRESS - TO GO DOWN AN OCTAVE
SUSTAIN IS OFF. PRESS S TO CHANGE.
FILTER IS OFF. PRESS F1 TO CHANGE.
F3 - LOW PASS OFF
F5 - BAND PASS OFF
F7 - HIGH PASS OFF
CUT-OFF = 8071 - (CSRDOWN) TO CHANGE.
CUT-OFF FREQUENCY = 441 HZ
RESONANCE = 15 - (CSRRIGHT) TO CHANGE.
PRESS - TO QUIT
```

Here both the low- and high-pass filters are on. Notice that the Cut-Off frequency is set to 441 hertz—approximately an A note.

Filters presets 3 items: the wave-form, volume, and ADSR. For an explanation of wave-form and ADSR (Attack Decay Sustain Release), see last issue's "Commodore Hornblower." Customizing these settings is simply a matter of changing certain POKE statements in the program's initialization routine—lines 870-1000. We are concerned with line number 900. Before changing any values, refer to the Variable Explanations below.

The variables you should use for changing the envelope are AD and SR. By poking AD you can change the Attack and Decay. Poking SR sets the Sustain and Release. For a full explanation on how to manipulate these two memory addresses, see the *Commodore 64 Programmers' Reference Guide*, pages 196-199.

The volume can be changed by poking VL with a number between 0 and 15. Currently the program uses a volume setting of 15—the loudest.

Variable	Explanation
CR	Control Register (sets wave-form)
AD	Attack Decay.
SR	Sustain Release
VL	Volume (also sets filters)
CU	Cut-Off frequency.
RE	Resonance

As you can see—or should I say *hear*—filters really add a lot to the already-versatile SID chip. With a little experimentation and imagination, some great effects can be accomplished. Next month, we'll take a look at the interaction of two different waveforms through synchronization.

HCM

HCM Glossary terms: buffer, frequency, hertz, RAM, ROM, SID, WOM.

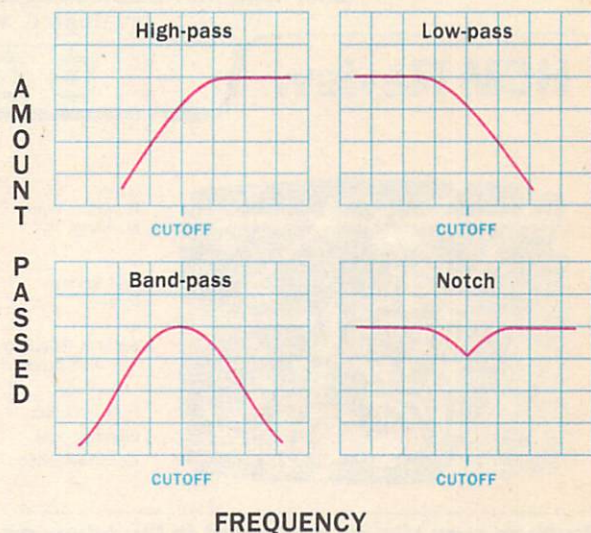
For your key-in listings, see HCM PROGRAM LISTINGS Contents.

Filters (C-64) Explanation of the Program

Line Nos.	
100-180	Program header.
190-220	Program initialization.
230-410	Main play routine.
420-530	Turn filters on and off.
540-670	Set Cut-Off and Resonance.
680-860	I/O subroutines.
870-1000	Initialization subroutine.
1010-1210	PEEKable POKEs subroutine.*

*See sidebar on PEEKable POKEs

Figure 1



PEEKable POKEs

You've probably heard the terms ROM and RAM in reference to computer memory, but have you ever heard of WOM? Well, if you've programmed the Commodore's SID chip, then you've used it. WOM is Write-Only Memory—in other words, it can be POKED, but not PEEKed. In the case of Commodore's sound chip, PEEKing any one of the SID chip's first 25 registers will return a value of 0, no matter what is actually stored there. When trying to create and edit different sounds, this quirk can become quite irritating.

In this month's "Commodore Hornblower" program, we found a way to solve the problem of write-only memory. In lines 1040-1210 of the *Filters* program, you'll see a series of DATA statements. These DATA statements hold the machine language for a routine called *PEEKable POKEs*. As the title suggests, *PEEKable POKEs* will allow you to PEEK the C-64's sound chip (memory locations 54272 to 54300). Now, to find out the volume setting or what waveform a certain voice is using, simply PEEK it.

In order for *PEEKable POKEs* to work, the normal PEEK and POKE routines have to be intercepted so that whenever the computer encounters one of these two statements, *PEEKable POKEs* is called instead. Once called, *PEEKable POKEs* will determine whether it is the SID chip

that is to be accessed or not. If not, then control is given back to the normal ROM routine. If it is, one of two things will happen, depending on whether the statement intercepted was a POKE or a PEEK.

In the case of a POKE, the number to be stored is actually placed in two different locations. Not only is it placed in the SID chip, but it is also placed within a storage buffer located at 49295-49323. By placing this number into the storage buffer, we can PEEK it out later! Once this has been done, we exit the routine.

The way the new PEEK routine works is very simple. Instead of PEEKing the SID chip, which is a useless process anyway, this routine gets its value from the storage buffer. Since the POKE command has been forced to store all of its SID chip POKEs here, we can be sure of a correct reading.

If you want to use *PEEKable POKEs* in your own programs, simply copy the subroutine located in lines 1010-1210 of the *Filters* program. Now, at the beginning of your program initialization, GOSUB to this routine, and *PEEKable POKEs* will be activated automatically. Always be sure to call this subroutine before any POKEs to the SID chip are made, or they will not be recognized when you try to PEEK the values back out.

HCM

HCM Review Criteria

Each month, *Home Computer Magazine* (HCM) reviews products designed for the Apple II Family, Commodore 64 and VIC-20, 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, C-64 & VIC-20, 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

Performance:
Engrossment:
Documentation:

* 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.

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?

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.

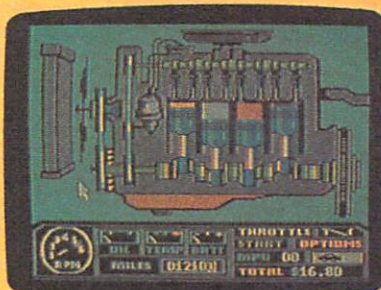
*Compu-prestidigitation

(kóm•pū•prēs•teh•dī•jeh•tā•shūn) —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.

INJURED ENGINE

A Review
by **Rhea J. Grundy**
HCM Staff

HCM Review



Name: Injured Engine
Program Type: Education Simulation
Machines: Apple II Family, Commodore 64
Distributor: Imagic, Inc.
2400 Bayshore Frontage Rd.
Mountain View, CA 94043
(415) 940-6030
Price: \$39.95
System Requirements:
Disk drive, joystick, or mouse (Apple only).
Performance: Poor Fair Good Excellent
Engrossment:
Ease of Use:
Apple
C-64
Documentation:
Cost/Benefit:

Your car engine is made of more than doohickies, gizmos, and thingamajigs. Learn the parts and their function before you need to repair them.

No matter how much you like to drive your car, let's face it: you're going to have to maintain it or suffer the stranded-on-the-side-of-the-road blues. Changing the oil once in a while and looking at the tires for wear may be all you do to participate in the maintenance of your vehicle, but that's a start. *Injured Engine*, from Imagic, may help you take one more step toward automotive-repair self-sufficiency.

The Engine, Explained

Injured Engine will demonstrate the processes that occur inside an internal combustion engine as it runs. You begin your engine anatomy lesson by selecting On the Road or In the Shop (Apple version) and Normal Simulation or Troubleshooting (Commodore version).

Selections from In the Shop and Troubleshooting offer five problem sets. The first set simulates a major repair problem that you must find as fast as possible. Each inspection and repair costs you in time and money—in this respect the program is a real-life, as well as real-time, simulation.

The Apple version allows you to select an engine part for inspection and repair from the main engine screen. For instance, mouse-clicking your pointer-cursor on the radiator produces a separate screen that shows the relationship of each part of the cooling system. An on-screen text window describes the function of the system—in this case, to remove the heat produced by the engine's combustion chamber. You can scroll down through the text to study the inter-relationships of the parts; this text gives you a succinct overview of an engine's function; therefore, there is little need for written documentation, which is limited at best. Or, you can choose the Magnifying Glass option to inspect the part's system to see whether it is in working order or needs replacement.

The Wrench option is selected to fix a worn or defective part. Choosing the Dollar Sign option informs the fledgling compu-mechanic how much it will cost to inspect and repair the injured engine—in other words, how much it's going to cost to get "On the Road Again."

The Troubleshooting section of the Commodore version is analogous to the Apple program's In the Shop problem-solving sets—but the similarity stops there. Part diagrams are not integrated with part names; instead of pointing your cursor at a part, you must choose from names on a list. It's distracting to make repair selections from a table instead of from a diagram.

Another major difference between the two versions is in the development of program graphics. On the Apple IIc with double hi-res graphics, you have color resolution with 16-color capability. To achieve the same ef-

fect on the IIe you need an extended 80-column card with connector J1 pins 50 and 55 jumpered together. The variations in color on the II+ or the IIe without this enhancement are less than outstanding. In contrast, the C-64 version uses multi-color mode; though colorful, the low resolution is a drawback.

Also, the Apple version lets you select the Keyboard, Joystick, or Mouse option to freely move around the engine screen and diagnose problems on the spot. In contrast, with the C-64 product, your only option for play is with a joystick—you sit, joystick in hand, waiting (and waiting for the slow disk access) to skitter through a parts list making selections.

Diagnosing engine problems is more meaningful in the On the Road (Normal Simulation) option. Selecting this option gives you three choices for playing. Begin with 1,000, 40,000, or 80,000 miles. Start your engine and let it run. You'll notice right away that as the miles rack-up, so does the cost of running your car. Oil, temperature, battery, throttle, MPG, a pollution indicator, RPM (the motion of the engine parts is not shown in direct proportion to the RPM), total miles, and cost gauges keep you informed of the engine's repair status. If the temperature starts to go up with the oil pressure, the MPG rapidly begins to drop, the RPM increases, and the pollution indicator begins to produce blue and black smoke—the piston rings are probably broken, the valves will need replacing, and the oil is five quarts low.

And The Bill Is . . .

The repair bill adds up fast. If you've neglected repairing your car completely, replacing *everything* in the engine will cost you a mere \$2900, or so. The engine, however, will noisily grind and sputter to a halt before everything gets dirty, clogged, or broken.

Injured Engine will teach you the names and general locations of major engine parts and a little about their function. As a teaching tool for youngsters, it is analogous to the plastic "Visible V-8 Operating Engine" model (Revell, Inc., Venice, CA 90291) that has been available in toy and hobby shops for years. Don't expect, however, to gain the expertise to do major (or minor) repairs yourself. Perhaps the most important benefit that you will gain by using this game is an increased awareness of your automobile. You may be a bit more savvy when you take your car in for repair and know more about the repair lingo that garage mechanics love to rattle-off.

HCM

SIDEKICK

A Review by Dana M. Campbell
HCM Staff

They sized each other up slowly, keeping a wary eye alert for any tricky fast moves. Finally, Smedley snarled "There ain't room in this office for both of us, Jones." With that he reached for his weapon of choice just as Jones yelled "DRAW MISTER!" Suddenly, a burst of activity exploded in each workstation, and the bystanders ducked for cover.

By the time the dust settled, Jones had entered into his computer appointment-book all flight connections for his upcoming business trip, dialed the reservations, and typed in a few notes for later use in the presentation that he was writing on the screen. Smedley had merely turned his desk upside down in search of a calendar.

"Let me ask ya Jones," Smedley inquired dejectedly, "how'd ya do it?"

"Why, I called on my trusty sidekick, of course."

Sidekick is a "desk utilities" program that hangs around in the background of Jones' system's memory bank just waiting to be summoned—even when he is working with another program in memory. All he has to do is press the (ALT) and (CTRL) keys simultaneously, and the Sidekick menu pops up on the screen. It provides a notepad, calculator, appointment calendar, auto dialer, and an ASCII table in the form of colorful pop-up windows. Just one or two keypresses quickly takes him from one utility to another (and another and another all over the screen, if desired), and from any program to Sidekick and back. A smart Help window may also be called up for information appropriate to Jones' location in the program.

Keep Notes

The Notepad is a handy little empty window with full-screen text-editing capabilities that allow Jones to, for example, create and work on files within files, write memos, or leave a note for himself the moment a thought occurs to him. It can work with files ranging from 1,000 to 50,000 characters in size, with the same amount of memory reserved for the actual Notepad files. Notepad resides in memory to permit disk swapping, but its file size can be adjusted to work with smaller- or larger-sized files.

Jones appreciates the 48 Notepad commands, which include all of the functions of the Turbo Pascal editor, and most of those from the popular WordStar word-processing program. These include "find and replace," sorting, all kinds of cursor-movement and insert/delete options, printing, screen dumps, and time and date stamps on notes. Probably the most remarkable quality of the Notepad is its block-marking capability, which can be used to integrate parts of different files—i.e., inputting selected spreadsheet cells into a memo stored in the Notepad. This is what Jones did for the sales presentation he was writing.

At times, however, Jones becomes frustrated with his computer friend because there is no way to format anything in the Notepad itself; no way exists to set margins, there's no word wrap, and no fixed tab settings. However, tabs are automatically set to the beginning of each word on the line above the cursor.

This sidekick isn't a crusty old movie cowboy, but a set of desk tools that work just as faithfully.

Make Calculations

One of the first things Jones discovered about Sidekick is that—whether he's working on a spreadsheet or writing a program—the value of being able to make calculations right on the screen is immeasurable. Sidekick's 18-digit Calculator (including 4 decimal places) uses binary-coded decimal (BCD) arithmetic for accuracy. It works with hexadecimal and binary numbers in addition to decimal numbers, and converts a number in one of these modes to its equivalent in another. A memory function also exists.

Jones often likes to use the option that lets him transfer his calculations to other programs by programming the calculated total onto a key, and then hitting that key when he finds the right spot in the main program to insert it. However, there is no option for clearing just one programmed key—it's all or nothing.

Set Up Appointments

Jones used the Calendar window to store in one easily referenced place all of his flight arrivals and departures and other appointments. The Calendar lets him set up and name a separate calendar schedule for each person in the house and office for any day or every day from 1901 through 2099. The cursor

keys are used to quickly advance or backtrack days, months, and years. A line is available to name each day's schedule if desired, and each day is already divided into half-hour increments to enter appointments, reminders, etc.

Dial An Associate

On command, the Dialer will automatically dial any number on the screen from any program, or from a phone file that Jones set up using the Notepad. Once the Dialer is activated, the first number on the screen that the program recognizes as a phone number is pointed out, and will be dialed (via separate Hayes-compatible modem) once Jones gives the O.K. Or, he can immediately skip to the one he desires. Phone directories constructed using the Notepad may include each number, with a comment and an "identifier" that may be used with the Search command to speedily locate and dial a number.

Jones saved time during the "shootout" by going right to the Dialer window and highlighting the airlines' names and numbers that he had previously entered into his Calendar window. The Dialer automatically rang the reservation desks while Jones continued working.

The worth of the ASCII table became apparent the first time Jones flashed it up on his screen while working on a program. Instead of trying to recall the location of

"... whether working on a spreadsheet or writing a program, the value of being able to make calculations right on the screen is immeasurable."

a book in which an ASCII table resides, finding it, looking it up, and then returning to his program. Jones simply popped the table up on screen with one command. This ASCII table displays the entire 256-character set in decimal and hexadecimal numbers, the way they look on the screen, and the control character and mnemonic of each character.

PCjr Problems

Jones found that although *Sidekick* runs on the IBM PCjr in addition to the PC, the PCjr version suffers from several problems. These difficulties arise essentially because of differences in the way the two machines use memory—in particular the screen-memory buffer. The IBM PC always uses the same area (starting at location \$BFOOO). On the other hand, PCjr applications can move the buffer area to virtually any location in memory. But *Sidekick* always looks for the screen data at the same location, so when the *Sidekick* menu appears on the PCjr, your old screen may take on a rather strange appearance.

Jones noticed this problem when trying to use the calculator while running a BASIC program that used SCREEN 1 (Medium-Resolution Graphics). He pressed the (ALT) and (CTRL) keys, and as the *Sidekick* menu popped up, a screen full of IBM hieroglyphics took the place of his BASIC program. By exiting *Sidekick* with (ESC) his program returned to normal operation—but the background effect of *Sidekick* was lost.

These memory problems affect BASIC in another way, too. When accessing Cartridge BASIC, there are normally 60,130 bytes of memory available for program space. However, *Sidekick* uses some of the memory normally used by BASIC, thus reducing available program memory to only 23,586 bytes—a loss of 36,544 bytes!

Memory conflicts are not the only problem with *Sidekick* on the PCjr—it is presently incompatible with the internal modem as well. These problems are not necessarily "terminal" however. Borland International assured us that a new version (version 1.5, as opposed to 1.11C, which was used for this review) will take care of a number of these difficulties, so Jones will probably not want to take a full-featured *Sidekick* home to Junior until he finds an updated version.

A Versatile System

If Jones really wants to, he can get around this memory problem, however, by loading one of three scaled-down configurations of *Sidekick*—such as the Calculator and ASCII Table exclusively—appropriate for various applications. When he needs more freedom, he defines his own Notepad commands, and alters the defaults for the screen displays, window sizes and positions, file names, and directories.

Jones thinks that the manual that accompanies the disk is easy to understand, clearly detailed, and logically organized. Even Smedley could figure it out. Some suggested application scenarios are provided at the back of the book that are both amusing and helpful.

As if the 5 main options and their ease of use alone weren't enough, the reasonable price of *Sidekick* (especially in reference to the cost of programs that offer only one or two of these functions) set Jones back less than the cost of a Hewlett Packard Programmer's Calculator. After putting *Sidekick* on your desktop for a little while, it is easy to understand why the Smedleys of this computing world will never again be able to keep up with the Joneses.

HCM

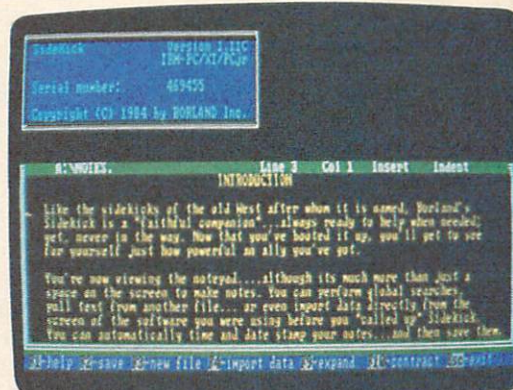
HCM Glossary terms: ASCII, binary numbers, binary-coded decimal arithmetic, control character, default, file, hexadecimal numbers, mnemonic, spreadsheet, word wrap.



Name: Sidekick
Program Type: Desk utilities.
Machines: IBM PC and PCjr
Distributor: Borland International
4113 Scotts Valley Dr.
Scotts Valley, CA 95066
(408) 438-8400
Price: \$49.95

System Requirements: None

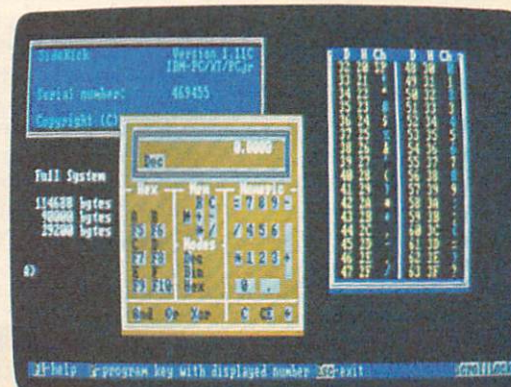
	Poor	Fair	Good	Excellent
Performance:				
PC				
PCjr				
Ease of Use:				
Documentation:				
Cost/Benefit:				



Sidekick's Notepad window, with demo text. This window may be enlarged or reduced to better view material "under" or within the Notepad.



The Appointment Calendar will scroll through every month from 1901 through 2099, as well as show each day's appointments in half-hour increments.



Part of the ASCII Table, which scrolls through all 256 characters, and the Calculator.

To a substantial number of people out there in the "real world," computers are very scary things. Science fiction movies and books have often portrayed them as evil machines bent on taking over the world. Who knows? Maybe they will someday—whoops, just kidding. At the very least, some people avoid computers because they fear that they won't be able to understand them. The darn things are just too complicated.

Judging by all the thick manuals that you have to wade through when you buy a new computer or one of those "heavyweight" software packages, you might tend to agree. Not surprisingly, some members of the computer community have picked up on this fear and are trying to make computers seem a little more "friendly." One software package that claims to be friendly is Arktronics' *Jane*, a home productivity package for the Apple II Family of computers, and soon to be available for the Commodore 64.

Jane uses icons and a mouse (à la Macintosh), and boasts a simple, easy-to-understand format. It comes with three separate programs: *Janewrite* (sound familiar?), *Janecalc*, and *Janelist*, plus a short tutorial. *Jane* works with either the Arktronics Mouse, the AppleMouse II, the Koala Pad, or a joystick.

Using Jane

Jane works best when you use a mouse or Koala Pad to move the cursor and select icons which perform specific jobs like editing, copying, organizing, etc. In this case, the cursor is shaped like a hand with a pointing finger. You move it over the icon you desire, and click the mouse button to select it. Once selected, the icon is highlighted, so you can tell at a glance which icon you are executing. The main-menu screen houses 13 icons, three of which select the programs *Janewrite* (depicted by a typewriter), *Janecalc* (a calculator), and *Janelist* (a file cabinet). Selecting one will start the corresponding program, and from there you will see even more icons and images specific to the particular program you are using. The other 10 main-menu icons access the editing and file-manipulating functions of the three programs.

And Now a Word From Janewrite

Selecting *Janewrite* (the typewriter icon) calls up the word-processor portion of *Jane*. Although it is not as powerful a word processor as some of the more well-known brands, *Janewrite* offers enough features to satisfy most around-the-house needs. *Janewrite* offers left, right, center, and full justification (edge alignment) of text. Unfortunately, it only justifies text *after* it has been generated, instead of as you're writing. Thus, you must select which type of text justification you want, click the mouse button, then select the text to be justified, and click the mouse button again in order to justify your text. This is just a minor annoyance compared to the much larger problem that you will discover when you begin typing.

Jane uses windows to display text and files. The window, however, displays only a portion of the text you are working on, resulting in a very distracting problem: When you type beyond the boundaries of the window, the screen must shift over to make room for your text. Although this shift takes just a moment, it's long enough that you may temporarily lose sight of the last bit of text you entered. If you are a fast typist, you can get far



enough ahead to cause even more aggravation. In order to compensate, you must slow down or stop typing, and wait until the screen shifts over—displaying more of your text. *Janewrite* operates in 80-column mode, but you can only enter 51 characters before you run out of window.

Another partial solution to this window-shifting problem is to select the computer icon that allows you to set the size of your text to either small, normal, or large (with normal being the default size). Changing to small will give you some relief, but it's not a complete cure—the shift can still lag behind your typing. Visually, this is extremely distracting and quite a nuisance.

Of course, you can always set the margin under 50 columns, but if you want a hard copy in 80-column, you have to reformat after you are through typing.

"While it is not as powerful a word processor as some of the more well-known brands, Janewrite offers enough features to satisfy most around-the-house needs."

Editing text with *Janewrite* is awkward at first, but it gets somewhat easier with practice. My main complaint is with the speed of the mouse. When changing icons and moving through text, it is quite slow. Another problem involves positioning the cursor (in whatever form it may be) into the text when editing.

It is difficult to place the cursor exactly where you want to add or delete text. Whenever I tried to find my edit point, I often found myself either in the sentence above or below the point where I wanted to be. This aspect of *Jane* caused me a lot of frustration.

Jane can adjust all margins to tailor your letter or document to the desired size. Line-spacing, titles, and page-length features are also included. Once the file is structured the way you want, you can get a hard-copy printout on either a single page, or on continuous paper. You can print your text in normal or bold type, with underlines, underlines and bold, and even subscripts or superscripts. *Janewrite* also has a search feature that allows you to find a specific word or phrase in your file. You can use this function to find and change specific words if you wish.

Calculator/Spreadsheet

Janecalc is a mini spreadsheet program that you can use for just about any home financial function. You could even apply it to a small business, but it really isn't designed for that type of use. The familiar window surrounded by icons shows you one portion of your spreadsheet at a time.

The spreadsheet is divided into a series of 18 rows and 24 columns which form cells. Each cell can be used

to store text, numbers, or equations. At the top of the window, beneath the icon row, is the display bar. When you enter equations or numbers into a cell, they first appear in the display bar before they are placed in the proper column. All spreadsheet calculations are performed in the display bar, and it can even be used as a one-time calculator. After each calculation that affects the spreadsheet has been computed, the program automatically updates all columns affected.

Entering data into your spreadsheet using the display bar and the hand icon is a slow process; however, once the data is all in, I like the way the program works. It's easy to enter new data, or make changes, and *Janecalc* lets you cut portions out of a *Janecalc* file, and paste them into a file in *Janewrite*.

The Electronic Filing Cabinet

Jane gives you a third program, called *Janelist*, which is an electronic filing cabinet. It lets you store addresses, make lists of all kinds, and it even prints out mailing labels. When you select *Janelist*, you may want to use one of two ready-made files: a business address mailer, or a personal address book. *Janelist* will also let you create your own file format. If you make your own file, you can enter up to 10 items, each up to 25 characters long. At the bottom of the window is a series of icons that you may select in order to sort through your files. *Janelist* will automatically search your files for a specific title, name, or letter. Files can be sorted alphabetically or numerically, and they can be printed out in a variety of formats.

Janelist stacks up as the better of the three programs, probably because it is the simplest to use. The general slowness of the mouse inhibits it some, but overall, it is *Jane*'s friendliest offering.

Printing mailing lists is the final feature of *Janelist*. You can enter as much information as will fit on the size of label that you are working with, and you can format it any way you wish.

The documentation provided is well done, and the graphics and sound effects are not objectionable—and I don't recommend a color monitor because it's basically a monochrome system.

Sorry But . . .

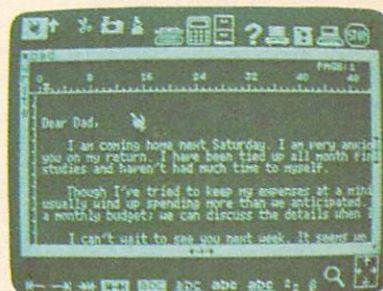
Jane is an interesting program—one that at first glance would seem to have a lot of potential, but instead clocks in on the clumsy side. Generally, everything seems to work, but I find myself feeling that *Jane* really isn't too great a package. It's just too unwieldy. The main problem is that there are too many different icons, and they are a little too "cutesy." Couple that with the general slowness of the mouse, and you wind up with a program that really doesn't do anything to make your job that much easier. I think it would have been better for Arktronics to combine some of the similar functions—e.g., all the editing commands—into just one icon.

The problem with the windows is one that I personally cannot accept, but it may not bother other people. The best thing to do is go down to your local computer store and give *Jane* a try before you buy it.

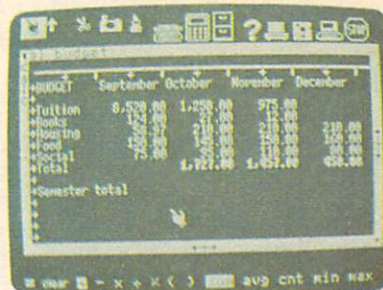
You don't have to look far to find excellent productivity packages for the Apple II Family of computers. *AppleWorks* offers a more powerful word processor and spreadsheet, and is much faster. [See our review of *AppleWorks* in Vol. 5, No. 2—Ed.] And when you consider the added cost of purchasing a mouse with *Jane*, *AppleWorks* is competitively priced.

Name: Jane
Program Type: Integrated data base, spreadsheet, word processor.
Machines: Apple II Family
Distributor: Arktronics
 P.O. Box 4190
 Ann Arbor, MI 48106
Price: \$125
System Requirements: Disk drive. Mouse recommended.
Performance: Poor Fair Good Excellent
Ease of Use: _____
Documentation: _____
Cost/Benefit: _____

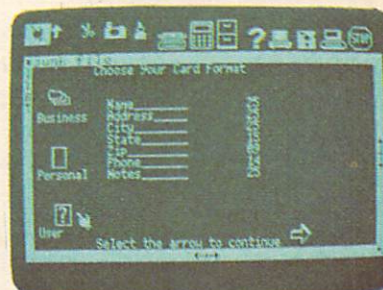
Janewrite:
 Notice that the text continues past the edge of the window. When working in the 80-column mode, the window must shift over when you type beyond the border.



Janecalc:
 Notice that the calculator functions at the bottom of the screen. These are used for all spreadsheet calculations, and can even be used for one-time calculations.



Janelist:
 Personal format for card file. This is a standard format; but *Janelist* also gives you the option to create your own file format.



The concept of making computers friendlier is a worthwhile one, but unless the "friendly" computer or program helps us perform a task easier, there really isn't much of a point in having it around. *Jane* will do everything it says it will, but it doesn't offer that much improvement over what you can do without it. **HCM**

COUNTERPOINT

How does *Jane* run? Pretty slow. Even though *Jane* performs the functions it promises, its actual performance defeats its objective—that being an easy-to-use program. It is friendly in the sense that its icons and manual help you slip comfortably into the computing world, thus avoiding intimidating commands and procedures. However, the difficulty encountered in using the mouse, pointers, and windows condemn *Jane* to being merely a program created with good intentions that failed in its application.

—Dana M. Campbell

S lay a dragon, explore the moons of Jupiter, the bottom of the sea, or myths from the history of mankind. All this and more is to be found in the interactive fiction realm of computer gaming. Computers are almost ideally suited to adventure gaming—their operating speed and memory provide for fast-paced, error-free game play. Computers allow such games to be enhanced with maps, floor plans, and graphic representations of people, places, and things.

Early computer adventures, such as *Hellfire Warrior* and even more archaic text and graphic games, were quite popular with the more dedicated crowd but suffered eventually from the "player burnout" syndrome. Even the finest story or cleverest plot begins to wear thin after five or six exposures—so new adventure software was always needed. Now, however, two new software systems—*Adventure Construction Set* by Electronic Arts and *Adventure Master* by CBS Software—allow you not only to play, but to actually create new adventures for the game systems, cut from the cloth of your imagination.

Both systems allow input of text and graphics to describe outdoor areas, rooms, creatures, things, magic spells, special effects, and conditions of movement in the adventure. Beyond that point, there is little similarity between the two. Given the numerous and substantial differences between the packages, I think an analysis of each system will be much more appropriate than a point-by-point comparison.

First, The Good News

Stuart Smith's *Adventure Construction Set* (ACS), is an easily understood, elegant, and enjoyable scheme that provides something very near to a complete theater with actors, set construction crews, and a prop department—a theater that seems limited only by your imagination. In game play, up to 4 characters can participate at one time, with only one joystick required to control all 4 (although two sticks may be used).

All characters are generated by ACS on command, and you may save, transfer, or destroy them at your whim. They are firmly rooted in the classic *Dungeons and Dragons* tradition of each being an individual with their own inborn traits, such as wisdom, constitution, and dexterity. These traits are randomly generated, so no two characters are ever identical. When a character's turn comes up, it is depicted on screen along with its surroundings and companions. All objects and character figures are selected from an editable "palette" or kit, which also contains symbols for mountains, forests, wall elements, etc. Movement is depicted with a simplified kind of animation controlled by joystick.

Also shown on the screen are three vertical color bars: red and green to the right of the playing "field," and green to the left. These represent your life force, magical power, and movement rate—all of which vary during each turn. Beneath the encounter scene is a blue menu screen that allows the player to access ten options for game action as well as for adding, subtracting, or saving a character or game.

In melee action, the program generates hits and misses and displays screens for all combatants involved in each round. The program also performs the many tedious but necessary bookkeeping tasks required to lend reality to adventure games. You soon learn the wisdom of the old saw about "traveling light" as the

Worlds in Creation

A Review of Adventure Construction Set and Adventure Master

by Scott Darroch

Turn your wildest imagination into your own personal adventure game with these innovative software packages.

program relentlessly inhibits your movement the more you acquire additional possessions. As you move through an adventure, your character also grows in ability and knowledge in the use of weapons and magical skills.

Characters become somewhat sluggish when a party of four attempts to travel across country. Otherwise, movement is quite smooth, with one exception: If two characters bump into each other while holding melee weapons, an automatic one-round fight begins. While this is somewhat annoying, I have never lost a character to such intramural violence.

Encouraging Creation

That the game plays in such an enjoyable and responsive manner is a tribute to *Adventure Construction Set's* entire system, which—with its excellent documentation and ease of operation—not only allows, but actually encourages the creation of a milieu that is rich in complexities, multi-valued, and truly interactive.

When entering the ACS program, you are presented with a menu displayed in the form of a decision tree. Here, you may change the adventure's name, byline, theme music, and text introduction, or try saving, exiting, or letting the program finish your game setup. You also have a remaining option to "do more detailed work." This option takes you to a series of decision tree menus. The second level contains features to let you edit your world map, change the creatures on your map, edit or add a region to the map, or once more "do more detailed work." The final level of the tree offers options for editing the master lists of creatures and things, or the master graphics program. Both of the two latter menus deserve a closer look.

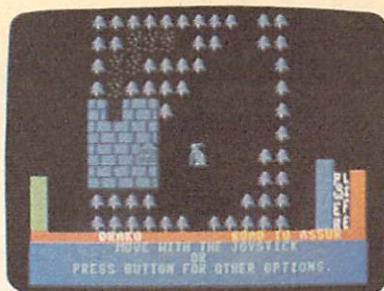
The Map level is where you draw your world map, choosing from 16 prepared terrain features; or, create your own from scratch. Using the joystick, you may place selected terrain where you will. The screen also displays a small lighted square within a larger grey one to indicate on which portion of the world map you're drawing. Editing and Adding Regions involves drawing the floorplans of encounter areas, examining and determining the contents of the rooms, and setting entrances and exits to rooms and regions.

Editing Creatures accesses your adventure's list of monsters, those both deliberately placed and randomly

"If Adventure Construction Set is a theater, full of props, sets, and actors, then I can only say that the Adventure Master program is little more than an empty stage with some fancy lights thrown in."

This is a screen from
Adventure Construction Set's
built-in program, Rivers of Light.

   	
Name:	Adventure Master
Program Type:	Text adventure
Machines:	C-64, IBM PC, PCjr, Apple II Family
Distributor:	CBS Software Inc. One Fawcett Place Greenwich, CT 06836
Price:	\$44.95
System Requirements:	none
	Poor Fair Good Excellent
Performance:	██████████
Engrossment:	██████████
Documentation:	██████████



	
Name:	Adventure Construction Set
Program Type:	Adventure game
Machines:	Commodore 64
Distributor:	Electronic Arts 2755 Campus Drive San Mateo, CA 94403
Price:	\$40
System Requirements:	Disk drive
	Poor Fair Good Excellent
Performance:	██████████
Engrossment:	██████████
Documentation:	██████████

Graphic screen from the *Clever Catacombs* of Adventure Master. This image was created using the graphics program.

encountered. The third menu level enables you to manipulate your master lists of creatures and things, and to edit the graphics palette itself. Creature lists access 8 groups (friends, enemies, neutrals, etc.) that you may add to or mutate at your pleasure. The "thing" list is composed of 13 groups of items—from weapons (including laser guns), to doors (or transporters), to open space obstacles of your own making.

Entering the graphics program gives you access to 3 lists: terrain, things, or creatures. When a list is selected, all of its items are shown. You may then bracket any item, and a magnified picture of it comes up so you can add, delete, or change the color. You may also bracket an empty space and create your own picture using the color swatches and cursor. The resolution in the graphics edit window is similar to *MacPaint*'s fat-bits mode, where any single pixel can be turned on or off at will. Like *PCjr ColorPaint*, the system's only major drawback occurs when you change a color that has already been used—the color changes *everywhere* on the screen. The third-level menu also contains instructions for copying and erasing game files.

Adventure Construction Set is currently available for the Commodore 64, but versions for Apple II Family and IBM PC and PCjr computers are forthcoming.

And Now . . .

If *Adventure Construction Set* is a theater, full of props, sets, and actors, then I can only say that the *Adventure Master* program from CBS Software is little more than an empty stage with some fancy lights thrown in. *Adventure Master* is, simply stated, a program for construction of a text game, with a skimpy and difficult graphics system added on.

In play, the system is somewhat similar to *King's Quest*, but it lacks the graphic input needed to visualize the situations presented to you. What graphics there are eat up a large amount of memory, which cuts into the number of rooms, areas, and passageways that can be described or depicted—so most information is provided by short bits of text. Because the game can be adjusted to require a bewildering array of magic passwords or items that must or must not be taken, the game can quickly deteriorate into an endlessly frustrating game of "rats in a maze."

When using *Adventure Master* to program an adventure, you may experience a curious sensation: From the near-total *restraint* of your character in gameplay, you as author have almost total *control*. You can set goals, describe places, plan events, name secret passwords, hide objects, and draw on an empty canvas to illustrate your game. In fact, you are required to do all of these things as a prerequisite to playing, because *Adventure Master* is nothing more than an empty data-base program, a series of labeled boxes into which you place your written instructions. Unfortunately, the player is so restricted by these "rules of the game" that he or she can get frustrated easily. Only when all preconditions are met may the player advance or receive any messages other than "You cannot go that way" or "I do not understand that."

Documentation is clear enough in explaining how to construct the adventure, but does little to help the player avoid or escape these "dead end" situations.

This program is no worse than many text adventure programs floating around, but it is certainly no better than many other similar programs available at little or no expense. The addition of a graphics section, which even a commercial artist found difficult to operate, is far from sufficient compensation.

Adventure Master is available for the Commodore-64, Apple II Family, and IBM PC and PCjr computers, with no real notable differences visible between versions.

[Note: TI-99/4A owners, don't feel left out. We just received *TI Adventure Editor*, a text-adventure game-builder for evaluation. Watch for a review in a future issue.—Ed.]

Last Words

After looking at the two program packages, I was struck by the often misleading nature of first appearances. Stuart Smith's *Adventure Construction Set* comes packaged in cardboard, and a collage of stock shots and pictures of friends adorn it. The CBS product, on the other hand, is stylishly turned-out in splashy professional graphics and a top-of-the-line rigid plastic case. But once you complete an inspection of the contents, you may feel that on one side of you is a simple bag, loaded to the top with treasure—on the other, a handsome chest which is, to your dismay, almost empty.



Romancing the PCjr:

A Review of the Quadjr Expansion Chassis

by David G. Brader
HCM Staff

This method of adding to PCjr's powers looks promising, but it has its pitfalls . . .

People who have (or are able to acquire) an IBM PCjr are indeed fortunate. Not only do they receive a computer with great color graphics capabilities, but they may also greatly expand its overall processing and storage capabilities by adding hardware and software. We have reviewed the *Tecmar JrCaptain*, the *Legacy II*, and other smaller attachments for the PCjr. [See *Jr. Addition: A Review of the Tecmar JrCaptain Peripheral* in Vol. 4, No. 4, and *Legacy II for the PCjr* in Vol. 5, No. 2—Ed.] Continue now with me in exploring yet another option for increasing the PCjr's capabilities through Quadram's *Quadjr Expansion Chassis*.

Beauty Is Only Skin Deep

The first thing that is apparent when examining the *Quadjr Expansion Chassis* is its IBM-quality look. Quadram definitely knows how to produce a well-finished unit. If the case didn't sport the Quadram logo, I would swear it had been built by Big Blue. But looks can be deceiving, can't they?

The *Quadjr Expansion Chassis* as supplied is composed of three major pieces of

hardware: (1) the external power supply module, which has nearly the same appearance as the PCjr's; (2) the main chassis (containing the expansion slot and a second disk drive), which snaps on top of the PCjr and in turn has the PCjr's cover snapped on top of it; and (3) the right-hand Side Assembly module which contains the parallel printer interface, battery-powered clock/calendar, and a switch for "PCjr" configuration.

For an additional fee, you can obtain the *Quadjr Memory Board*, which comes with 128K of RAM and is expandable to 384K. This board must be mounted inside the *Quadjr Expansion Chassis* prior to the chassis's attachment to the PCjr.

The *Quadjr Expansion Chassis* system makes use of the IBM disk-drive controller board instead of replacing it (as with the *Legacy II* expansion system). The special cable supplied with the unit is split, so some of the signals come from the added circuits in the Side Assembly module of the *Quadjr*.

My testing environment for the *Quadjr Expansion Chassis* consisted of a Sears 13" TV/monitor operating

in RGB mode, an expanded PCjr (128K RAM with a single disk drive) with an IBM PCjr modem card installed, and the Quadram *Quadjr Expansion Chassis*, which contained a second floppy-disk drive, a parallel printer port, a battery-powered clock/calendar, and the optional memory board. This *Quadjr Memory Board* had 384K of RAM, for a system total of 512K.

Hardware Reliability

The first *Quadjr Expansion Chassis* received for review failed to function at all. After trying it on two different PCjrs, I contacted Quadram. They immediately sent another unit, this time with the 384K memory board option. But after running this second system for about four hours, it started to behave erratically. In trying to restart it (by powering down and back up again), I consistently obtained an Error H message, an indication of disk drive failure. And, the disk drives were not

"With the dictionary loaded into the RAM disk . . . You will be amazed at the new lightening speed of the spell-checking function."

accessed by the system at all during the boot-up attempt. After I left the power off for about five minutes or more, the system would once again boot properly. Changing to a different PCjr (one without a modem board) seemed to fix the problem. (The first unit that we obtained from Quadram had the same symptom as soon as it was first started up—marginal design?) I should note that both PCjrs used in the tests work well with other equipment and attachments. Over a period of three days I tried to get through to the Quadram Technical Support people, leaving messages to call back each time—I was not successful. On the fourth day, Quadram returned the calls, and expressed surprise at the hardware problems we were having.

Checking The Paperwork And Software

Two documents come with the Quadram package. One is the *Quadjr Expansion Chassis Installation Manual*; the other is the *QuadMasterjr User's Manual*, which describes the use of the software contained on the QuadMasterjr diskette supplied with the hardware. Both booklets are well-organized and clearly written, but the software manual should describe several specific configurations for popular software packages. This would be most useful for new owners who just want to

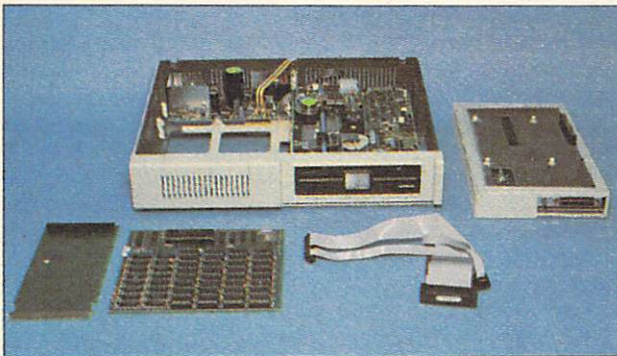
get their favorite spreadsheet or word processor operating. I spent a great deal of time discovering combinations of software options that *won't* work with a given application—a real pain. The only software I ran that didn't require special attention was Microsoft's *Flight Simulator*, which did recognize the extra memory automatically, thus allowing the program to use this memory for better performance.

Features of the QuadMasterjr software diskette include programs to "install" RAM disk drives, printer "spoolers," and software to utilize the clock/calendar hardware and move the video buffer into a different area of memory. A program called QSWAP allows software swapping between two parallel printers (in case you have a second parallel port configured into your system).

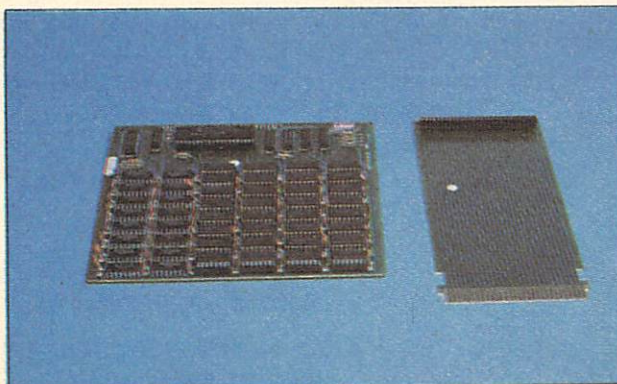
Trying to set up this system for a specific application using the generalized documentation provided is hit and miss, unless you have a working knowledge of the PCjr's hardware design. In some cases, the printer buffer will not work; in others, having the video screen memory relocated causes the system to hang up. If you try to make full use of this system without this knowledge, get ready for unexpected results and some frustration.

It Works Great . . . Almost

The software supplied on the QuadMasterjr diskette can be used to build up a special CONFIG.SYS file containing the "device drivers" that you wish to employ in a specific system configuration. For example, if you wish to use the IBM Writing Assistant software and make use of the 384K of RAM that is optionally installed in the Quadjr, you might configure the system to have the



The Quadjr Expansion Chassis comes with a special cable for connecting the second disk drive to the Junior, and a module that fits to the side of the expanded Junior chassis containing the battery-operated clock/calendar, parallel printer port, and other control circuitry. Also shown is the extra-cost memory card and its bus-extender card. Not shown here is the external power supply module that sits on the floor (similar to the PCjr's).



The optional Quadjr Memory board is purchased with 128K of RAM and comes with an adapter card that connects it to the Quadjr bus. The board has room for additional RAM chips which the owner may install to add an additional 256K for a total of 384K as seen in this photo.

Name:	Quadjr Expansion Chassis and Quadjr Memory Board
Product Type:	Hardware/software expansion system
Machine:	IBM PCjr
Distributor:	Quadram Corporation 4355 International Blvd. Norcross, Georgia 30093 (404) 923-6666
Price:	Quadjr Exp. Chassis: \$675 Quadjr Memory Board with 128K of memory: \$275
System Requirements:	none
Hardware	Poor Fair Good Excellent
Performance:	████████████████████
Software	
Performance:	████████
Ease of Set-up:	████
Documentation:	████████████████████
Cost/Benefit:	████████████████████

video buffer in the lower portion of memory, and part of the expansion RAM dedicated to a RAM disk that holds the spell-check dictionary and IBM Writing Assistant application programs. This configuration can be set up in the CONFIG.SYS file (general instructions are included in the *QuadMasterjr User's Manual*) and placed on a special boot disk that is used with the IBM Writing Assistant diskette. Then, when you boot from this disk, it will configure the system specifically for the IBM Writing Assistant software.

Here are the steps for one way that I managed to configure the Quadjr system with memory expansion, as described above:

"I spent a great deal of time discovering combinations of software options that won't work with a given application—a real pain."

Step 1: Build a PC-DOS boot disk by using the FORMAT /S command from PC-DOS and then use the COPY command to copy the following files from the Quadram QuadMasterjr software disk onto this disk:

```
JRVIDEO.SYS
RAMDRIVE.SYS
QSPPOOL1.SYS
QUADCLOCK.SYS.
```

Step 2: Generate a CONFIG.SYS file containing the following lines on the same disk:

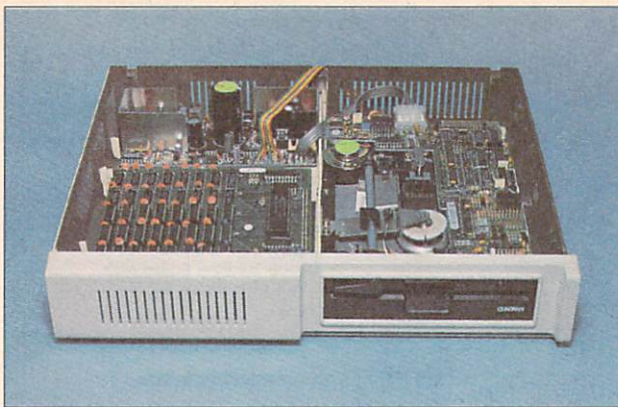
```
DEVICE = JRVIDEO.SYS
DEVICE = RAMDRIVE.SYS 300
DEVICE = QSPPOOL1.SYS 60 48
DEVICE = QUADCLOCK.SYS
```

Step 3: Next, on your IBM Writing Assistant diskette, modify the G.BAT file to contain the following lines only:

```
COPY A:WRITE.* C:
COPY A:WORDPRF.* C:
C:
WRITE
```

Step 4: Place the "PC/jr" switch on the rear of the Side Assembly module in the "jr" position with the power off.

Continued



Here you see the Quadjr Expansion Chassis with the Quadjr Memory card and its adapter installed. The memory board adapter must be plugged into the memory card on one end, pass under the disk drive, and plug into the side assembly module on the other end.



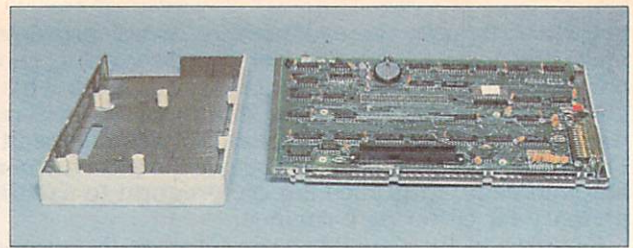
The Side Assembly module has a switch to change from PC mode to PCjr mode. The benefits provided by this switch are uncertain. The connector for the parallel printer port is visible here. This port may be treated the same as the IBM parallel port LPT1.

Now power-up the system with the special boot disk (just described in Step 1) in drive A, then replace the boot disk with the modified IBM Writing Assistant disk and type G followed by a carriage return. Your system will now have all of the IBM Writing Assistant applications, including the dictionary, loaded into the RAM disk—it will operate much faster this way, and you can put your data files on a diskette in Drive B. You will be amazed at the new lightening speed of the spell-checking function.

When using the IBM Writing Assistant with the Quadjr-equipped PCjr as described above, I was able to build a document about three and one-half pages long before running out of memory. While I printed the document (actually causing it to go into the printer buffer) the prompt indicating that the system was ready appeared on the screen—but an attempt to continue working with the system while it was printing caused it to hang up. So much for printer buffer use in the PCjr mode . . .

PC Or Not PC—That Is The Question

The "PC/jr" switch at the rear of the Side Assembly module does change the system so that it boots in 80-column mode (when in the PC position), but other changes that it may cause are not evident. The manual states that in the PC position you can "run PC-dedicated software." But I tried to run the IBM Writing Assistant program in this mode, without success.



Taking the Side Assembly apart reveals the user-replacable battery for the clock/calendar. The clock/calendar can be automatically "installed" any time the system is booted up.

Careful Consideration Needed

The clock/calendar feature and parallel printer port work as expected—without any surprises. If you can find a unit that is reliable and get optional memory configured the way you desire, the additional capabilities provided by the Quadram Quadjr Expansion Chassis make the PCjr a formidable computing machine.

It would be unrealistic to try to explore all of the possible combinations of Quadjr's options with numerous software applications in this short review. I suggest that if you do decide to purchase the Quadjr Expansion Chassis for your Junior, insist that the dealer set up your first two or three application diskettes. Make sure that they work flawlessly, and then try the Quadjr on your own PCjr in the store, if you can, before you commit to the purchase.

HCM

COUNTERPOINT

Quadram promises to provide a remedy for the main problems mentioned in reference to the PCjr—limited memory and only one disk drive. After trying several PC products that won't run on Junior without these additions, I have to say that Quadram doesn't keep its promise—they still don't run. The expansion to 512K of RAM looked good when I saw it on the initialization screen, but my pleasure turned to disappointment every time I tried to access it. The only exception to this occurred when I used the RAMDRIVE—it proved to be a real time-saver when programming in BASIC and swapping different files in and out of memory. However, there are many memory expansion packages on the market. If savvy PCjr owners decide to look past the Quadjr's flashy exterior, the lion's share of this add-on market will probably opt for a more "workable" alternative.

—Roger Wood.

HCM Glossary terms: application program, boot disk, CONFIG.SYS file, device drivers, parallel port, printer buffer, RAM disk drive, spooler, video buffer.

Thinking of Subscribing? Remember these time-worn truths:



"A watched pot never boils."

"Patience is a virtue."

"Good things come to those who wait."

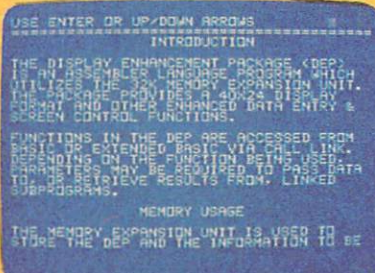
"Allow 6-8 weeks for delivery of your first issue."

HOME COMPUTER
magazine

THE DISPLAY ENHANCEMENT PACKAGE

A Review by David Reese

HCM Review



Name: Display Enhancement Package
Program Type: Utility
Machines: TI-99/4 and 4/A
Distributor: Oak Tree Systems
3922 Valentine Rd.
Whitmore Lake, MI 48189
Price: \$29.95
System Requirements: 32K expansion, and one of the following configurations:
Mini Memory, cassette recorder and cable;
Mini Memory & disk drive; Extended BASIC & disk drive; Editor/Assembler and disk drive.
Performance: Poor Fair Good Excellent
Ease of Use:
Documentation:
Cost/Benefit:

*A display that seems as big as Texas
when you're squeezed for space under normal 28-column mode.*

I was beginning to get tired of all those snickers from salespeople and friends with so-called "legitimate" computers. You've heard the remarks. "You use a TI? When are you going to get a computer with some real features?"

About one feature, I knew they were right—the screen display. The 99/4A's 28 columns of text is simply not a feature, it's a handicap. Luckily, however, I found Oak Tree Systems' 40-column *Display Enhancement Package* (or *DEP*).

The *Display Enhancement Package* requires the 32K memory expansion. It is available in three different versions: Editor/Assembler, Extended BASIC, or Mini Memory. Each version is provided on diskette, with the exception of Mini Memory—it comes on tape.

"Real Features"

I use the Extended BASIC version, and swear by it. I must admit, however, that when I received the program, I was a little disappointed. I thought that I would be able to insert the disk, press a few keys, and be in 40-column Extended BASIC! In fact, what I found was a very powerful series of assembly-language subroutines.

These routines include not only a 40-column display, but also a four-page "data storage area" that may be filled with information, then moved up and down either a line at a time, or a page at a time. Additional options include the ability to set up a fixed area at the top of the screen that displays information while "working" data is scrolled up and down behind it, and a number of data-entry features that surpass even Extended BASIC!

The *DEP* is more than a program, it's a programming language—an "Extended Extended BASIC." After all, one of the things that makes TI's version of BASIC so powerful is the number of assembly-language subroutines available with *CALL*—*CLEAR*, *HCHAR*, etc.—and this package adds even more by using the *CALL LINK* statement.

These extensions don't come without a price, however. Many of the statements available in Extended BASIC may *not* be used when in *DEP*'s 40-column mode. These statements are *INPUT*, *PRINT*, *ACCEPT*, *CALL COLOR*, *CALL SCREEN*, *CALL HCHAR*, *CALL VCHAR*, *CALL GCHAR*, *CALL CHAR*, *ON ERROR*, *TRACE*, and *BREAK*. The *DEP* provides equivalent statements for all of these commands except for the latter four. In addition, sprites

are not available because the Video Display Processor must be set to Text mode to use the 40-column capabilities of the computer. You can switch back into 28-column mode with a *CALL LINK* ("MODE28") at any time, whether in program or command mode, and all the normal commands and features are again available.

Here are some examples of added features:

CALL LINK ("MODE40")
sets the screen in 40-column mode.
CALL LINK ("MODE28")
returns the screen to 28-column mode.
CALL LINK ("SCRLDN")
scrolls the screen down one line.
CALL LINK ("LOCK", 10)
keeps the first ten lines on the screen from scrolling.

Limitations

In order to display numeric data with this program, it must first be converted to string data using the *STR\$* function. Also, the *ON ERROR* statement is not available in 40-column mode—only 28-column. Programming allowances must therefore be made to allow for poten-

tial operator-input errors (like nonexistent file names) while in 28-column mode.

ALWAYS SAVE YOUR PROGRAM PRIOR TO TESTING!!
Program crashes can be fatal if they occur when you're in

40-column mode. In many instances, if an error occurs, the screen turns strange colors, and the console may "lock-up." It is possible that you will need to turn the computer off and on, losing hours of programming effort. Another point worth mentioning is that *DEP*'s programs are not portable. That is, any program created with *DEP* must also run with *DEP*.

A Good Buy

Even with the above limitations, *DEP* is a good buy. For the game programmer, such a product would be a waste of money. But, for productivity applications, *DEP* is a real plus. This product can be a useful tool for unleashing the capabilities in the computer that you and I know is one of the best around. If you find yourself squeezed by a 28-column screen, Oak Tree Systems' *Display Enhancement Package* is your answer.

HCM

HCM Glossary terms: assembly language, display, sprite.

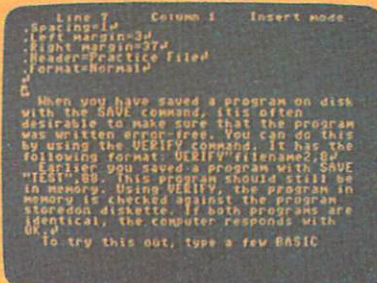
*"The 99/4A's 28 columns of text
is simply not a feature,
it's a handicap."*

SkiWriter II

A Review
by Dana M. Campbell
HCM Staff

If written printouts aren't enough to satisfy your urge to communicate, try sending your message over the wires as well.

HCM Review



Name: SkiWriter II
Program Type: Word Processor
Machines: Commodore 64 (IBM PCjr version being developed; release date unknown.)
Distributor: Prentice-Hall
P.O. Box 819
Englewood Cliffs, NJ 07632
Price: \$69.95
System Requirements:
TV or monitor, modem and/or printer, disk drive or cassette player.
Performance:

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

SkiWriter II is a fine little cartridge word processor for the Commodore 64. What makes it unique is that when you're done writing, you can send your text to any other computer connected to a modem and phone line. That's right—it also serves as a smart terminal.

SkiWriter II allows you to "converse" back and forth on the screen with your pals, send or receive documents, or hook up to information services like Compuserve. The SkiWriter II manual makes everything very easy to set up. And, even though it only discussed using a Commodore 1600 VICmodem or 1650 Automodem and communicating with other SkiWriter II systems, we tested it with the Hayes Stack Smartmodem and "talked" to an IBM PCjr, and it worked great. A handy on-screen timer is provided so that you can keep those phone bills down.

Getting Your Text Down

Along with the cartridge and the manual comes a cheap, paper keyboard overlay that, having been bent to fit in the package, refuses to lie flat. However, even in its crippled condition, it does help you remember the function of each editing key. You'll need the help, because this program has switched the key locations of some common Commodore editing commands, such as [DEL] and [-], and [RETURN] and F3.

Writing with SkiWriter II has been made simple and easy. To begin just push-in the cartridge, choose **Edit** from the main menu, and there you are: a nearly blank screen, with only a line and column counter on the top line. You are now free to begin committing your thoughts to the screen. When you type to the end of a line, your words automatically wrap down to the next line. You can worry about formatting your text later.

Text can be deleted by the character, word, or block, but once it's deleted it can't be retrieved—there is no "undo" or clipboard capability. You can Find and Replace words and phrases throughout your document, one replacement at a time—it won't work for the whole document. It's fairly easy to define a block to be deleted, moved, or copied, but you must remember to go back to the original block and remove the markers that were placed to define the block—that is, if you want the text to look clean. This is a bit of a hassle.

Indenting is also a prob-

lem. When you hit [RETURN], you simply end a paragraph and start in column 1 of the next line—it won't indent for you automatically. There is an option that allows you to specify hanging indents (where a paragraph's first line starts to the far left, and its following lines begin a few spaces to the right). However, it took a phone call to Prentice-Hall before I could get it to work. The manual's instructions on it are confusing and ambiguous.

By loading a file while another is already in memory, it is possible to merge several files together. The new file will be placed at the cursor's location in the first file. Without a "clipboard," this method of integrating various pieces of text by creating individual files of the pieces seems to work best.

Making It Look Good

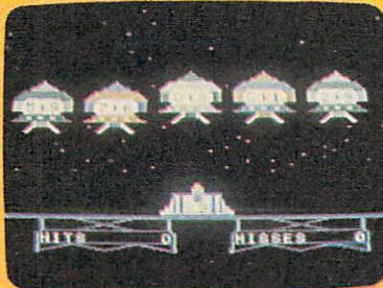
Something called the Dot Lines command takes you to the menu screen for formatting your text. It's called Dot Lines because a proper formatting command is preceded by a dot (period). You may specify the spacing between lines, left and right margins, page numbers, headers, comments, and centered or left- or right-side justification (alignment). You cannot insert tabs, which makes it difficult to construct charts. A preview screen allows you to view your document on the screen formatted in the manner that you have set up before you print it out.

No parameters are listed in the manual for any of the dot lines commands. Thus, we don't know such information as the maximum length for headers or margins, nor whether there is any way to set page numbers at the bottom of pages instead of at the top. The manual also failed to clearly explain the proper print option to choose for your printer. The company recommends choosing Commodore Printer first, and if that doesn't work try Parallel Printer, and then maybe Other Printer.

SkiWriter II is a good program for the novice word processor because it doesn't boggle the mind with a flood of commands—the program is simple and easy to learn. However, its limited number of options may hamper heavy-duty users, though its communications capability does give this program an extra appeal it might not otherwise garner. Overall, SkiWriter II seems to do exactly as it promises, and does it well.

HCM





Name: Alien Addition
 Program Type: Educational game
 Machine: TI-99/4A
 Distributor: For TI-99/4A
 Triton Products, TexComp,
 and Unisource catalogs.
 Price: \$5.95 to \$10.95 TI
 Cartridge
 System Requirements: TI-99/4A console, monitor
 Poor Fair Good Excellent
 Performance: _____
 Engrossment: _____
 Documentation: _____

Name: Alien Addition
 Program Type: Educational game
 Machines: Apple II Family, * C-64, *
 and IBM PC & PCjr*
 Distributor: For Apple II Family,
 * IBM PC & PCjr, * C-64*:
 Developmental Learning
 Materials
 DLM Teaching Resources
 1 DLM Park
 Allen, TX 75002
 Price: \$44 diskette
 *We were unable to obtain these newly-
 published versions in time for review.

ALIEN ADDITION

A Review by Tom Green
 HCM Staff

*Is the arcade-style edu-game out of date?
 Reaching for a futuristic theme, this program now
 seems a bit old-fashioned.*

To succeed in arcade game playing, you must draw on certain natural reserves, such as raw determination, the ability to concentrate, coordination, and quick-reflexes. Above all, you must pay attention. Naturally, educational programmers have realized that this same level of attention is also desirable in a learning situation—so many teaching games use arcade-type action to grab a youngster's attention, and create a focal point for learning. *Alien Addition* is a typical example of this popular method. Planting you smack in the middle of a futuristic nuclear war, it thoughtfully provides you with one defense—the right answers.

Alien Addition is one of six programs in a series of educational packages offered by Developmental Learning Materials designed to teach the basics of math—addition, subtraction, multiplication, and division. *Alien Addition* instructs the player on the summing of numbers between zero and nine. We tested the version available for the TI-99/4A. Newly-published versions for the Apple II Family, Commodore 64, and IBM PC and PCjr were not yet available to us in time for this review.

Time Is Of The Essence

The scenario of play is familiar—defender versus attackers. Five alien ships, each displaying an addition problem, are lined up across the top of the screen. Each problem consists of two numbers. The ships descend from left to right, one at a time. Positioned under the ships is a cursor-controlled "laser" cannon that displays the player's numeral input. The player must move the cannon right or left so that it is under the lowest ship, enter a number representing the answer to the ship-displayed problem, and "fire" away. A correct answer destroys the ship, and a new one replaces it at the top of the screen. An incorrect entry causes the aircraft to descend more quickly. If a ship descends to the last level, the cannon is vaporized into a mushroom cloud, and play starts anew.

lowest and highest scores.

Three options change the complexity of play: (1) the skill level may be set on a scale of one to nine, to control the speed of the ship's descent; (2) the range of whole numbers used for addition problems may be set to limits of 3, 6, or 9; and (3) the time of game play may range from one to five minutes.

Alien Addition's graphics make full use of the 99/4A's color palette—bright ships illuminate from a starry background, and red flashes signify a victorious shot. Sound enhancements are limited to a few beeps as the ships attack or as an object is destroyed.

Food For Thought

A few years ago, *Alien Addition* was considered innovative. But, since this and many other similar "edu-games" have been released, the arcade format has been shot down with its own ammo. One obvious objection you could raise against this game's format is that it uses a laser defense/nuclear war context to teach children mathematics. Another, less-obvious objection maintains that this concept has been used so often it is simply boring.

However, *Alien Addition* proves to be a fast and effective drill (and its low selling price for TI-users is an "additional" selling point). If you don't want to become so much atomic fluff, you'd better know the right answers. In reality, knowing the correct answers may avoid destruction (sometimes by causing destruction)—but this knowledge can also build, promote growth, and lead down other enlightening paths.

Newer educational programs are emerging which use the computer's "modeling" abilities to create computer simulations—software that mimics real-life situations, and helps answer "what-if" questions. Perhaps this new direction in software will stimulate the imaginations of some in the computer/teaching field to bring us more original educational games for students of all ages.

HCM

A Day At The Races:

A Review of Kwik-Load! And Mach 5

by Randy Thompson
HCM Staff



Get your disk drive up and running, and your C-64 up to speed!

It is a fact: At approximately 2.5k baud, the Commodore 1541 disk drive is the slowest disk drive in the computer industry. For anyone doing extensive work on the Commodore, this speed limitation can become quite an annoyance. Being an impatient person, I decided to look into some programs that promise to remedy the situation by speeding up the 1541's operation. What I came up with were two very competent pieces of software—*Kwik-Load!* and *Mach 5*.

Introducing The Drivers

Kwik-Load! comes on disk. It is a short machine-language program that is stored in memory at \$C000. Accompanying *Kwik-Load!* is another "Kwik" program entitled *Kwik-Copy!*.

Mach 5, on the other hand, comes on a cartridge. Accompanying *Mach 5* is a disk that contains some added utilities, which will be discussed later.

The Speed Limit

Only program loading speeds can be increased by *Kwik-Load!* and *Mach 5*—save operations are unaffected. Due to compatibility problems, speeding up the Commodore's save process would require modifying the 1541 itself. Also, because both of these programs rely on changing the load vector at \$330-\$331, the load time of most auto-boot or heavily copy-protected software will not be increased.

Gentlemen, Start Your Engines . . .

My first choice of action was to conduct a race. The race track: the tracks found on the disks of some of the Commodore 64's most popular programs. The results are shown in Figure 1.

The winner? Well, in every race but two, it was a photo finish. In the races that were not close, it was *Mach 5* all the way. *Flight Simulator II* loads in less than half its normal time when used with *Mach 5*, yet no difference is apparent when it is used with *Kwik-Load!*. While loading *Sysres* (a utility program reviewed in Vol. 5, No. 3), *Kwik-Load!* seems to drop its engine. It's true, *Mach 5* doesn't set any records while loading *Sysres*, but at least it finishes the race.

Although *Mach 5* claims to load software up to 500-percent faster, such performance is yet to be seen. I think "up to" are the operative words here. *Kwik-Load!* advertises a more meager, yet accurate figure of three to one. Both programs' speeds are determined mainly by the size and type of file being loaded. The larger the program, the larger the increase in speed. Also, machine-language programs do not seem to be affected by the speed increase as much as BASIC programs are.

And That's Not All

As if speeding up the 1541 disk drive isn't enough, both of these products come equipped with additional utilities.

Kwik-Load! comes with another misspelled program: *Kwik-Copy!*. *Kwik-Copy!* is actually two programs in one—a disk copier and a disk editor. I was more impressed with *Kwik-Copy!* than I was with the featured program, *Kwik-Load!*. With just one disk drive and only three swaps, *Kwik-Copy!* will backup a disk in four and one-half minutes. With two drives, a disk can be copied in a little under three minutes. If this seems like it takes a long time, try using the *Copy/All* program provided with the 1541 disk drive—now that takes a long time.

The disk-editing feature of *Kwik-Copy!* is very well-written and user friendly. If you've never used a disk-editing program, this is a good one to start with. Of course, any program that allows you to modify the bits and bytes of a disk is not intended for the complete novice. But for the more experienced user, such a program can be an invaluable tool.

Mach 5 comes with several built-in two-character commands for such things as displaying the disk directory or deleting a file. Besides DOS, other features are included too. To obtain a hard copy of the screen, simply enter a [-] H. A [-] R will disable drive rattle (the clatter the drive makes when it encounters an error). All of the commands added by *Mach 5* can be listed to the screen at any time by entering a [-] M.

Mach 5 is also accompanied by two programs that come on disk. One, called *Basic Boot*, gives you an extra 4K of memory for BASIC programming. How does it do this? Well, it moves BASIC ROM up from memory locations \$A000-\$BFFF into \$B000-\$CFFF. Of course,

FIGURE 1

Program Name	Load Time			Program Type
	Normal	Mach 5	Kwik-Load!	
Cities of Gold	1:20	1:20	1:20	ML
EasyScript	1:03	1:03	1:03	ML
Flight Sim. II	2:50	1:00	2:50	ML
Logo	2:00	0:45	0:45	ML
Outline Editor	1:04	0:14	0:14	HCM
Snap-Calc	0:33	0:09	0:09	HCM
Sysres	0:16	0:16	*crash*	ML
Kwik-Load!	0:06	0:03	N/A	ML

Time is in min/sec format

Program Type codes:

ML = commercial machine-language program
HCM = BASIC program from HCM



Name: Kwik-Load!
 Program: Disk utility
 Machines: Commodore 64
 Distributor: Datamost Inc.
 20660 Nordoff Street
 Chatsworth, CA 91311
 (818) 709-1202
 Price: \$19.95
 System Requirements: Commodore 64, 1541 disk drive.

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



Name: Mach 5
 Program: Cartridge-based disk utility
 Machine: Commodore 64
 Distributor: Access Software Inc.
 925 East 900 South
 Salt Lake City, Utah 84105
 Price: \$39.95
 System Requirements: Commodore 64, 1541 disk drive.

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

under such an arrangement, \$C000-\$CFFF is no longer open to machine-language programs. When you RUN *Basic Boot*, the fast-loading feature of *Mach 5* is disengaged. This can be remedied by entering SYS 57013. The SYS command does not, however, restore the extra commands that *Mach 5* usually provides.

Along with *Basic Boot*, *Mach 5* also contains a *Disk Organizer* program. For those of you who have trouble remembering which programs are on which disks, this is the program for you. *Disk Organizer* will read in the directories of several different disks, and save the information onto one data disk. Set up like a simple data base, this program can sort, search, and list the contents of your disks. You can make printouts containing certain file types or directories, and can even find the track and sector in which a program is located.

As with any program, *Disk Organizer* is not without its faults. First, the program's documentation manages to skip one very important detail. According to the manual, "it would be a good idea to do a sort." Take my word for it, in order to use the program's search feature you *must* have previously executed a sort—no ifs, ands, or buts. Failing to do so can cause great frustration. Second, *Disk Organizer* will not work without the *Mach 5* cartridge installed. I believe that this was done in order to discourage unauthorized distribution of the *Disk Organizer* program.

Compatibility

The most common problem with program utilities is compatibility. Sure, it may be a great utility, but if it doesn't work with the programs you happen to own and use, what good is it? With these two utilities, the most common form of incompatibility I encountered happened when I loaded copy-protected software. Most of the commercial programs that I tested would not "crash," they simply disconnected the fast loading features of both of these products. *Kwik-Load!* seemed the most prone to becoming disengaged.

Probably the most dramatic example of incompatibility that I found happened when I tried using *Sysres*. Apparently, *Kwik-Load!* and *Sysres* are not on speaking terms. When I tried to "kwik load" *Sysres*, the disk drive was thrown into a tail spin, while the computer sat and sulked—refusing any input other than the power switch.

Mach 5 and *Sysres* seem to have a more tolerant relationship. *Sysres* does not load any faster with *Mach 5*, but it does load. Once booted, *Sysres* adds its own DOS commands for disk access. When using these new commands to load programs, no speed increase is gained—

but, when using the conventional LOAD "FILENAME", 8 command, *Mach 5* can "do its thing." While *Mach 5* is plugged in, *Sysres* may bomb in several ways. For example, pressing [RUN/STOP] and [RESTORE] will almost always lock-up the computer. Also, using *Mach 5*'s new [↑] load feature can be dangerous. In fact, using any of *Mach 5*'s added commands is futile—they are all disconnected by *Sysres*.

Both *Kwik-Load!* and Commodore's DOS *Wedge* program are stored in the same location in memory, so they cannot be used together. This can be a major drawback for those who use the *Wedge* program frequently. *Mach 5* isn't perfect either. For example, never try to switch in character ROM from BASIC with *Mach 5* installed—it won't work!

Believe it or not, you can use *Kwik-Load!* in conjunction with *Mach 5*. In fact, *Kwik-Load!* loads almost twice as fast as usual with *Mach 5* installed. Which program is doing the actual loading after both programs are in memory, I don't know. I admit that using both of these utilities at once is not a very practical arrangement, but it is an interesting one nonetheless.

Conclusion

Speed-wise, both of these products largely improve the 1541's normal load time. As to which one is quicker, I would have to say *Mach 5*. True, most of the races were a tie, but it was *Mach 5* that never failed to reach the finish line. Also, the fact that *Kwik-Load!* itself has to be loaded, reduces its speed factor. With *Mach 5*, you simply plug it in and forget about it.

In the additional-programs category, *Kwik-Load!* gets my vote. The *Kwik-Copy!* program that is provided with *Kwik-Load!* is well worth the price of this product alone. *Mach 5*'s *Disk Organizer* is not without merit, but the practicality of such a utility has always escaped me. After owning several such programs, I have yet to use one. I found that organizing and updating all my disks took much more time than it was worth. The memory-expanding *Basic Boot* program is also not the most useful of utilities. With 38K of memory available for BASIC programming, running out of memory on the 64 is the least of my worries.

Overall, both *Kwik-Load!* and *Mach 5* are excellent buys for people wishing to increase the speed of their disk drive at a minimal cost. So, don't let your 1541 slow you down—put your disks into overdrive. See you at the races . . .

HCM

HCM Glossary Terms: auto-boot, baud, bit, byte, DOS, RAM, ROM, sector, track, vector.



Exiting Error Routines

For the proper operation of any program requiring a disk drive, error-checking is critical. Unfortunately, in order to use the **ON ERR GOTO** statement to handle errors in Applesoft BASIC you need some "inside" help.

Many different ways are available to handle possible program errors, but here we're concerned with getting back into your program after you've handled the error. You've got 3 different possibilities: (1) re-RUN the program, (2) **RESUME** execution with the same statement which caused the error, or (3) **GOTO** a selected place in the program, depending on what caused the error.

Alternative 1 is fine if you don't care whether data in memory is lost. Some educational programs and games can satisfactorily use such a procedure, but for productivity programs with large data bases, it is simply not acceptable. Using the **RESUME** command (alternative 2) is fine if you are sure that the error that caused the problem will be corrected before you **RESUME**, such as closing a disk drive door. But often only alternative 3 will do.

When you execute an **ON ERR GOTO** statement in an Applesoft program, the BASIC interpreter sets a flag at page zero location \$D8 by placing a byte greater than \$80 in this location. When an error occurs, Applesoft checks this location and branches to the line you've specified in your **ON ERR GOTO** statement. When an **ON ERR GOTO** statement is executed, the line number where the error occurred is placed on the stack along with some other information used by the **ON ERR GOTO** routine. If you exit your error routine using a **GOTO** statement instead of a **RESUME** command, this information will come back to haunt you. Applesoft uses the stack for several other purposes, such as addresses for **RETURNS** from **GOSUBs** and counters for **FOR-NEXT** loops. By a simple **CALL -3288** command you can clear all the information which would be used by the **RESUME** command off the stack. Then, you could **GOTO** the same part of the program that caused the error, but not the same line. For example, if a user inputs an invalid file name during a **SAVE** operation, you can use a **GOTO** back to the input-file-name routine, only if you clear the stack first by using the **CALL -3288**. (See the Apple version of "Mine Over Matter" in this issue beginning at line 2000 for an example of this technique.) Be sure, however, that you enter at the same level of **GOSUB**, or you will find your program behaving very strangely.

One last note: the **RESUME** command can cause a program to hang-up indefinitely (and subsequently bomb) if you are in the middle of a **FOR-NEXT** loop when the error is encountered, or if the error was caused by a **GET** statement. A short machine-language program from the "Apple BASIC Programming with ProDOS" manual (page 72) fixes this problem. Here's the BASIC loader program for that program:

```
1 FOR I=0 to 9:READ ZZ:POKE 768+I,ZZ:NEXT I
2 DATA 104,168,104,166,223,154,72,152,72,96
```

If you find that your program crashes after you try a **RESUME** command, insert this routine early in your program and then use a **CALL 768** from your error routine before you **RESUME**. This machine-language routine places one of the error-code bytes needed by **RESUME** back onto the stack in the proper place so that the **FOR-NEXT** and **GET** information will not be lost during the **RESUME** procedure:

—Roger Wood

HCM Glossary terms: BASIC loader, error routine, machine language, page zero.

TECH NOTES



Merging Programs From Tape

Probably the best utility a Commodore owner can have is a merge program: a utility that allows the user to take two separate programs and combine them into one. So far, two different merge routines have been published in "Home Computer Tech Notes" (see Vol. 5, No. 1, and Vol. 5, No. 3.) Both of these routines work fine—fine, that is, if you happen to own a disk drive. For those of you who own Datasets instead of disk drives, the following routine was written for you.

To use this routine, you must first type it in. Once keyed-in, **SAVE** it to tape (or disk). Be careful—this merge program erases itself from memory when **RUN**, so **SAVE** it first. Let's say that you want to merge a program called MERGEFILE with one called MAINFILE. To do this, you must first **LIST** the program MERGEFILE as a **SE**quential text file onto tape. This is done by loading MERGEFILE into the computer and entering the following:

```
OPEN 1,1,1,"TEXTFILENAME":CMD1:LIST:PRINT#1
```

Then press **[RETURN]**. Now, when the computer prompts you, press **[RECORD]** and **[PLAY]** on the cassette. The program will now be saved as a text file. When the cursor returns, enter **CLOSE 1** to close the text file.

The actual TEXTFILENAME can be replaced with the name of your choice. Now, **LOAD** and **RUN** the merge program given here. Once **RUN**, a short set of instructions will appear. You should now **LOAD** the MAINFILE into the computer. When the program has been loaded, insert the tape with TEXTFILENAME, make sure that it is rewound properly, and enter:

```
SYS 49152,"TEXTFILENAME"
```

Again, the TEXTFILENAME should be whatever you named your program when it was saved as a text file. Once you press **[RETURN]**, the computer will merge your two programs. When the tape stops and the cursor returns, the two programs will have become one. If two programs that are being merged contain identical line numbers, the lines in memory will be replaced by the ones on tape.

Programs that are **LIST**ed as **SE**quential text files are stored onto tape (or disk) much differently than if they had been **SAVED**. **PRo**gram files are stored in a compact format: each keyword or function—such as **PRINT** or **PEEK**—is tokenized into one character. A **LIST**ed file saves every character separately as if it had been typed-in. The merge program presented here takes advantage of this by fooling the computer into thinking that the program lines being read-in from the text file are actually being typed-in. Unlike many merge routines that use this same method, the program lines that are being merged are never listed to the screen. Instead, they are placed directly into the input buffer and then merged into program memory.

One thing that you may notice when you use this program is that the screen display will scroll up during a merge. This is because the computer is responding as if someone had actually pressed **[RETURN]** for each program line that was entered. The longer the program being merged, the more times the screen will scroll.

—Randy Thompson

HCM Glossary terms: input buffer, program file, sequential text file, tokenize.

For your key-in listing, see HCM PROGRAM LISTINGS Contents

Tape-Merge Explanation of the Program

Line Nos.	
100-180	Program header.
190-200	READ and POKE merge routine.
210-280	Print instructions on screen.
290-490	DATA containing machine language.

HOME COMPUTER



Using Special Character Graphics



How many times have you searched through the back of the BASIC manual looking for a special character required for your program? Quite often, you may find one within the IBM's large character set that is suitable for your needs; however, there are circumstances that may require you to design your own custom characters.

One drawback to this on the IBM PC is that your Color/Graphics interface card may not have access to the entire character set in medium- or high-resolution graphics modes (screen modes 1 or 2). The PCjr, however, always has full access to these characters, no matter which mode you use. Both PC and PCjr users may at times also need a special character that is not supplied in the standard character set, or face a situation when the desired character from the built-in character set is inaccessible. The routine listed here may help solve this dilemma for you. This routine gives you a method of displaying special characters on any of the graphics screens (it will not work with the text screen), as well as a couple of examples of how to use these characters. You will still need to design your own characters if you wish to use characters other than the ones shown in the examples.

One method we could have used involves placing **DRAW** commands into a string array, with each element indexing a different character. We could then substitute the **DRAW** command for the **PUT** command in line 300. This method works best when you have a large number of special characters, such as an entirely new character set. The execution time for this method, however, is comparatively slow.

The method used in this program involves **DRAW**ing each character and saving its pattern in an integer array with the **GET** command. The **PUT** command can then be used to place the character anywhere on the screen. This is much faster than using the **DRAW** command to place the characters on the screen. This method is efficient when there are only a few special characters required. As you can see from the listing, a short one-line subroutine is required for each character because the **PUT** command must use an array with only one dimension. Multi-dimensional arrays that might be able to contain several shapes are not allowed. This means

we must have a separate **PUT** statement for each shape's array.

You are not limited to a normal single-character size when you design your own characters. If you have a special need for a character that is, in effect, several characters long, don't worry—you will only need to make a slight modification to the routines in lines 300 or 310. Line 300 places a character for the symbol **M/S** (meters per second) on the screen. This symbol is squeezed into the space of one character. The second routine in line 310, however, places on the screen a character that is two characters wide. This is the symbol commonly used to indicate the **[RETURN]** or **[ENTER]** key.

Notice the differences between the two lines. The **PRINT** statement moves the cursor two places to the

```

100  * * * * *
110  * CHARACTER GRAPHICS *
120  * * * * *
130  COPYRIGHT 1985
140  EMERALD VALLEY PUBLISHING CO.
150  BY WILLIAM K. BALTHROP
160  HOME COMPUTER MAGAZINE
170  VERSION 5.4.1
180  IBM PCjr WITH CARTRIDGE BASIC or
190  IBM PC WITH BASICA and
200  COLOR/GRAPHICS ADAPTER and
210  COLOR MONITOR
220  DEFINE GRAPHICS CHARACTERS
230  SCREEN 1:COLOR 0,0:DEFINT G:DIM GEN
    T(18),GMS(10)
240  CLS:DRAW "BM100,100BD4E3DG2DE2DGDUFU
    2ER6U3LD2LU2":GET (100,100)-(115,10
    7),GENT
250  CLS:DRAW "BM100,100ND3F2E2D3NG4NE3B
    RNR2DF2GL":GET (100,100)-(107,107),
    GMS
260  USE GRAPHICS CHARACTERS WITH TEXT
270  CLS:LOCATE 12,1:PRINT "PRESS ";:GOS
    UB 310:PRINT "TO CONTINUE";
280  LOCATE 16,1:PRINT "THE CAR CAN TRAV
    EL AT 30":GOSUB 300:LOCATE 22,1:EN
    D
290  ROUTINES TO PLACE CHARACTER
300  PRINT " ";:PUT (8*(POS(XP)-2),8*(CS
    RLIN-1)),GMS,PSET:RETURN
310  PRINT " ";:PUT (8*(POS(1)-3),8*(CS
    RLIN-1)),GENT,PSET:RETURN
    
```

right in the second example, and only one place to the right in the first. The offset used with the **POS** function is increased by one in line 310 to reflect the longer character size. You may wish to experiment with your own values to determine the effects you require.

—William K. Balthrop

HCM Glossary terms: string array, integer array, multi-dimensional array.

TECH NOTES



Making Your Own Tex-Sette™ Adapter

Does your TI-99/4A fail to control your cassette recorder through the remote jack? One possible solution for this is not difficult at all. If you can solder two wires together, then it is likely that you can solve this problem yourself.

The remote plug on the 99/4A cassette interface cable contains two wires—the ground and the lead. The lead is located on the tip of the plug, while the ground is the shaft. It is possible for the remote jack on a recorder to have these two wires switched—thus making it incompatible with the cassette-interface cable. [Note: In very rare cases, the recorder remote signal circuit may be incompatible—Ed.] Fixing this is simply a matter of switching the two wires so that the recorder's ground is the same as the cable's ground. So that we don't have to modify the recorder itself, we've come up with a gadget called the Tex-Sette adapter. Here's how to build your own:

First, you'll need to collect everything that is shown in the parts list. Two holes will have to be made in the enclosure—one for the jack, and the other for the plug. Solder one end of your two-conductor wire to the jack. Mark the strand that is soldered to the lead connection of the plug. Then take the marked wire and solder it to the ground connector of the jack. The ground on a jack is the outside rim. Now solder the remaining wire to the jack's lead connector. Finally, carefully slip all the wires into the enclosure, making sure that no bare connections are touching, and close it all up. Your finished product should look somewhat like the picture below.

Testing your adapter is fairly simple. Using an ohm meter, or some other kind of continuity tester, check the connection between the tip of the plug and the rim of the jack. If they are connected, you have correctly built the adapter. If not, carefully read the instructions again to see what went wrong.

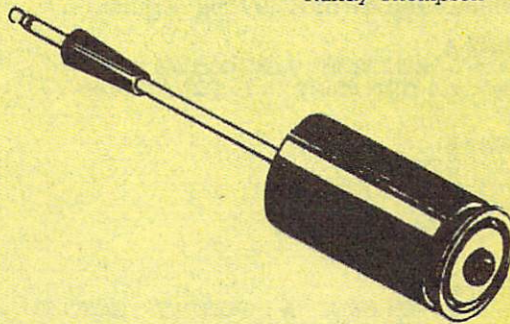
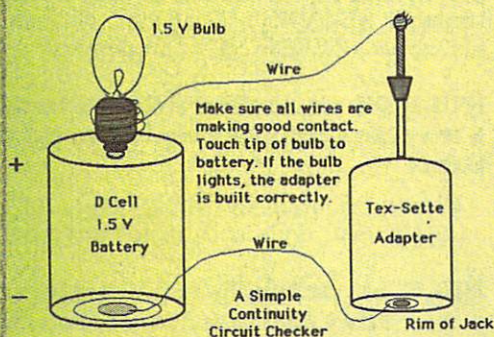
To use your latest creation, insert the remote control plug (the black one) of the 99/4A cassette interface cable into the Tex-Sette's jack, and put the plug from the Tex-Sette adapter into the cassette recorder's remote jack. The remote feature on your recorder should now be working in conjunction with your TI-99/4A. Besides making cassette operation more convenient and "foolproof," you will now be able to load data files under program control—expanding the usefulness of your home computer.

PARTS LIST

- | | |
|--|--|
| 1 3/32" submini phone plug | Radio Shack Part 274-289 or equivalent |
| 1 3/32" submini phone jack | Radio Shack Part 274-292 or equivalent |
| Enclosure—35mm film canister, or similar container | |
| 5" two-conductor wire | |

[Home Computer Magazine, the publisher, and the author shall not be held liable for unsuccessful project completion. Proceed at your own risk.]

—Randy Thompson



PART TWO

by John P. Russo
and the HCM Staff

Adding snap to BASIC requires strategy and efficient coding. Here are some snappy new ways to RUN your program in the fast lane . . .

Speeding Up A BASIC Program

Have you grown impatient with slow-running BASIC programs? Perhaps you've imagined pouring a bit of jet fuel into your computer to get it moving. Relax! Before you do something that drastic, remember: Some programs run much faster than others. Even though interpreted BASIC is inherently slower than compiled languages, an efficient use of code will often greatly enhance the running speed of any program.

In Part 1 of this 2-part series (see Vol. 5, No. 3), we gave you 8 rules for speeding up BASIC. We also used benchmarking as a tool for comparing the "before and after" effects of certain rules. Now, we continue with 6 more speed-enhancing guidelines. These rules apply to almost any version of BASIC. However, the examples discussed here deal with Apple II (Applesoft) BASIC, IBM PC DOS 2.0 BASIC (Disk), Commodore 64 BASIC, and TI-99/4A Extended BASIC. We chose TI Extended BASIC to maintain maximum compatibility between the different BASICs.

RULE 9 shows how to use conditional (IF-THEN) statements efficiently. Many versions of BASIC (including Applesoft and Commodore) lack IF-THEN-ELSE statements, so programmers improvise in various ways to approximate these commands. Suppose, for example, that we wish to test a number, A, to see whether it is positive and then print a message about the result. If we could, we would use:

```
IF A > 0 THEN PRINT "A IS POSITIVE"  
ELSE PRINT "A IS NOT POSITIVE"
```

In fact, the line above is a legal statement in IBM BASIC and TI Extended BASIC. However, if an IF-THEN-ELSE statement is not available, we probably would write one of the three following segments:

```
SEGMENT A  
10 IF A > 0 THEN PRINT "A IS POSITIVE"  
20 IF A <= 0 THEN PRINT "A IS NOT POSITIVE"  
30
```

```
SEGMENT B  
10 IF A > 0 THEN 40  
20 PRINT "A IS NOT POSITIVE"  
30 GOTO 50  
40 PRINT "A IS POSITIVE"  
50
```

```
SEGMENT C  
10 IF A > 0 THEN PRINT "A IS POSITIVE": GOTO 30  
20 PRINT "A IS NOT POSITIVE"  
30
```

Normally, I prefer SEGMENT A, because it's the easiest to read. Unfortunately, it is also the slowest of the three segments. SEGMENT B executes more rapidly, but is also more difficult to read. SEGMENT C is more readable than SEGMENT B, because it does less

"hop-scotching." It turns out that SEGMENT C is also the quickest of the three segments. The built-in IF-THEN-ELSE statement available in IBM BASIC and TI Extended BASIC is the quickest for those systems; but on Apple and Commodore computers, RULE 9 provides reasonable savings.

RULE 9: The fastest way of implementing the statement:

```
IF CONDITION THEN STATEMENT 1 ELSE STATEMENT 2  
is
```

```
IF CONDITION THEN STATEMENT 1: GOTO XX STATEMENT 2  
where XX is the next line number.
```

The situation is somewhat more complex if there are more than two possible conditional outcomes. Suppose, for example, that we wish to optimize a section of code which keeps track of the number of positive, negative, or zero numbers. Consider the next segment:

```
SEGMENT D  
10 IF A = 0 THEN ZEROS = ZEROS + 1: GOTO 40  
20 IF A < 0 THEN NEG = NEG + 1: GOTO 40  
30 POS = POS + 1  
40
```

Now, if it is known that most of the numbers will be positive and that very few will be zero, it would be better to reorder SEGMENT D to obtain:

```
SEGMENT E  
10 IF A > 0 THEN POS = POS + 1: GOTO 40  
20 IF A < 0 THEN NEG = NEG + 1: GOTO 40  
30 ZEROS = ZEROS + 1  
40
```

In SEGMENT E, very little time will be wasted in applying tests which have no chance of success. In an actual RUN, SEGMENT E will RUN as much as 25 percent faster than SEGMENT D. In general, we can state:

RULE 10: Order IF-THEN statements in such a way that those likely to be successful come first.

Closely related to RULE 10, and unfortunately sometimes at odds with it, is the following rule:

RULE 11: Order IF-THEN statements in such a way that the least time-consuming tests come first.

As an example of an application of RULE 11, suppose we are writing a program which tallies the number of "illegal" strings in a long list. For this example, let's suppose that a string will be deemed illegal if it fails either TEST A or TEST B, defined as follows:



TEST A: The string only contains upper-case alphabetical letters.

TEST B: The length of the string is less than 9 characters.

Note that TEST A will require a loop and the MID\$ function, while TEST B will require only the very fast LEN function. All other things being equal, it makes more sense to try TEST B first. If TEST B fails, there is no need to apply the very time-consuming TEST A.

To set the stage for our next rule, assume that you've been fine-tuning a program and have determined that most of the execution time cost is connected with a certain loop. Within that loop, a variable is subjected to a number of tests. For an easy way to improve the speed of the loop, merely initialize the variable very early in the program. This early initialization will cause the variable to be entered at the beginning of the symbol table which is set up during program execution. Each time a variable is referenced, this table is searched, and entries located near the beginning will be found more quickly.

As an example of how well this can improve performance, consider the next program segment, which gets single characters from some source, say, a file, and counts the number of vowels and consonants. For this illustration, assume that ten string variables are used in lines 10-90 (thus introducing ten entries into the variable table). By adding 5 LET A\$ = "A" and thus placing A\$ first in the symbol table, as much as two seconds can be shaved off the execution time of the loop. This is a good place to recall that RULE 3 suggested that NEXT should be used with FOR loops instead of NEXT N. If NEXT N is used and N is located at the end of a long variable table, the search for N can waste considerably more time. (TI users note: RULE 3 does not apply to TI BASICs.)

```
10 REM ASSUME THAT 10 VARIABLES ARE INTRODUCED IN
    LINES 10-90
```

```
...
```

```
100 FOR N = 1 TO 1000
110 REM ASSUME A$ IS OBTAINED IN SOME WAY BY THIS LINE
120 IF A$ = "A" OR A$ = "E" OR A$ = "I" OR A$ = "O"
    OR A$ = "U" THEN VOWELCOUNT = VOWELCOUNT + 1:
    GOTO 140
130 CONSCOUNT = CONSCOUNT + 1
140 NEXT
```

To summarize:

RULE 12: The most frequently referenced variables should be initialized early in the program.

It is inevitable that a long program will require the use of subroutines. If the same lines are used several times in a program, then it makes sense to have the lines included only once as a subroutine. The next rule deals with the physical location of subroutines.

RULE 13: Always give subroutines line numbers that will be searched for first.

This rule's application varies somewhat from system to system. When the Apple or IBM interpreter encounters a command such as GOSUB 1000, the line numbers in the program are scanned—starting with the lowest in the program. If the program must check many line numbers before reaching the desired number, then execution time will be increased. This means that a frequently called

subroutine placed at the end of a very long program could require a significant amount of extra time.

The TI-99/4A, however, uses the opposite rule: The most frequently used subroutines should have the *highest* line numbers—because the TI machine starts its search from the last line number and works backward.

Commodore computers use a different method. They conduct a *logical* search for subroutines. Depending on the subroutine's line number, a search may start from the beginning of a program, or from the location of the current program line. Thus, there are two good places to put your subroutines—the beginning of a program or right after the GOSUB statement.

Our final rule also involves subroutines. Clearly, a subroutine call involves some overhead, since at least two jumps must be made—one to the subroutine and the other back to the main program. If one is willing to give up the space savings of a subroutine, and perhaps lose some readability, then putting the subroutine "in-line" can be an effective way to increase execution speed.

RULE 14: Time can be saved by eliminating the overhead of a subroutine call and replacing the call with in-line code.

As a simple example, compare the execution times of the program segments below. (Each segment checks the first character of A\$ and increments a counter if the character is equal to Z.)

BEFORE RULE 14:

```
10 FOR N = 1 TO 1000
15 REM ASSUME THAT A$ IS OBTAINED BY THIS LINE
20 GOSUB 50
30 NEXT
40 END
50 IF LEFT$(A$, 1) = "Z" THEN ZCOUNT = ZCOUNT + 1
60 RETURN
```

AFTER RULE 14:

```
10 FOR N = 1 TO 1000
15 REM ASSUME THAT A$ IS OBTAINED BY THIS
20 IF LEFT$(A$, 1) = "Z" THEN ZCOUNT = ZCOUNT
```

FIGURE 1

Execution Times Before and After RULE 14

	Apple II	Commodore	IBM	TI-EX
Before	7.4	5.8	7.4	34.93
After	5.8	4.8	6.1	32.60

Summing Up

Although we have focused on speeding up existing programs, it is clear that the rules above can help if you keep them in mind when constructing new programs. A "good" program usually has to strike a balance between efficiency and structure, but a number of the ideas given here do not involve space-time tradeoffs or compromises in good program structure.

The rules presented here do not, by any means, provide the only methods available of increasing execution speed. In particular, we have not dealt with I/O considerations, largely because they are usually so machine-dependent. However, we have covered most of the efficiency rules which are reasonably broad. Additional rules which are less general or perhaps even machine-specific might be found by carefully reading your computer's reference manual.

HCM

HCM Glossary terms: benchmarking, compiled language, initialization, interpreted language, string.

HOME COMPUTER

VOL. V NO. 4 ★ ★ ★ ★

INTERNATIONAL EDITION

Mergers Multiply

Tight Industry Spawns Co-Ventures and Coups

Blue Chips Scrutinize Apple, Lotus Reels-In VisiCalc Makers

Are mergers made in heaven? Or are they spawned in desperation? In the stagnant waters of today's computer industry, the truth is that companies are coming together out of a sheer sense of survival.

Case in Point: Apple Computer Corp. may soon announce several joint ventures with other companies, including corporate monolith AT&T. These moves reflect Apple's concern that if it doesn't play hardball in the big business world, there will be no minor league to return to. Slow home sales may mandate success in the business field to survive.

Speculation about the new Apple/AT&T combination focuses on the Macintosh as office machine. Rumor has it that AT&T will begin marketing the Mac through its nationwide chain of phone centers, perhaps bundled with a phone. Future Mac models may even include a built-in phone—turning the Mac into a sophisticated voice/data workstation. Others, including General Electric and Xerox, are also rumored to be interested in acquiring Big Red—the ultimate joint venture.

Several other upstarts have also felt the pinch of the present market; some are even fleeing Chapter 11 by merging with long-time rivals. This includes Software Arts, creator of VisiCalc, who is now letting itself be absorbed by Lotus Corp., of 1-2-3 fame. Besides this outright takeover, Lotus has joined with Intel—bundling Intel's Above Board (a memory expander for the IBM PC that provides up to 4 megabytes of RAM) with an updated version of 1-2-3. More co-ventures include the acquisition of Software Distribution Services by Ingram Corp., a video and book distributor; a deal between Xidex and Dysan, floppy disk suppliers to Kodak; and an alliance between Borland and AST Research Inc., bundling Borland's Sidekick with AST's multifunction board for the IBM PC.

“QUOTABLES”

**“You'll have to convince me
that the voracious little s.o.b.
won't eat my copy.”**

—ABC Anchorman Ted Koppel,
a typewriter advocate, referring to computers.

What's News—

Texas Instruments representatives said they preferred to use TI's marketing resources to appear at more than 100 “vertical market” tradeshow this year, rather than to attend COMDEX and the National Computer Conference this summer.

Later this year TV manufacturer Zenith plans to introduce a TV that uses the digital technology of computers. Viewers may eventually be able to watch 2 channels at once, as well as use their TV as an intelligent terminal.

The business market may take another look at Apple's Macintosh when MacCharlie begins shipping. MacCharlie will make the Mac fully IBM-compatible, allowing it to run 10,000 more programs and perform all networking capabilities of the IBM PC.

Though Japan's MSX computers still have not hit the U.S. in force as predicted, 15 Japanese firms are working on Version II models. The Japanese may enter the U.S. low-end market if Atari and Commodore seek the high end with their new computers.

Panasonic's new photocopier system allows copies to be edited electronically without altering the original. In seconds, information and graphics may be moved, centered, deleted, enlarged, or reduced.

After 10 years of seeing the world run with its inventions, Xerox has introduced a new batch of office products that include 3 personal computers, 3 laser printers, 2 word processors, and network applications. Xerox's 4,000 member direct-sales force will be touting “solutions for specific jobs.”

Mac Battleground

Apple Strategy Hinges On Third-Party Development

Chairman Jobs Goads On Lotus and Microsoft

Swirling around Apple's Macintosh-led journey into the business market is a pitched battle between rivals for Mac's software crown. Both Microsoft and Lotus are rushing to complete their integrated office packages for the Macintosh, Apple's featured product.

Apple is anxiously awaiting Microsoft's Excel and Lotus' Jazz, hoping that one or both will ensure Mac's place in the PC-dominated office market. Chairman Jobs may even be playing one against the other. Both Microsoft and Lotus, however, are behind schedule—even though an early summer release seems essential in capturing a respectable share of this growing market before IBM grabs the spotlight with its soon-to-be-introduced PC2.

After a slow quarter, resulting in plant closures and threats of more lay-offs, Apple's Mac strategy may be a case of do-or-die. The company recently announced the demise of its Macintosh XL—formerly, the Lisa. And, with the inevitable shift away from the Apple II series (and the home market in general) everything seems to depend on third-party development of businessware for the Mac.

In The Wake of Junior

Big Blue Concentrates On Its XT and PC2 Computers

Publishers Puzzled Over Bundling, PC Future

Software publishers don't seem to be too worried over the death of IBM's PCjr, and are releasing a substantial number of new programs—caught in the pipeline—this quarter. Although publishers aren't seeking new PCjr products, they are continuing to produce and distribute current PCjr products. Considering that there are 300,000 PCjrs already sold and possibly 200,000 warehoused, and that much of this inventory is compatible also with the PC, publishers are optimistic about the future—especially since IBM has recently cut the retail price of the PCjr by 27%.

Meanwhile, IBM has launched a new promotion which “bundles” business software with their PC XT in shipments to dealers. Even though IBM calls it “business as usual,” retailers aren't so sure. Some think it's an attempt by IBM to dump existing PCs, making room for their expected new PC2 while lowering prices for existing PCs. IBM isn't advertising this promotion, leaving it up to its dealers to determine whether to pass-on the free programs as a buying incentive or sell them for added profit.

INDUSTRY JOURNALTM

© COPYRIGHT 1985 EMERALD VALLEY PUBLISHING CO.

EUGENE, OREGON

NO CENTS

Commodore Sales Drop

Publishers Ignore C-64 For Apple, IBM Aftermarket

Is The C-128 Only A Stopgap For Coming Amiga?

Software sales for the Commodore 64 are dwindling as the machine ages and as software makers continue to narrow their focus to programs for Apple and IBM machines. To increase their sales volume and overall profits, Electronic Arts, CBS, and other companies have cut 15 to 40% off prices on older Commodore (and Atari) programs, and have cut dealer costs on newer titles about 16%. A boost in educational program sales may come from Commodore's efforts to capitalize on the school market with its C-64 and new 128 computers. The company has doubled the size of its educational marketing division and is going after volume local and state school contracts.

The Commodore 128, with C-64 and CP/M compatibility, was to start shipping in May. However, some market analysts have said that it may be a short-lived machine, released as "a stop-gap" product until Commodore can release its much-talked-about Amiga computer this summer. Another short-lived product, the Commodore Executive portable computer with 64K and 5-inch screen, is being advertised for under \$400 by a national liquidator.

New Atari Maneuvers

ST Delayed Once More Until Mid-Summer Heat

European Release Precedes American Debut

Atari's new 512K Macintosh-like ST, originally scheduled for release in April, will not be sold in the U.S. until July, according to Atari officials. The company has recently reversed several other earlier decisions regarding the ST computer. For instance, it is now shipping the ST to Europe to capture that market first before trying the U.S.. Atari also pulled out of the June Consumer Electronics Show, and has decided to sell the machine through computer specialty stores (where it will directly compete with the Macintosh) rather than through mass merchandisers.

IBM Printer Wars

Big Blue Goes Up Against Dominant Japanese Units

Epson and NEC Lower Prices To Counter IBM Move

IBM may crack the hold that Japanese printer makers have on the U.S. market by releasing printers of its own. The first is a \$549, high-speed (200 cps) graphics printer. It replaces the slower Epson printer that IBM sold under its own name, and is expected to add fire to the "printer wars." Meanwhile, printer market leaders Epson and NEC of Japan cut up to 20% off the prices of their printers, and plan to release ink-jet and laser printers this year. Adding fuel to the fire is Hewlett-Packard, which released a laser printer with many of the capabilities of Apple's LaserWriter—for half its price.

Tandy Price Cuts

Aggressive Plan Puts Computers In Low Range

Company Claims Model 1000 Better than PC, PCjr

Tandy's inflexibility on prices became history when it cut the price of its IBM XT-compatible Model 1200 computer 33% in April. Tandy chairman John Roach has also hinted that the Model 1000's \$1,199 price may be cut below the \$1,000 mark this year. Industry analysts predict that the 128K machine's price will drop 25%. But aggressive pricing is only one of Tandy's strategies for gaining market share for the 1000. Proclaiming that the Tandy 1000 is what the PCjr and PC should have been, Roach has said that it may start selling the 1000 through its Home Education System—called by others, "Tandy Tupperware."

THROUGH THE LOOKING GLASS

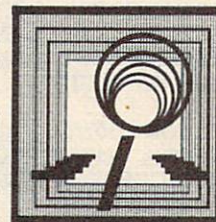
Chips With Built-in Sensors May Lead To More Human-Like Machines and Computers

A new factor has entered the cybernetic scene: the "smart" sensor. This latest technological marvel is designed to give machines the ability to see, hear, touch, even smell the environment and react appropriately.

One of the first of these devices is a small, saucer-shaped apparatus designed to sniff out deadly gases. What makes it different from older industrial sensors—which are much like simple thermostats hooked to expensive computers—is its ability to discern within the device itself one stimulus (in this case, a specific gas) among many, and then to initiate the right series of responses. This is possible because the device combines a computer chip containing the necessary "firmware" with the detector itself. Although most of the attention is now directed at its industrial and military applications, the smart sensor will inevitably show up in the home environments.

Some of the immediate uses for these sensors include blood testing, monitoring toxic substances (both in and out of the body), even detecting diseases, such as cancer. We may soon see small micromachines that can monitor blood-sugar levels in diabetics and internally administer insulin. Dentists are talking about toothbrushes that detect tooth decay and gum disease. Car-makers are dreaming of very automatic assembly lines. But the real impact may be far more sweeping—changing the very nature of machines and their relationship to humankind.

Smart sensors may finally bring many science-fiction dreams to life, including humanoid robots and the ultimate user-friendly computer. Computers, for example, now depend mostly on keyboard input to detect our wants and desires. But many other forms of interaction are possible, as millions of the so-called handicapped are beginning to discover. And the new microchip-sensing devices can only help advance this quest for a more adaptable and human computer interface. Star Wars applications aside, this technology could do much to enhance and protect life right at home.



Algorithm-A- Invisible Ripples

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. A flow chart is a handy tool 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 in each issue.

This issue's featured algorithm solves a problem encountered while writing the game/simulation program, *Mine Over Matter*. The problem was to set four different parameter values for each screen location: (1) ore quality, (2) ore depth, (3) the environmental impact factor of the site, and (4) the quantity of ore. These values had to be random enough to make the game challenging, yet predictable enough to involve skill.

In *Home Computer Magazine* Vol. 5, No. 2, the program *Ripples* in "IBMpressions" created a 3-dimensional surface drawing, like concentric ripples in a pond. A cross-section of this pattern forms a sine wave. This algorithm proved perfect for our purpose—we used a similar equation, but made the ripples *invisible*, representing changing values instead of physical waves.

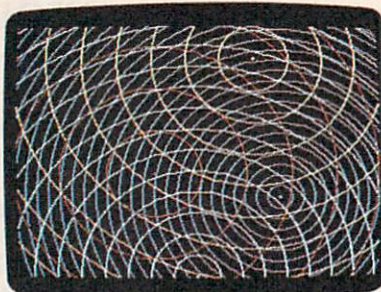
We created 4 different circles (with random center locations) and selected values based on the ripples of a sine wave emanating from those centers. To further complicate things, each ripple pattern was given a different frequency. The photo here is a graphic representation of the *Mine Over Matter* screen with each parameter shown as a different-colored circle.

```
200 DIM CEN(4,2), V(4)
210 FOR COUNT=1 TO 4
220 CEN(COUNT,1)=INT(RND*SX)+1
230 CEN(COUNT,2)=INT(RND*SY)+1
240 NEXT COUNT
...
500 FOR COUNT=1 TO 4
510 DIS= SQR((CEN(COUNT,1)-X)^2+(CEN(COUNT,2)-Y)^2)
520 V(COUNT)=INT(SIN(COUNT*0.4*DIS)*50+50)
530 NEXT COUNT
540 RETURN
```

In the above program, the CEN(,) array indexes the screen coordinates for the 4 centers of the ripples—e.g., CEN(3,1),CEN(3,2) would be the X,Y coordinates for the center of the third parameter. These centers are randomly selected in lines 210 through 240. SX is set to the character width of your screen (or the character width of the playing area), and SY to the number of character lines in the playing area. The array V() is used to contain the 4 parameter values for a particular screen location X,Y.

The heart of this algorithm consists of the equations in lines 510 & 520. First, we needed to find the DISTANCE

Tricks

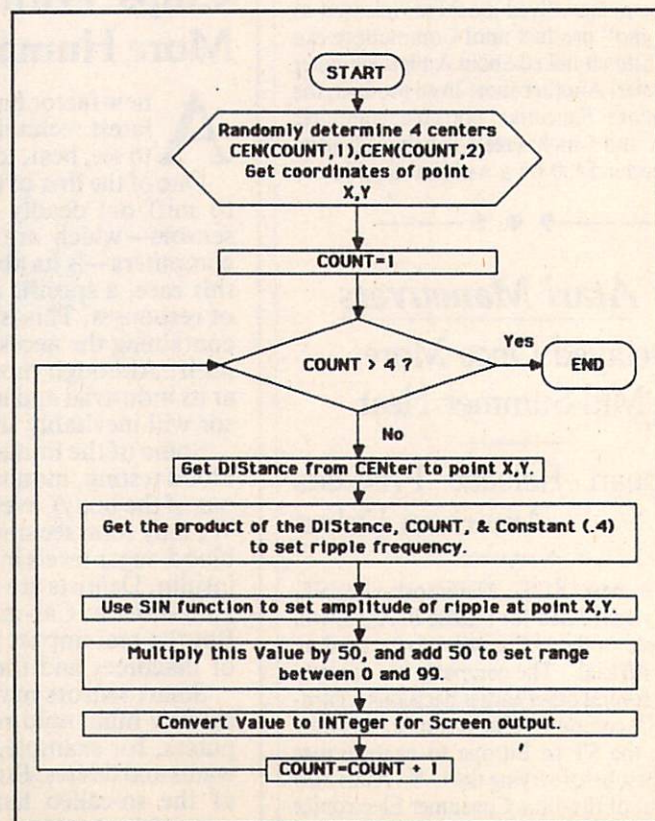


from a ripple's center to the screen position being tested. This was done using the distance formula based on the Pythagorean theorem:

$$DIS = \text{SQR} ((X' - X)^2 + (Y' - Y)^2)$$

In our example above, CEN(COUNT,1) is X', and CEN(COUNT,2) is Y', and X,Y is the screen location. To give each ripple a different frequency, we first multiplied together a constant (0.4), and the COUNTER (0.4*COUNT). The SIN of this value times the DISTANCE gives us the relative parameter value for that screen location. Because the SIN of a value is between -.99 and +.99, we set the range between 0 and 99 by multiplying by 50 and adding 50.

The following flow chart illustrates this algorithm.



HCM

HCM Glossary terms: algorithm, flow chart, parameter, counter, constant.



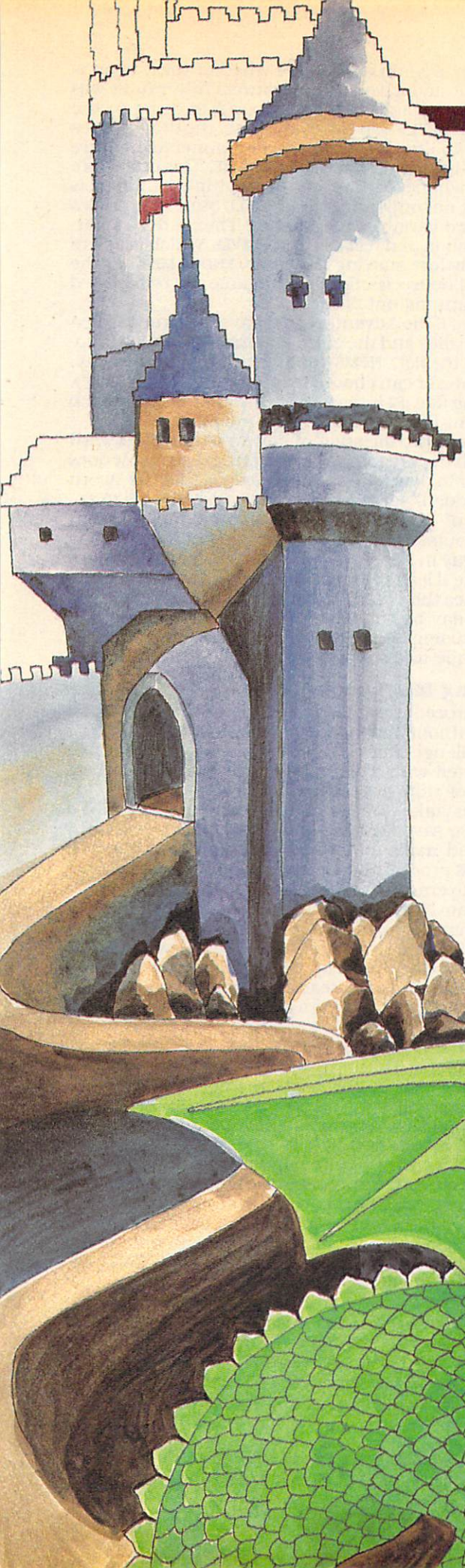
Build a LOGO Adventure

Part 4 of 4
by Andrew Keith

and the HCM Staff

In this final chapter, our Adventure reaches its completion phase. Now, the question is not what procedure to write but, "Can you save the realm?"

In the first three parts of this adventure-building tutorial, we showed how to sort all keyboard input into logical pieces that can be handled by LOGO procedures. We also developed a general scenario, including a map of "places," objects, and procedures for manipulating the objects and moving around. Once these procedures for handling sentence analysis, movement, and object manipulation are in mind, it is possible to construct many different LOGO Adventures. This last installment describes the hardest (and probably the most enjoyable) part of writing an Adventure Game, whether it's in LOGO or any other computer language: developing a story. The process by which a story is developed is intangible, and like any creative process, highly individual—there is no set formula. The story can evolve out of a particular theme and setting, as did the Adventure we present here. Or, you could just as easily start with a set of characters and situations and then decide how they will all interact. Likewise, you can begin by defining the ultimate goal and then develop the story by working backwards from that ending. How you come up with the final story is up to you. What we can demonstrate are the tools for making the story work; and so we will focus largely on how to implement these tools.



Filling In the Details

Here are some details that must be resolved when creating your scenario: How does the Adventurer win? What obstacles must be overcome and how? What are the alternate endings, and at what points can they occur? Which objects are used? What characters are encountered along the way, and how do they react to the Adventurer?

In our scenario, the characters include a witch, a dragon, some poisonous serpents, and a large frog. Each poses a particular problem for the Adventurer to solve. To complete the story, you (or another person if you prefer not to peek at the ending) will need to type in the new procedures, and make the final additions and modifications to the procedures already developed. Notice that in this issue we have included all of the procedures necessary to play *LOGO Adventure*—so even if you haven't been following this series, you can still play the game.

One modification involves the way **SET.UP** will be handled. It is probably safest to just key-in the new **SET.UP** procedure (see the listing). This must be saved in its own separate file: **ADVSETUP**. Next, erase **LOGO**'s memory (either restart, or type **ERALL** on IBM or Apple systems, or **ER ALL** on the C-64) and key-in the rest of the listing. This set of procedures should be saved with the name **ADVENTUR**.

At the beginning of the game, the modified procedure, **ADVENTURE**, will read **ADVSETUP** from the disk and execute it. Because **SET.UP** is used only once, it can be erased afterward to free up memory space. **ADVENTURE** does this, then performs a garbage collection to free up additional memory before calling **GETCOMMAND**.

Flags Keep Track of the Action

Some new commands have been added to **VERBLIST**, as have some variables that serve as *flags*. These flags, **:F1**, **:F2**, **:F3**, and **:F4**, will signal whether certain points in the story have been reached.

To demonstrate how flags are used along with other tools to test for conditions that affect the plot, let's go through the process of moving the dragon into its cavern lair. First, we need to determine whether the Adventurer is standing in the right place. The following line will take care of that:

```
C-64:      IF :HERE = 4 THEN IF :F2 = 0 PR (THERE IS
           A DRAGON HERE.) MAKE 'F1 :F1 + 1
IBM and Apple: IF :HERE = 4 (IF :F2 = 0 (PR (THERE IS A
           DRAGON HERE.) MAKE 'F1 :F1 + 1))
```

This line is inserted into **ID.LOC** and tested each time we move. If we are in Room 4, we are in the lair of the dragon. If flag **:F2** is set to zero, it means that the dragon hasn't yet been killed, so we see the **DRAGON HERE** message. Flag **:F1** is incremented, signaling that we have met the **DRAGON**. At this point, we might try typing something like **KILL THE DRAGON**. The program takes us down to these lines in the **KILL** procedure:

```
C-64      TEST (ALLOF :OBJ = "DRAGON :F2=0
           :HERE = 4)
           IFT THEN HAVE.IT? "SWORD PR (THE
           DRAGON DIES.) MAKE "ITEMS LPUT ((A
           DEAD DRAGON) 4) :ITEMS MAKE "F2 1
           STOP ELSE MAKE "F1 :F1 + 1
IBM and Apple IFT (IF HAVE.IT? "SWORD (PR (DRAGON
           DIES.) MAKE "ITEMS LPUT ((A DEAD
           DRAGON) 4) :ITEMS MAKE "F2 1 STOP)
           (MAKE "F1 :F1 + 1))
```

Without the sword, the Adventurer cannot kill the **DRAGON**, so the **:F1** flag is once again raised by one. Once the program loops back up to **ID.LOC**, this line is encountered:

```
C-64      IF :F2 = 0 THEN IF :F1 > 1 PR (THE DRAGON
           KILLS YOU.) ENDGAME
IBM and Apple IF :F2 = 0 (IF :F1 > 1 PR (THE DRAGON KILLS
           YOU.) ENDGAME))
```

ID.LOC once again tests to see whether the dragon is alive (**:F2=0**), and whether the unarmed Adventurer was unfortunate enough to stand around too long in the dragon's lair (**:F1>1**). If this is the case, the dragon does what one would expect such an ill-mannered creature to do, sending the program to **ENDGAME**. This procedure is called whenever one of the multiple endings is reached. It not only prints **THE END**, but it also erases the lists and variables in memory. Then it does a garbage collection and returns to **TOplevel**, which is where you were before starting the game. **ENDGAME** does the necessary "housecleaning" so the game can be replayed without running out of memory.

And what if the Adventurer *did* have the sword? Then the dragon dies and the object ((**A DEAD DRAGON**) 4) is appended to the list, **:ITEMS**, for an extra touch (this way, the Adventurer can choose to lug the dragon around!). Finally, the flag **:F2** is reset to one, so that **ID.LOC** will no longer print the **DRAGON HERE** message.

This raises the question of how to obtain the sword to dispatch the dragon in the first place, because it does not appear in plain sight. The answer is that the sword is in a "hidden" location. You can cause objects to appear out of nowhere by setting them up in room numbers outside the map. In this game, the sword is tucked away in Room 50. When certain actions are performed (we'll leave it to you to figure out which actions will produce the sword), these objects can appear in the room, or may be handed to the Adventurer by calling the **PUT.IN** using either **:HERE** or **-1** as an input. Objects can be made to disappear by reversing this process.

Managing Memory in LOGO

These procedures take up a lot of space, and making them fit without making major concessions to the story was a challenge. Often, while developing this game, we were greeted with a dismaying message that said we were out of storage space.

You must take some steps to keep memory clear. To begin with, save **SET.UP** as a separate file as described earlier, and make sure it is not among your general **ADVENTURE** procedures. Before saving your work, type **ER NAMES** to erase all of the lists and variables set up by the program. If you don't do this, they will end up being saved and loaded in along with the procedures. This will crowd memory during the bottleneck, when **SET.UP** is read in and temporarily needs the space.

You can always get a reading of how much memory you have left by performing a garbage collection. Do this by typing **RECYCLE** on IBM and Apple systems or **GCOLL** on the C-64. By typing **PRINT NODES** (the C-64 simply needs **:NODES**) and hitting the **(RETURN)** or **(ENTER)** key, you can discover exactly how much memory you have left. This garbage collection frees up areas no longer in use, and **NODES** tells you how many nodes of storage are left. (In **LOGO**, a node is 5 bytes).

If you want to try your hand at playing the game before analyzing the printout to see how the story works, have someone else type in the procedures for you. [All of the procedures will be included in *ON DISK* Vol. 5, No. 4—Ed.] If you are playing and meet an untimely (and perhaps unexpected) demise, just type **ADVENTURE** and you'll find yourself back in the old glen . . .

From here on in, you need only apply your own creativity to what is presented here. For example, you could add scoring procedures, magic spells, cryptic messages—anything you want. We hope this series of articles has provided some new insights about **LOGO**'s list-processing capabilities, and some incentives to explore further.

HCM

HCM Glossary terms: flag, garbage collection

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

HOME COMPUTERTM

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



Joysticks For Business

Simple Cursor Control For The IBM PC

Kraft Systems has introduced its Executive Cursor Control joystick system for the IBM PC family. Attempting to bring the joystick from the gaming room into the board room, Kraft Systems' new program offers dedicated support for Lotus 1-2-3 and IBM's TopView programs. Using a joystick provides the user with a familiar and simple method for moving the cursor precisely about the screen. The system comes with two speeds, slow and fast, and allows the user to operate the joystick with one hand and



the keyboard with the other. The Executive Cursor Control package retails for \$69.95.

Kraft Systems
P.O. Box 1268
Vista, CA 92083
(619) 724-7146



Carrying On With Math

Learn Addition The Computer Way

Renaissance Learning Systems announces Carrying: Regrouping for Addition, a self-contained math package for the Apple II Family of computers. The program teaches the concept of carrying and features a diagnostic practice session that identifies a

child's weaknesses and strengths. A teacher's version includes a teacher management disk for keeping track of up to 80 students in 6 different classes. It retails for \$45, and the home version retails for \$35.

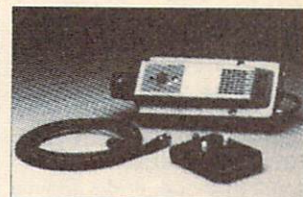
Renaissance Learning Systems
Tecumseh Building
Jamesville, NY 13078
(315) 445-2508



Catch That Image

Digetizer Reproduces C-64 Video Images

Cardco has announced a video digitizer for the Commodore 64. Digi-Cam comes complete with digitizer, all software, cables, and a Panasonic monochrome camera for \$250. The Digi-Cam image can be seen on your computer monitor, stored on disk, transmitted through a modem, and printed on most printers. Digi-Cam produces a 320 x 200 dot screen image in 5



grey scales. Paint Now, a software program included with the package, lets you edit any image created with the digitizer.

Cardco Inc.
300 S. Topeka
Wichita, Kansas 67202
(316) 267-3807



More Memory For The TI-99/4A

An Alternative To The TI Box

CorComp is adding the 32K Stand Alone black box to its line of "daisy chain" products for the TI-99/4A. The Stand Alone plugs directly into the 99/4A console or can be daisy-chained

through the TI Speech Synthesizer. A full line of additional "daisy chain" products for the TI-99/4A is presently in development. This first unit retails for \$119.

CorComp Inc.
1255 North Tustin Ave.
Anaheim, CA 92807



Let Your Finger Do the Moving

PC Touch Pad Makes Data Entry Easy

Key Tronic has released another keyboard for the IBM PC. The Key Tronic KB5153 utilizes a touch pad for precision pointing capabilities. The touch-pad portion of the keyboard operates in 4 modes: cursor key, mouse, absolute, and a function key mode, which allows you to combine some of the other functions to suit specific needs. The Key Tronic KB5153 is part of



the company's Professional Series line of keyboards tailored to business and professional applications.

Key Tronic
P.O. Box 14687
Spokane, Washington 99214
(509) 928-8000



Do Your Science Project On Your Apple

Science Toolkit Includes Sensor Probes

Science Toolkit Master Module combination hardware and software package is now available from Broderbund Software for \$59.95. Available for the Apple IIe or IIc, Science Toolkit Master Module comes with a temperature-sensing probe, a light-sensing probe, and a special

interface box that connects these devices to the joystick port. It also comes with 4 on-screen lab instruments: a thermometer, a light meter, a timer, and a strip chart. The product comes with a 128-page user's manual with instructions for a variety of experiments.

Broderbund Software
17 Paul Drive
San Rafael, CA 94903-2101
(415) 479-1170



Get Physical With Your C-64

Muscle Out Commodore's Troubles

Howard W. Sams & Co. is marketing a new book for the C-64. The Commodore 64 Troubleshooting and Repair Guide details specific C-64 malfunctions and offers advice on developing custom hardware and software tools. With step-by-step diagnostic techniques, preventive maintenance pointers, servicing tips, and troubleshooting methods, the book is designed to help people avoid downtime and high repair bills. It is priced at \$18.95.



Howard W. Sams & Co.
4300 W. 62nd St.
Indianapolis, IN 46268
(317) 298-5400



Word Processing for Science, Academia

Technical Writers Take Heart

Lifetree Software is shipping Volkswriter Scientific, a microcomputer word processor aimed at the scientific and academic markets. Volkswriter Scientific is a bit-mapped word processor for the IBM PC featuring over 400 scientific and mathematical characters,

multiple type-styles, and the Roman and Greek alphabets. It offers on-screen visual composition of text with formulas and symbols, on-screen tutorials, and 9 different help menus. Volkswriter Scientific retails for \$495.

Lifetree Software
411 Pacific St.
Monterey, CA. 93940
(408) 369-3070



MIDI Magic Makes Music

MIDI Interfaces Available For Apple II, C-64

Passport Designs has introduced 2 new advanced programs for MIDI: the MIDI/4 plus, and the MIDI/8. Both the 4- and 8-channel recording programs offer auto-correct, punch in/out, fast forward/rewind, sequence chaining, sync-to-tape, and MIDI and drum machines. Both programs

offer multi-track recorder qualities with unlimited over-dubbing, real-time editing, tempo control, and accurate recording of all controllers. MIDI 4 and MIDI 8 are available for the Apple IIe and the C-64. The MIDI/4 plus retails for \$99.95, and the MIDI/8 retails for \$149.95.

Passport Designs Inc.
625 Miramontes St.
Half Moon Bay, CA 94019
(415) 726-0280



Learn Language Arts and Library Skills

Scholarly Travels With King Arthur

Two new educational programs are available for the Apple II Family from Learning Well: Word Magic, a language-arts program that teaches children reading skills as they travel through King Arthur's countryside; and Library Adventure, a program that teaches library reference skills as children move through a

simulated library. Word Magic can be modified to suit the skill level of the child, and it includes cumulative scorekeeping capabilities. Library Adventure allows teachers to create their own questions, and comes with two levels of play. Both programs retail for \$49.95.

Learning Well
200 South Service Road
Roslyn Heights, NY 11577
(800) 645-6564



Mac Makes MacMusic

Singing Out In Four Voices

Hayden Software has introduced a new music program for Macintosh. MusicWorks, developed by MacroMind, lets you create and perform music on the Macintosh. The program allows you to compose on a staff or plot notes on a special grid, and features eight instrument sounds that can play up to 4 voices at a time over an 8-octave range. A full range of special

effects let you vary the tempo, intensity, timbre, and meter. The program comes with sample songs which can be played, edited, and replayed. Included in the users manual is a quick-start tutorial, musical instruction for beginners, and a glossary of musical terms. MusicWorks even prints scores for one instrument or whole ensembles. It retails for \$79.95.

Hayden Software Co. Inc.
600 Suffolk St.
Lowell, MA 01854
(617) 937-0200



HOME COMPUTERTM

product news

TI-Writer Lives On

Second Generation Released For 99/4A

Tex-Comp has taken the original TI-Writer word-processing program, added new character sets, lower-case letters, and all of Texas Instruments' upgrades, and released 99-Writer II for the TI-99/4A. A new user's manual has been developed

to accompany the disk, which is compatible with Dragon Slayer Software's Auto Spell-Check program. The \$19.95 program loads with Extended BASIC, Editor Assembler, or Mini Memory, and requires a disk drive and 32K memory expansion.

Tex-Comp
P.O. Box 33084
Granada Hills, CA 91344
(818) 363-7331



Brown-Bagging Words

Word Processing for Apple II

Software Resource Group has introduced Brown Bag Software, an integrated data-base management system and word processor for the Apple II Family. Claiming its product is "software for the proletariat," Brown Bag Software offers on-line help at all times, data- and text-merging between modules, and search and replace functions. It is designed so that persons with very little actual com-



puter experience will be able to use it. Brown Bag Software retails for \$49.95.

Software Resource Group Inc.
1095 Airport Road
Minden, Nevada 89423
(702) 782-9731



Some New Sights & Sounds

Music Programs Enhanced

Sight & Sound Music Software has made price cuts and enhancements to its line of music software for the Commodore 64. Sight & Sound has added recording and note displays and more preset instrument sounds and background accompaniments to their Incredible Musical Keyboard package. The Kawasaki Rhythm Rocker now offers more

extensive recording capabilities, music printouts, note displays, and a new program—Magical Musiquill—on the flip side of the disk. Music Processor now features music printouts, music editing, and easier recording capabilities. Along with these enhancements, Sight & Sound has reduced prices on these programs by \$10.

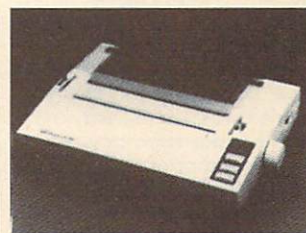
Sight & Sound Music Software Inc.
P.O. Box 27
New Berlin, WI 53151
(414) 784-5850



Covering The Printer Spectrum

A Printer For Both IBM and Apple

Epson America has released the Spectrum LX-80, a new dual-mode, dot-matrix printer featuring plug-in compatibility with the IBM PC, PCjr, and the Apple IIc. It offers fast draft copy and near-letter-quality capabilities, and prints in 80-column mode. The Spectrum LX-80 prints at 100 characters per second (draft), and 16 characters per second (NLQ). It comes with tractor and friction paper feed and offers a cut-sheet feeder as an option.



The Spectrum LX-80 prints in a variety of typesizes, and is available with a generic parallel printer interface cartridge. It retails for \$389.

Epson America Inc.
2780 Lomita Blvd.
Torrance, CA 90505
(800) 421-5426



C Is For Commodore

A C-Language Compiler for C-64, 128

Abacus Software is releasing a C-language compiler for the Commodore 64 & 128. Priced at \$79.95 Super C-language compiler is the first full C compiler to work on the C-64. Super C is a complete development system with an editor

capable of handling source files up to 41K in length. The compiler produces 6510 machine code. Super C's library supports standard and Commodore-oriented functions and conforms to the Kernighan & Ritchie standard.

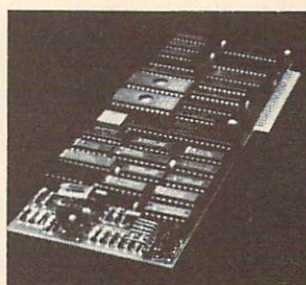
Abacus Software Inc.
P.O. Box 7211
Grand Rapids, MI 49510
(616) 241-5510



Popular Program Runs On All Apples

AppleWorks on the II+

Videx Inc. has announced a new product that enables AppleWorks to be used on any Apple II+ that has a Videoterm 80-column card, and the one-wire shift modification. The program replaces the special keys of the Apple IIe with commands available on the Apple II+ keyboard. Called the AppleWorks Modifier, the program requires 64K of RAM, and provides 10K of



editing memory. AppleWorks Modifier is \$49.

Videx Inc.
1105 N.E. Circle Blvd
Corvallis, OR 97330
(503) 758-0521



Are 2 Drives Better Than 1?

Increase Juniors Memory

Racore Corporation is introducing a new product in its line of companion peripherals for the IBM PCjr. The Drive Two Enhancement Package allows users to increase memory up to 512K and operates in DOS 2.0, DOS 2.1, or DOS 3.0. Other features include a parallel printer port, clock calendar, memory-expansion slot, and side-bus expansion.

The package fits snugly on top of Junior, and can be installed in less than 10 minutes using just a flat-head screwdriver. The Drive Two Enhancement Package retails for \$675. Racore is planning to market a Direct Memory Access product for the PCjr in mid-June.

Racore Corporation
10 Victor Square
Scotts Valley, CA 95066
(408) 438-7255



Decoder Ring Grows Up

Solve Cryptic Messages With Computer

Arden Enterprises has released CryptoCompute for the TI-99/4A allowing users to decode cryptograms using their computer rather than penciling guesses into the newspaper. The program is available in two versions: Version 1, requiring a disk drive and Extended BASIC; and Version 2, requiring a disk drive, Extended BASIC, and

32K memory expansion. CryptoCompute comes with 47 original cryptic messages from assorted subjects, or you can key-in cryptograms from other sources. A subscription service is available that provides new data disks each month containing an original set of cryptic messages. Both versions are \$19.95.

Arden Enterprises
P.O. Box 89
Walkersville, MD 21793
(301) 845-6024

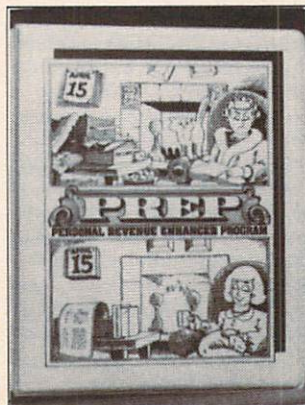


Balancing The Books

Program Helps Manage Money

A Personal Revenue Enhancement Program (PREP), for managing expenses, checking accounts, taxes, credit-card expenditures, and cash has been released by U.S. Digital. PREP lets the user define up to 250 spending and income categories. The program is MS DOS- and PC DOS-compatible and is currently available for the IBM PC for \$59.95.

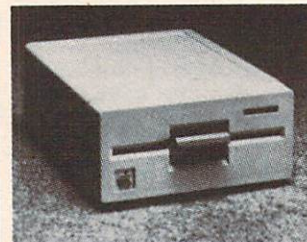
U.S. Digital Corporation
5687 S.E. International Way
Portland, OR 97222
(503) 654-0668



Designer Disk Drives From Apple

Disk Driving in Style

Apple Computer has released a new compact, single, half-height, floppy-disk drive for the Apple II, II+, and the Apple IIe. Called the UniDisk, it is fully compatible with the more than 10,000 software programs available for the Apple II Family. The new drive provides 143K of storage capacity, and is functionally identical to Apple's Disk II drive, which it replaces. UniDisk may be purchased with or without a controller card, however, the first UniDisk purchased needs the card in order to be connected to the computer. Subsequent disk drives can be plugged into the back



of the first disk drive. UniDisk's control card is not compatible with the controller card shipped with the Disk II, so users who have a Disk II connected to their system must purchase a UniDisk with a controller card. UniDisk retails for \$429 with controller card, and \$329 without.

Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014
(408) 996-1010



Go For The Gold . . . Again

Epyx Sequel has 8 New Events

Epyx has released a sequel to its program Summer Games, called Summer Games II. It contains 8 new ways to compete for Olympic gold: cycling, kayaking, fencing, equestrian competition, and several new track and field events. Summer Games II's graphics and animation include colorful opening and closing ceremonies complete with doves, and a sound track of 18 national anthems. Summer Games II runs on Apple II Family and C-64 systems, and retails for \$40.

Epyx Inc.
1043 Kiel Court
Sunnyvale, CA 94089
(408) 745-0700

mer Games II's graphics and animation include colorful opening and closing ceremonies complete with doves, and a sound track of 18 national anthems. Summer Games II runs on Apple II Family and C-64 systems, and retails for \$40.



Cramming In More Memory

Apple Add-on Adds Extra Muscle

BFM Products has announced a new plug-in memory expansion board for Apple II, II+, and IIe computers. Called the Cramapple, it's available in either 128K or 512K versions. Cramapple can be accessed as a single large disk

of memory, or as distinct banks of memory equal to 3, 35-track floppy disks. Depending on the applications, Cramapple can speed processing time by up to 50 percent. It is priced at \$149 for the 128K board, and \$349 for the 512K board.

BFM Products
P.O. Box 942
Felton, CA 95018
(408) 263-9378



algorithm – A set of rules or procedures used to solve a problem. (See the Algorithm-A-Tricks column in this issue.)

application program – A software program that is not a necessary part of the computer's environment (like the Disk Operating System software or BASIC Input/Output System software) but is applied to the solution of a user's problem. Examples are word processing, spreadsheet, and data-base programs.

array – A collection of items (data), identified by a common variable name and arranged in such a way that the computer can search via the array's subscript to find and retrieve specific items. (See "subscript.")

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

aspect ratio – The ratio between the width of a pixel and the height of a pixel on the screen.

assembly language – A programming language that is one step up from machine language. Instead of using numbers—as in machine language—short combinations of letters, called mnemonics, are used to code programs.

auto-boot – A program that will automatically run when loaded.

BASIC loader – A BASIC program that loads a machine-language routine into memory.

baud – A unit of speed in data transmission, usually indicating the number of bits per second.

benchmarking – A program that compares the capabilities of 2 or more systems or programs relative to each other—i.e., for processing speed.

binary-coded decimal arithmetic – Arithmetic using a 4-bit representation of a decimal number.

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

bit – Contraction of Binary digit. It is 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.

boot disk – A diskette that has a special recording in a certain spot that is searched for and used by a computer system to initialize its environment and Disk Operating System.

buffer – A temporary storage area for data.

byte – A sequence of eight bits used to represent one character.

compiled language – Any computer language where the source code is translated into machine language before the program is run.

concatenate – To link together, as in appending one file to another.

CONFIG.SYS file – When an IBM PC or PCjr that has just been "booted" from a PC-DOS boot disk, the DOS immediately searches the disk for this file. If it is there and has coded instructions following the correct syntax, the system will activate other software referenced in these lines to reconfigure the system.

constant – A value which remains unchanged throughout a program's operation.

control character – A character that is not intended to be printed, but which has special meaning to peripheral devices or as a delimiter in data transmission.

counter – A control variable used in a loop to step sequentially through a process.

data segment – A portion of memory indexed for data rather than code.

default – Some programs or systems allow you a choice of several options. If you do not pick one, one is automatically assigned, by default.

device drivers – Software programs that enable the Disk Operating System to interface with nonstandard input/output devices and even "fool" the system into treating part of memory as an input/output device. (See RAM Disk.)

dialogue box – On the Macintosh, a special window that appears on screen to inform the user of present options.

dimension – In an array, a section identified by a particular subscript.

display – Data shown on a video screen or monitor.

DOS – An acronym for Disk Operating System.

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).

expression (mathematical) – A group of mathematical terms that express a value.

file – A collection of related records (data items), treated as a unit—i.e., an address list may compose one file, a writing project another, etc.

flag – A variable that signals a specific occurrence in a program's execution, which can be tested later for program control.

flow chart – A chart which presents a graphic representation of a program's or algorithm's flow of control.

frequency – The rate at which a sound wave moves—directly related to pitch.

garbage collection – A pause in normal program operation when the system frees up memory space by deleting un-needed variable storage and scratch-pad memory.

hard-coded – When values of variables or constants are specified within the program code itself and not subject to change by user input.

heap – On the Macintosh, a general RAM memory area used for programs and data on demand.

hertz – A unit of frequency that is equal to one cycle per second.

hexadecimal numbers – A base 16 numbering system using decimal digits 0 to 9 and the letters A through F.

hypotenuse – In a right triangle, the side opposite the right angle.

initialization – The setting of program counters, addresses, and switches to starting values at specific points in a routine.

input buffer – An area of memory reserved to store input data until the program can process it.

integer array – An array of integer numbers. (See "array" above.)

interpreted language – Any computer language whose source code executes one instruction at a time, as opposed to a compiled language.

interrupt – A break in a program's execution, caused by either an external source or a signal that directs the computer away from the program sequence.

legal input – Data entered into the computer in a format acceptable for the program being used.

loop – A sequence of instructions in a program that repeats until a set of conditions is satisfied.

machine language – The native language of the microprocessor in a computer expressed in terms of binary ones and zeroes.

mnemonic – A code or symbol that helps people remember something specific, often made up of letters from the word or phrase it represents.

multi-dimensional array – An array with more than one subscript, such that the data is arranged in a matrix.

page zero – The lowest 256 (\$100 hexadecimal) locations in the 6502 microprocessor's memory, the processor used in Apple computers. Due to special addressing modes that use the area, it is often used by interpreters and operating systems for storing special codes and pointer addresses.

parallel port – The connector on a computing device that transmits and receives data over several wires (usually 8 paths) simultaneously or in "parallel." This port is most often used to connect a printer to a computer.

parameter – A variable used to control a particular process.

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.

printer buffer – An area of Random Access Memory in the computer used to temporarily store information that is awaiting transmission to a printer while the main portion of the computer is used for some other function. (See "spooler.")

program file – As opposed to a data file, a file containing program code that can be stored and retrieved.

radian – A unit of measure in an angle. Computers use radians instead of degrees to measure angles. There are 6.28318530718 (2π) radians in a circle (360 degrees).

RAM – Random Access Memory. It can be programmed, erased, or altered by users.

RAM disk drive – An area of Random Access Memory set aside for use, through a "device driver," with the Disk Operating System as a floppy disk drive. This has the great advantage of speed and the drawback of forgetting everything when the power is turned off.

ROM – Read Only Memory. The fixed, permanent memory bank of a computer inaccessible to users.

sector – Circular sections on a floppy disk, similar to grooves on a record.

sequential text file – A disk file of ASCII characters arranged sequentially.

SID – Sound Interface Device chip in the Commodore 64.

spooler – A "device driver" program that intercepts data inbound from an input device or outbound to an output device (such as a printer) and temporarily stores it in memory set aside to free up the computer's time for handling other functions. This type of software only works on computers with certain hardware features.

spreadsheet – A program for manipulating numbers in tabular form. A grid is displayed, and users may put numbers into the cells of the grid and specify relationships between rows and columns. A change in one cell may affect the entire grid.

sprite – A user-defined moveable graphic character.

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.

string – A consecutive set of similar data items—usually bits or characters.

string array – An array of strings. (See "array" and "string.")

subscript – An index for an array that indicates a specific element or identifies a dimension. (See "element" and "dimension.")

tokenize – The process in which a BASIC command or function is abbreviated into one character.

track – Cross-sections of a disk. Similar to slices in a pie.

transient code segment – Code brought into memory from disk because the entire interpreter cannot reside in memory at one time.

vector – A pointer or passageway to important routines in the computer's memory.

vector (physics) – A quantity which not only has magnitude, but also direction.

video buffer – Also known as screen memory, an area of Random Access Memory that is used to store the current image data used to produce the display that is seen on the monitor or TV.

window – An area of the screen reserved for output. Other parts of the screen outside the current window will not be affected by output from the computer.

WOM – Write-Only Memory.

word wrap – A feature of some word-processing programs that will automatically move a word to the next line if it cannot fit in its entirety at the end of the line being worked on.



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.

Look at the Key-in-Reference (Figure 1 below) see how each character actually appears in the listing. By checking any questionable characters with the Key-in-Reference, you can reduce errors to a minimum.

Figure 1: Key-in Reference

100	REM	1234567890	!@#\$%^&*()_+={} ~\`/;:~>^
		ABCDEFGHIJKLMN	OPQRSTUVWXYZ[]- _<~>^
		abcdefghijklmnopqrstuvwxyz	~\`/;:~>^

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 twenty 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 \$, and the uppercase B for the number 8. The last error is especially likely when working in hexadecimal numbers which are composed of 0-9 and the uppercase letter 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 key in the listings character for character and space for space.

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: 8SHIFT CRSRLEFT- tells you to hold the [SHIFT] key































down and press the cursor left key (on the bottom right of the keyboard) eight 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 or 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 place.

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. (Notice that the hand symbols always appear within quotation marks—as in a print statement.)

Figure 2: C-64 Special symbols

When you see:	Press the keys:	To get this display:
"CRSRDOWN"		
"CRSRRIGHT"		
"SHIFT CRSRLEFT"		
"SHIFT CLR"		
"HOME"		
"SHIFT K"		
"CMDR K"		
"F1"		
"SHIFT F2"		
"CTRL RVSON"		
"CTRL RVSOFF"		
"CTRL BLK"		
"CMDR BLK"		
"3SHIFT CRSRLEFT"		
"SHIFT CLR 2CRSRDOWN"		

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 (special memory cards, Speech Synthesizer, 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 medium from HCM will bear a revised version number.

	5	4	1
Volume no.	_____		
Issue	_____		
Version	_____		
1	= original program		
2	} = no. of update		
:			
:			
n			
End of Program = HCM			

HOME COMPUTER™

PROGRAM LISTINGS

CONTENTS

	Software Instructions Page No.	 Page No.	 Page No.	 Page No.	 Page No.
Run-Day-View	16	90	94	97	99
Programmer's Window		74	72	75	73
Trig-Trix	19	102	105	108	110
Programmer's Window		78	76	79	77
Archeodroid	22	112	114	115	117
Programmer's Window		82	80	83	81
Mine Over Matter	24	119	121	124	126
Programmer's Window		86	84	87	85
One-Liners	27	27	27	27	27
MAC-ROs (Picture Maker)	28	29			
IBMpressions (Eggs)	30			31	
Razzle Dazzle (Composer)	32				33
Apple Seedlings (Pie Chart)	34	35			
Commodore Hornblower (Filters)	36		128		
C-64 Tech Note (Tape Merge)	55		128		
IBM Tech Note (Character Graphics)	56			56	
Build A LOGO Adventure	63	88	89	88	

SPECIAL NOTE: TO RUN HCM PROGRAMS ON THE APPLE II+ YOU MUST HAVE AT LEAST 64K.

☐

WHAT IS A PROGRAMMER'S WINDOW?

Home Computer Magazine has always served those eager to expand their programming knowledge by including specific information about the software published in these pages. This information has, in the past, been part of the main introductory and instructional text (now identified with the "Software Instructions" edge-of-page tabs), and consisted of comments about each machine version and a line-numbered annotation of each version. Now we have consolidated the older formats together with two new features in one separate section: the "Programmer's Window." This section now includes, for each program, 4 different pages—one for each machine version (Apple II Family, Commodore 64, IBM PC & PCjr, and TI-99/4A). Each of these machine-specific pages 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)

REMARKS

One of the most useful features of *Run-Day-View* is its ability to set markers. In the C-64 version of this program, markers are stored in the two-dimensional array `MK()`. The first dimension (a value 1-4) determines the particular marker within a day, and the second dimension (a value 1-31) determines the particular day within a month. For example, `MK(2,15)` will give you the appointment number for the second marker on the 15th of the month. Because only 4 markers can be set per day, the first dimension cannot exceed a 4. The flow chart describes how this program uses the `MK()` array to set markers.

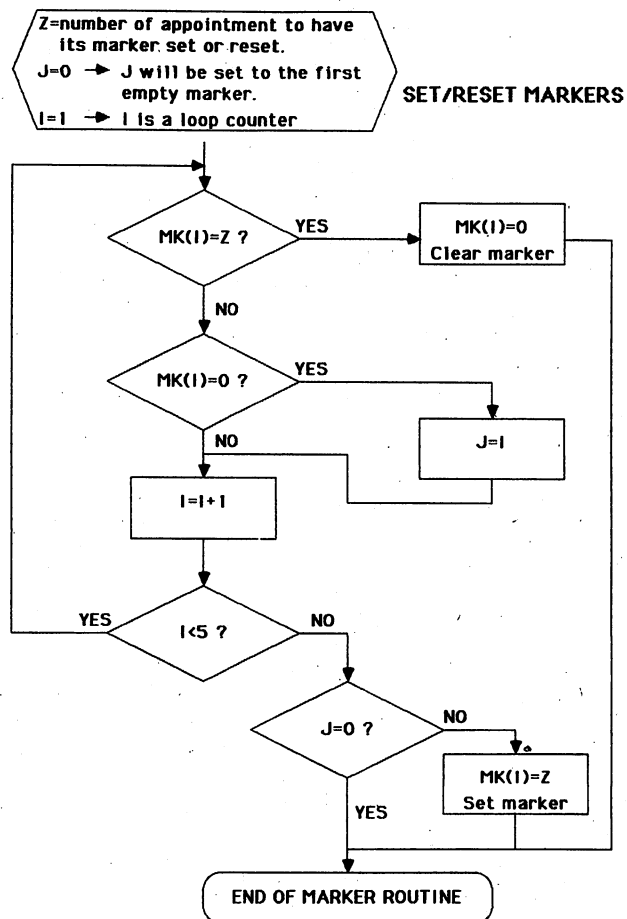
When the markers routine is first entered, the `Z` variable is set to equal the appointment that will have its marker set or reset. This appointment is represented by a number between 1 and 18—1 being the first appointment of a day and 18 being the last. Next, each marker—represented by the `MK()` array—is searched to see if a marker is already set for that appointment. If the marker has been set, then it is cleared and the routine ends. If a marker has not been set for that appointment, one of two things can happen. If an empty marker was found, then that marker is set for the appointment stored in `Z`. If there are no empty appointment markers, then the marker entry is ignored and the routine ends. Empty markers are monitored in the `J` variable. If `J` stays set to zero, then there are no empty markers; otherwise, `J` equals the last empty marker found.

HCM Glossary terms: array, dimension.

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200-290	Display title screen and initialize variables.
300-400	Main menu.
410-480	Edit appointments menu.
490-770	Set date and time.
780-910	Edit appointments.
920-1070	Set markers.
1080-1110	Sort markers.
1120-1230	Print routines menu.
1240-1430	Print one day in book.
1440-1560	Print a selected day in book.
1570-1660	Print appointment book.
1670-2070	Print weekly summary.
2080-2240	Print phone numbers.
2250-2320	Input a file name.
2330-2530	Save routine.
2540-2810	Load routine.
2820-2900	Print a page to the screen.
2910-2930	Buzz and pause.
2940-2980	Jump pages in appointment book.
2990-3140	Edit phone numbers.
3150-3220	Add a phone number from appointment editing screen.
3230-3580	Input routine.
3590-3620	Input one character.
3630-3670	Find the first day of a month.
3680-3710	Find day of the week based on the first day of the month.
3720-3780	Print the status line.
3790-3800	Clear the middle of the screen.
3810-3820	Place cursor at X,Y.
3830-3870	Exit program.
3880-3920	Data for days and month.

DESIGN FOCUS



DIRECTORY OF VARIABLES

Variables	Functions
TS	Program title.
AP\$()	Appointment text.
MK()	Markers for each appointment.
PN\$()	Phone numbers.
MS()	Months of a year.
ND()	Number of days in each month.
M	Current month.
DW\$()	Days of the week.
DW	Day of the week.
D	Day of the month.
FD	First day of the month.
YR	Year.
BT	Beginning time for appointments.
FL\$	File name.
KS	Key input.
K	ASCII of key input.
P	Position of cursor in edit modes.
E	Edit mode flag.
I,J	Loop counters.
A	Utility variable.
B	Utility variable.
C	Utility variable.
L	Utility variable.
S	Utility variable.
SS	Utility variable.
T	Utility variable.
X,Y	Utility variables.



Run-Day-View

REMARKS

Due to memory considerations on the TI-99/4A, *Run-Day-View's* appointment markers are not stored as numeric variables, but as ASCII characters in a string array. (See the "Home Computer Tech Note" for the TI-99/4A in Vol 5, No 2.) Because there are 4 possible markers per day, and a maximum of 31 days in a month, a 4-element, 31-character long array is used. This array is `MKS()`.

Now, if your first marker is set for the second appointment on the third day of the month, then the third character in `MKS(1)` would be set to an ASCII two. Got that? Well, to make things easy, two subroutines are used in order for the program to index into this array. Lines 5050-5060 will return the appointment number to which a certain marker is set. The routine located in lines 5070-5080 will set a marker to a given appointment. For an illustrated view of how the marker array works, refer to the pictorial on the right.

Even with this method of crunching memory, 99/4A users with no memory expansion and Extended BASIC should be careful with the amount of data you try to enter. Under such an arrangement, an average of 50 entries can be made without getting an out-of-memory error. Any more than that, and you're walking on thin ice. As a rule of thumb, save your data often to avoid losing anything important.

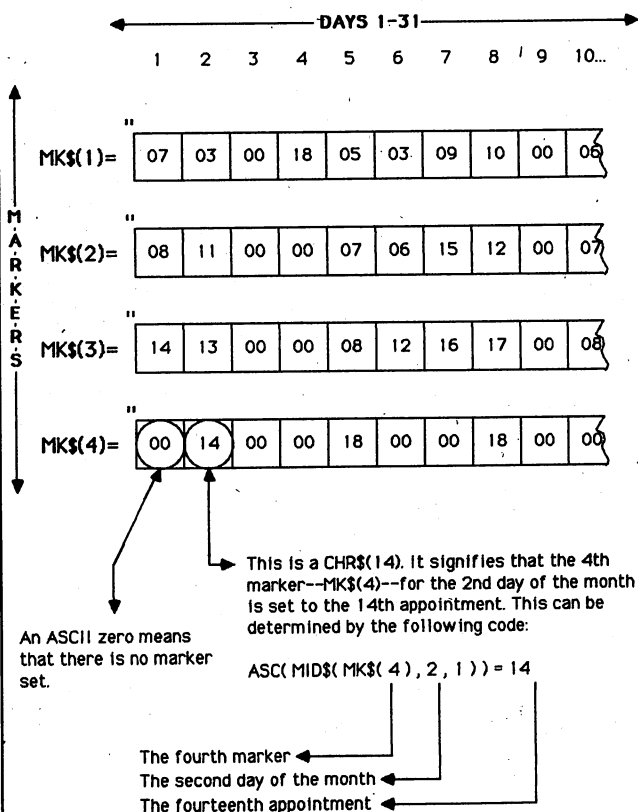
HCM Glossary terms: array, element.

LISTING ANNOTATIONS

Line Nos.	
100-180	Program header.
190-250	Initialize variables.
260-280	Display title screen.
290-350	Main menu.
360-420	Edit-appointments menu.
430-500	Print-routines menu.
510-1090	Set date and time.
1100-1210	Edit appointments.
1220-1360	Enter an appointment.
1370-1410	Input a letter (A-R).
1420-1510	Print a page to the screen.
1520-1570	Print markers.
1580-1700	Print time and appointment.
1710-1800	Input a day of the month.
1810-1970	Set markers.
1980-2030	Exit program.
2040-2440	Load and save routines.
2450-3180	Print weekly summary.
3190-3290	Print appointment book.
3300-3450	Print a page from book.
3460-3750	Print on day of appointments.
3760-3800	Input printer parameters.
3810-3890	Check for non-numeric character.
3900-3990	Find first day of month.
4000-4050	Find day of the week based on the first day of the month.
4060-4080	Input one character.
4090-4140	Error message.
4150-4190	Prompt to position paper.
4200-4230	Input a menu selection.
4240-4270	Input a (Y/N) response.
4280-4500	Edit phone numbers.
4510-4640	Sort phone numbers.
4650-4820	Print phone numbers.
4830-4970	Input phone number from edit-appointments screen.
4980-5040	Initialize markers.
5050-5080	Set and read markers.
5090-5110	Data for day of the week and months.

DESIGN FOCUS

THE MARKER ARRAY



DIRECTORY OF VARIABLES

Variables	Functions
APS()	Appointments text.
MKS()	Markers for each appointment.
PN\$()	Phone numbers.
MS	Current month.
ND	Number of days in current month.
DW\$()	Days of the week.
DW	Current day of the week.
D	Day of the month.
FD	First day of the month.
YR	Year.
BT	Beginning time for appointments.
FL\$	File name.
BL\$	Blank characters.
K	Input variable.
S	Input variable.
I	Loop counter.
J	Utility variable.
A	Utility variable.
B	Utility variable.
C	Utility variable.
L	Utility variable.
M	Utility variable.
SS	Utility variable.
T	Utility variable.



REMARKS

The Apple version of *Run-Day-View* uses an elegant algorithm to handle its menu-driven format. Three major menus are used in this program: the Main Menu, the Edit Appointments Menu, and the Print Routines Menu. The Design Focus illustrates how each of these are linked together.

To avoid having to create **PRINT** statements for every menu, the Apple version uses an array to store its menu data. During the initialization of the program, the menu options are read from **DATA** statements and stored in the **MENU\$()** array. Now, whenever a menu is to be displayed, the array is referenced and the proper options are printed.

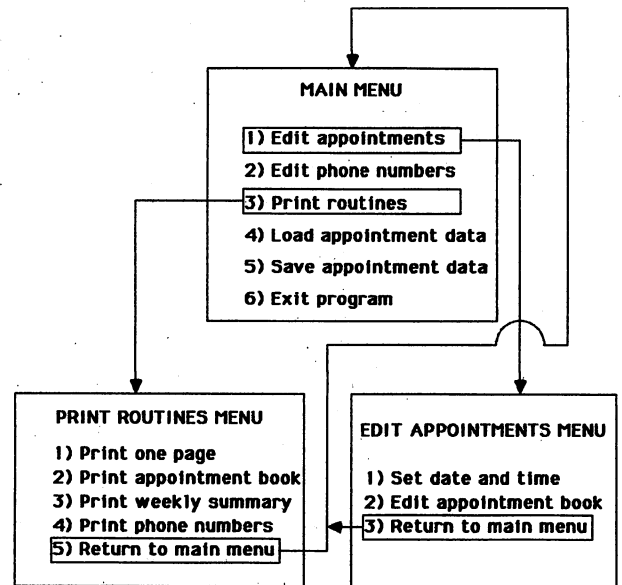
To save the programmer even more time on the creation of menus, a general menu subroutine located in lines 1690-1780 is used. This subroutine displays the different menu options and inputs a selection. In order for the subroutine to do all of this, certain parameters have to be passed to it. Besides **MENU\$()**, another array is used to pass the menu data. This array is **CX()**. Here is what can be found in this array:

Variables	Functions
CX(1)	The menu currently being displayed. This is represented by a value between 1 and 3, referring to the three different menus.
CX(2)	The number of options available in this menu.
CX(3)	Number of first menu option in the MENU\$() array. For instance, the first option of a menu is found by printing MENU\$(CX(3)) .

HCM Glossary terms: algorithm, array, initialize, parameter.

DESIGN FOCUS

MENU STRUCTURE



DIRECTORY OF VARIABLES

Variables	Functions
AP\$()	Appointment text.
MK()	Markers for each appointment.
PN\$()	Phone numbers.
MNS()	Months of a year.
ND()	Number of days in each month.
M	Current month.
DWS()	Days of the week.
DW	Current day of the week.
D	Day of the month.
FD	First day of the month.
YR	Year.
BT	Beginning time.
FL\$	File name.
MENU\$()	Menu-option data.
CH,CX()	Menu parameters.
LO(),LI()	Menu parameters.
IN\$	Input string.
CN,CN\$	Input parameters.
HC,HT	Input parameters.
HX,HZ	Input parameters.
N1,N2	Input parameters.
N3	Input parameter.
PD	ProDOS flag.
BK\$,BL\$	Special characters.
CR\$,DN\$	Special characters.
DR\$,ET\$	Special characters.
LFS,RT\$,UP\$	Special characters.
DI,IT,JI	Loop counters.
A,B	Utility variables.
C,DL	Utility variables.
DX,TB	Utility variables.
TM,TY	Utility variables.
VC,VT	Utility variables.
VT,VX	Utility variables.
WAO,WZ	Utility variables.
X,Y	Utility variables.
A\$,PB\$	Utility variables.
PFS,PG\$	Utility variables.
SS,TS	Utility variables.
X\$,Z\$	Utility variables.

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200-260	Main control loop.
270-310	Exit program.
320-620	Initialize program.
630-670	Main menu.
680-730	Edit appointments menu.
740-970	Set date and time.
980-1030	Edit appointment book.
1040-1470	Edit an appointment page.
1480-1570	Edit phone numbers.
1580-1680	Sort phone numbers.
1690-1780	General menu routine.
1790-1970	Input routines.
1980-2070	Print-routines menu.
2080-2170	Print one page from book.
2180-2240	Print appointment book.
2250-2520	Print weekly summary.
2530-2780	Print phone numbers.
2790-2870	Print prompt subroutines.
2880-2930	Turn on printer.
2940-2980	Find first day of the month.
2990-3020	Find days of the week based on the first day of the month.
3030-3190	Number entry.
3200-3430	Load appointment file.
3440-3630	Save appointment file.
3640-3740	Input file name.
3750-3800	Get a character from keyboard.
3810-3990	Error-handling routine.
4000-4180	Check file name.

REMARKS

In an IBM program like *Run-Day-View*, the way the computer accepts its information is very important. We want to be able to screen all input so that every character can be both stored on disk and output to the printer. To check the input, this IBM PC and PCjr version of the program uses the **INKEY\$** command in conjunction with the **INSTR\$** function.

First, a string called **IL\$** is initialized to contain any unwanted characters. Now, when a character is received from the keyboard, it is compared to **IL\$** with the **INSTR\$** command. If there is a match, we simply ignore the character and loop back up to the **INKEY\$** statement. Here's a simple routine that uses this same technique to reject certain punctuation marks.

```
100 IL$ = ",.?!;:"
110 K$ = INKEY$:IF K$ = "" THEN 110
120 IF INSTR$(IL$,K$) THEN 110
```

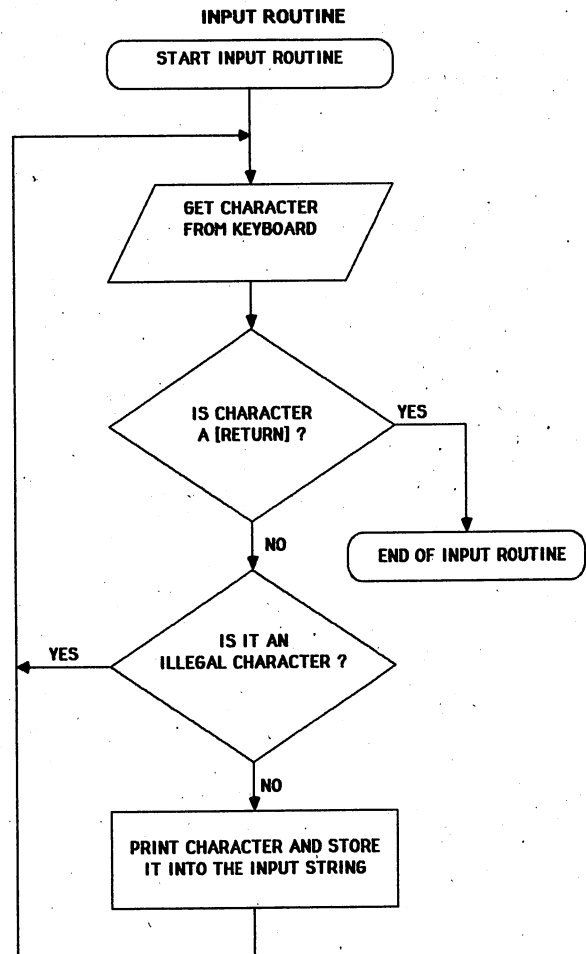
For a more detailed description of the actual input routine used by this program, see the flow chart on this page. This flow chart generally illustrates the subroutine located in lines 1050-1330 of *Run-Day-View*. Certain functions, such as checking for various editing keys, are not included in the flow chart. Also, depending on where this routine is called from, the IBM's function keys can affect the input.

HCM Glossary term: initialize, loop.

LISTING ANNOTATIONS

Line Nos.	
100-200	Program header.
210-250	Initialize variables.
260-270	Display title screen.
280-330	Main menu.
340-390	Edit appointments menu.
400-470	Print routines menu.
480-670	Set time and date.
680-770	Edit appointment book.
780-860	Input a phone number from editing screen.
870-910	Edit phone numbers.
920-960	Sort phone numbers.
970-1040	Print a page to the screen.
1050-1330	Input routine.
1340-1370	Input a menu selection.
1380-1400	Print "Press ESCAPE to exit."
1410-1440	Input a day of the month.
1450-1470	Beep and pause.
1480-1590	Set markers.
1600-1630	Sort markers.
1640-1690	Print a page to the printer.
1700-1730	Print appointment book.
1740-1920	Print weekly summary.
1930-1990	Print phone numbers.
2000-2080	Print one day to the printer.
2090-2100	General print subroutines.
2110-2190	Save appointments.
2200-2300	Load appointments.
2310-2320	Error handling for save.
2330-2340	Error handling for load.
2350-2400	Input file name and drive (A or B).
2410-2450	Prompt to position paper.
2460-2490	Exit program.
2500-2510	Print another?
2520-2580	Input a (Y/N) response.
2590-2630	Find the first day of the month.
2640-2670	Find the day of the week based on the first day of the month.
2680-2700	Data for days of the week and months.

DESIGN FOCUS



DIRECTORY OF VARIABLES

Variables	Functions
T\$	Program title.
AP\$(,)	Appointment text.
MK(,)	Markers for each appointment.
PN\$(,)	Phone numbers.
MS(,)	Months of a year.
ND(,)	Number of days in each month.
M	Current month.
DW\$(,)	Days of the week.
DW	Current day of the week.
D	Day of the month.
FD	First day of the month.
YR	Year.
BT	Beginning time for appointments.
FL\$	File name.
k\$	Key input.
k	ASCII of key input.
P	Position of cursor in edit mode.
E	Edit mode flag.
CL\$	Cursor left characters.
I,J	Loop counters.
A,B,C	Utility variable.
F,G,H	Utility variable.
L,S\$,T	Utility variable.
X,Y	Utility variable.

REMARKS

The Commodore version of *Trig-Trix* uses redefined characters to display its triangles. Page 110 of the *C-64 Programmer's Reference Guide* has a procedure for moving the character set to enable this redefinition. The 10240 bytes of memory that this allows for your BASIC programs is not enough for even a moderate-sized program such as *Trig-Trix*.

This program uses a modified version of this procedure that takes only a couple of more POKES but greatly increases available memory. You begin in the same fashion as before—disabling interrupts, and switching off input and output (I/O). But instead of moving the character set to start at 12288, we move it to start at 32768.

After re-enabling the interrupts and I/O and making modifications to the character set, three POKES are necessary to get the C-64 operating system and the VIC-II in sync with your new character set:

POKE 53272,49: POKE 56576,PEEK(56576) AND 253: POKE 648,140.

The VIC-II video chip can access only 16K bytes of memory at a time—called a Bank. Normally the VIC-II operates in Bank 0 (locations 0 to a 16383). To maximize BASIC program area, we've moved the character set to Bank 2. By POKEing 53272 with 49, we tell the VIC-II where to look for our new character set, which is at the beginning of this 16K memory Bank at 32768. Next, we tell the VIC-II that it is to use Bank 2 by setting bit 1 and clearing bit 2 at location 56576 with an AND 253. The last POKE 648,140 informs the operating system to look in a new location for the screen memory, starting 32768 bytes higher than before in Bank 2 at 33792.

Because of all of these pointer changes, you should avoid hitting the (RUN/STOP) (RESTORE) keys, as they will undo some of our changes. If you should accidentally hit these keys, reset the computer (by turning it off and on again) and reload the program.

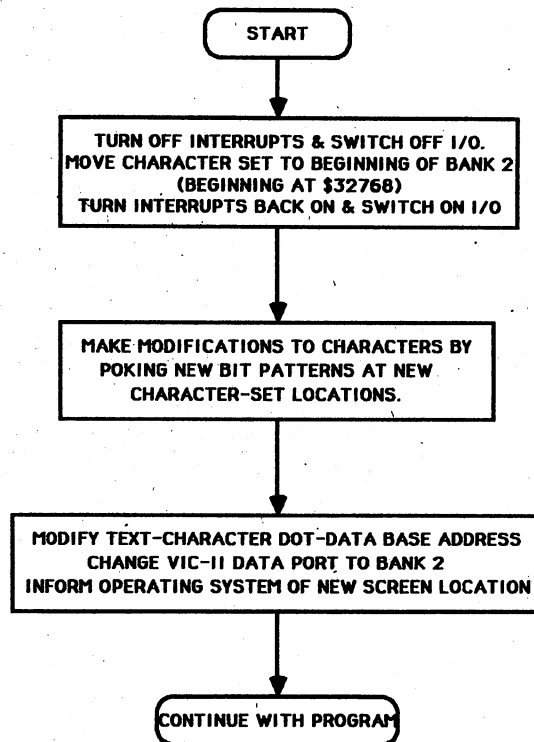
HCM Glossary terms: byte, interrupt, pointer.

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200-260	Initialize variables and print title.
270-340	Move and redefine characters.
350-410	Main menu.
420-480	Right Triangles menu.
490-560	Law of Sines menu.
570-630	Law of Cosines menu.
640-650	Link to draw screens.
660-770	Determine sides of right triangles.
780-1070	Determine angles of right triangles.
1080-1180	Common subroutines to all levels.
1190-1320	Law of Sines drill.
1330-1450	Law of Sines challenge.
1460-2360	Parser for input.
2370-2480	Law of Cosines drill.
2490-2600	Law of Cosines challenge.
2610-2670	Find sides equations.
2680	Calculating message.
2690-2730	Random functions, and center title.
2740	Place a character in parser tracker.
2750-2850	DATA for character definition.
2860-3320	PRINT statements for drawing graphics.
3330-3500	Input routines.

DESIGN FOCUS

DEFINING CHARACTERS WHILE MAXIMIZING MEMORY



DIRECTORY OF VARIABLES

Variables	Functions
A, A2	Values of angles
AS, A2S	Names of angles.
S1, S2, SL, SH	Values of sides.
S1\$, S2\$, S3\$	Names of sides.
TT\$	Menu title.
OP	Number of menu options.
OP\$	Temporary string for options.
PR	Number of problems.
PE	Constant for 3.14159.
P1\$(), P2\$()	Strings for possible problems.
P3\$()	String for possible problems.
DE, RA	Degree and radian functions.
AN, AS	Arcsine and Arccosine function.
RD	Rounding function.
B\$, BH\$, BH	Parser strings & pointers.
FMS	String used by parser.
S5	Parser status.
CP	Current position in input string.
DP	Decimal flag for parser.
JP	Syntax error flag for parser.
KS()	Control variables for parser.
SK\$(), SK()	Arrays to track parser input.
PCS	String holding parsed input.
INS	Input string for parser.
NS	Storage for numeric values.
XS, X1\$	Utility strings.
KS	Keyboard input string.
V, VR	Value of parsed input.
IV	Illegal value flag.
TR	Number of tries.
T, I, Q	Loop counters.
K, S, X, P, XC	Utility variables.
Z, E	Constants for 0 and 1.



REMARKS

Selecting "random" values for selected problems in *Trig-Trix* presents some interesting programming problems, but the TI BASIC `RESTORE` statement and the `SEG$` function solve them nicely. To illustrate how, we'll explain how the Determine Angles option gets the correct values to match the correct problem.

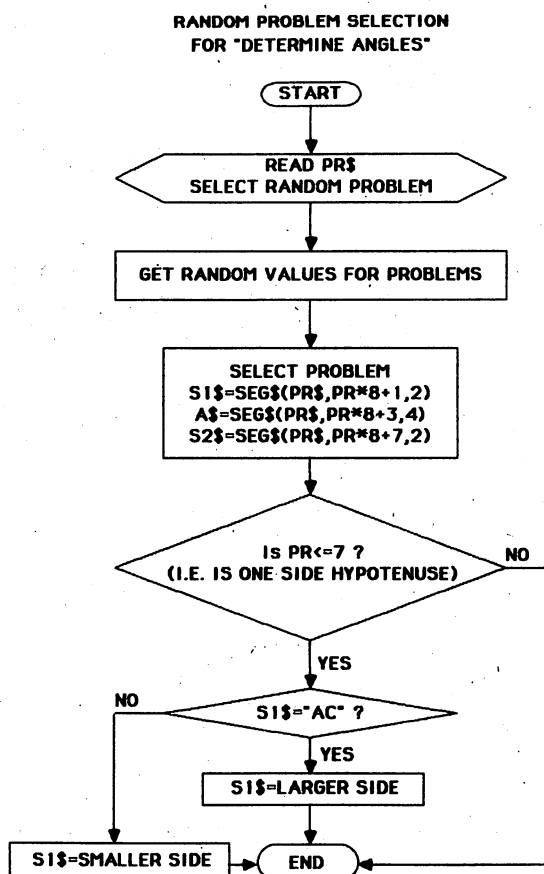
A string containing 96 letters is set up to hold all 12 different possible sets of 2-side and 1-angle problems. This string is located in the DATA statement in line 5550. Before entering this routine, a `RESTORE 5550` statement sets up a `READ` of this string in `PR$`. Upon entering the routine, a number from 0 to 11 is `RA`ndomly selected to choose a problem (line 1080).

To extract the 2 sides and an angle from the string, it is indexed in 8-character segments. The first 2 letters of any 8 compose a side, the next 4 letters are an angle (the lower case q was redefined as an angle symbol in the program), and the last 2 letters are the second side. For example, the first problem (`PR=0`) uses sides `AB` and `BC` to solve for angle `BAC`. By indexing into the string in multiples of 8, any one of the 12 problems can be defined. The Select problem box in the accompanying flow chart contains the TI BASIC code used to extract the names needed for any problem. You'll find this code in the program in lines 1590-1610.

Once the problem has been determined, the values are assigned. If `PR` is greater than 7, the problem is a `TAN`gent problem and either of the two sides can be the longer one. If `PR` is 7 or less, then the program finds the side (`S1$` or `S2$`) that is the hypotenuse ("AC") and assigns the larger length to that side before solving for the angle.

HCM Glossary terms: hypotenuse, string.

DESIGN FOCUS



LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200-350	Program initialization.
360-460	First menu.
470-550	Right Triangles menu.
560-630	Law of Sines menu.
640-710	Law of Cosines menu.
720-830	Display subroutines.
840-1070	Determine sides of right triangle.
1080-1460	Determine angles of right triangle.
1470-1580	Equations to determine angles.
1590-1620	Determine problem.
1630-1690	Miscellaneous messages.
1700-1830	Check for correct answer.
1840-2210	Law of Sines drill.
2220-2440	Law of Sines challenge.
2450-2490	Display routine.
2500-2520	Equation for Sines challenge.
2530-4060	Parser for expressions.
4070-4210	Input routines.
4220-4450	Law of Cosines drill.
4460-4500	Subroutines for getting problem.
4510-4620	Law of Cosines challenge.
4630-4710	Input subroutines.
4720-4870	Display routines.
4880-5000	Equations for determining sides.
5010-5020	Calculating message.
5030-5100	Random value routines.
5110-5240	Display and drawing routines.
5250-5490	Character redefinition and DATA.
5500-5570	Menu and problem DATA.

DIRECTORY OF VARIABLES

Variables	Functions
A, A2	Values of angles
A\$, A2\$	Names of angles.
S1, S2, SL, SH	Values of sides.
S1\$, S2\$, S3\$	Names of sides.
TTS	Menu title.
OP	Number of menu options.
PIE	Constant for 3.14159...
PR\$	String for possible problems.
PR	Problem number.
SC, NL	Location data for graphics.
BS, BH\$, BH	Parser strings & pointers.
FNS, OPS	Strings used by parser.
EQ\$	Function used by parser.
ST	Parser status.
CP	Current position in input string.
DP	Decimal flag for parser.
JP	Syntax error flag for parser.
SK1, SK2	Control variables for parser.
SK\$(), SK()	Arrays to track parser input.
PCS	String holding parsed input.
IN\$	Input string for parser.
N\$	Storage for input numeric values.
V, VR	Value of parsed input.
IV	Illegal-value flag.
TR	Number of tries.
T, I, K, S, X, P, CH\$	Utility variables.
Z, E, W	Constants for 0, 1, and 2.
DEG, RAD	Degree and radian functions.
ASN, ACS	Arcsine and arccosine functions.
RD	Rounding function.

PROGRAMMER'S WINDOW



Trig-Trix

REMARKS

When you enter an answer into the Apple version of *Trig-Trix*, it places each character into a separate element of the `ET$()` array. The program then evaluates the input and converts it into a numeric value. Here we are going to focus on how the numbers themselves are extracted from this array.

The flow chart demonstrates the process. Upon entry, the routine initializes the `DP` variable (a Decimal Point flag) to zero, and places the next character from `ET$(CP)` (the `CP` variable contains the Current Position in the input string) in `N$`. Next, we check to see whether this character is a period (for a decimal point) or a digit. If it is neither, the routine is terminated by a `RETURN`.

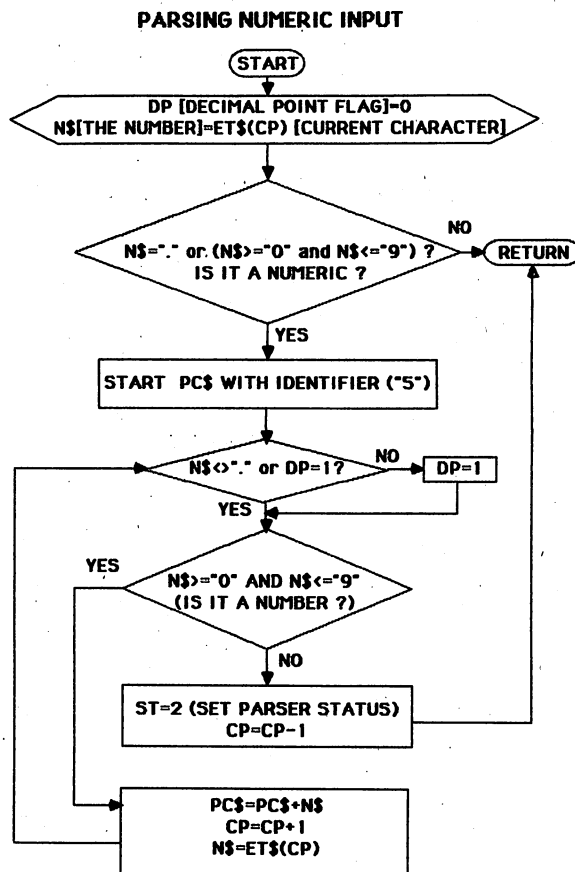
If the character is a digit or a period, then the `PC$` string is initialized to the character 5. This 5 will identify the rest of `PC$` as a numeric quantity when control is returned to the main part of the program. Later, as each quantity is evaluated, the 5 is stripped off and the `VALUE` of the rest of `PC$` is used for calculation.

We next enter the main loop of the routine. If `N$` is a digit, the logic passes through to the box at the bottom of the flow chart. Here `N$` is added onto `PC$`, `CP` is incremented, and `N$` is set equal to the next character in the `ET$()` array. Each character is similarly tested, and as long as it is a digit, the routine continues to add each digit to `PC$`.

As soon as a nondigit or nonperiod is encountered, `CP` is decremented so that the same `ET$(CP)` element of the array will be the next one accessed by the other routines in the program. When the first period is encountered, the `DP` flag is set, and the decimal point is added. The second period found, however, will cause the routine to be exited just as though a non-numeric character had been encountered.

HCM Glossary terms: array, initialize, legal input.

DESIGN FOCUS



LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200-250	Program initialization.
260-280	Main menu entry and exit.
290-390	Initialization subroutine.
400-550	Initialization DATA.
560-610	Right Triangles menus.
620-730	Find sides.
740-930	Find angles.
940-980	Law of Sines menus.
990-1030	Law of Sines drill.
1040-1130	Law of Sines challenge.
1140-1200	Law of Cosines menus.
1210-1340	Law of Cosines drill.
1350-1620	Law of Cosines challenge.
1630-1690	Common routines to all levels.
1700-2730	Parser for input.
2740-2900	Miscellaneous messages.
2910-3000	Equations to solve problems.
3010-3110	Extract problem from string.
3120-3160	Get random values.
3170-3180	Initialize character graphics.
3190-3270	Entry to print titles on hi-res screen.
3280-3340	Print to hi-res subroutine.
3350-3390	Character definition statements.
3400-3510	Menu subroutines.
3520-3550	Input subroutine.
3560-3600	Error-handling routine.

DIRECTORY OF VARIABLES

Variables

A, A2
 A\$, A2\$
 S1, S2, SL, SH
 S1\$, S2\$, S3\$
 TR
 PR
 CS(), SN()
 DG(), RA()
 CX(), CH()
 MES, MX
 BK\$, ES\$, BL\$
 B\$, BH\$, BH
 FM\$, OP\$
 ST
 CP
 DP
 N\$
 SK\$(), SK()
 SN()
 IN\$, FL\$, PC\$
 FV
 V, VR
 HC, HX(), VC, VX
 T, I, K, K1, K2
 S, X, P, CH\$
 IT, PX, ZH, X\$
 PI
 PW, OF, CH, HC
 PSS, PZ\$

Functions

Angle values.
 Angle names.
 Values of sides.
 Names of sides.
 Number of tries.
 Problem number.
 Arccosine and Arcsine functions.
 Degree and radian functions.
 Arrays for menus and title.
 Menu tracking and response.
 Character constants.
 Parser strings & pointers.
 Strings used by parser.
 Parser status.
 Current position in input string.
 Decimal flag for parser.
 Syntax-error flag for parser.
 Arrays to track parser input.
 Array to track input.
 Input strings for parser.
 Length of input string.
 Storage for numeric values.
 Text tracking arrays.
 Value of parsed input.
 Track cursor position.
 Utility variables.
 Utility variables.
 Counter variables.
 Variable for 3.14159...
 Graphics subroutine addresses.
 Variables for graphics.

REMARKS

IBM PC and PCjr BASIC's LOCATE statement turns the cursor on or off each time the statement is used. But this option is only active, however, when using SCREEN 0 (text mode). Thus, when prompting for input using INKEY\$ in a graphics mode (SCREEN 1 or SCREEN 2), the cursor remains off.

Trig-Trix solves this problem by creating a cursor with the LINE command, and then saving it in an array with the GET statement. Then, by using the PUT statement in its XOR mode (the default mode), we can place the cursor wherever we need it.

The accompanying flow chart details this procedure. During program initialization the cursor is created and placed in the CRS%() array (see line 1820). The rest of the flow chart is executed during the input routine in line 1550:

```
1550 K$ = "": YP = 8*(CRSRLIN-1) : XP = 8*(POS(XP)-1):
PUT (XP,YP),CRS%:
WHILE K$ = "": K$ = INKEY$:WEND:
PUT (XP,YP),CRS%
```

The line makes K\$ null, and finds the current cursor position using the CRSRLIN and POS() functions. Note that the argument in POS() is a "dummy" argument, so we just used the XP variable—any number could be used.

The first PUT statement places the solid box on the screen. Because the PUT statement is in XOR mode, if there is a character there, all pixels that are already "on" will be turned "off." This effectively places the cursor on top of the character just as we wish—causing the letter to show up in inverse.

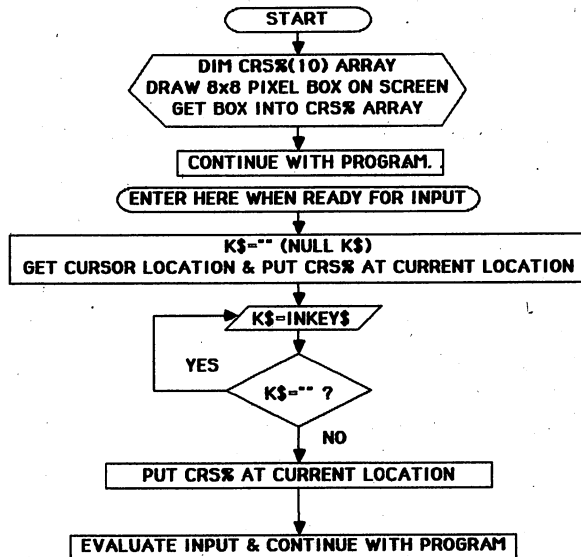
The WHILE loop waits until a key is pressed, then the second PUT statement restores the screen location to its original state before the first PUT statement was executed. For a detailed explanation on using character graphics, see the "Home Computer Tech Note" for IBM in this issue.

LISTING ANNOTATIONS

Line Nos.	
100-220	Program header.
230	Gosub initialization.
240-250	Main menu.
260-280	Right Triangles menu.
290-320	Determine sides of right triangles.
330-400	Determine angles of right triangles.
410-430	Law of Sines menu.
440-480	Law of Sines drill.
490-560	Law of Sines challenge.
570-590	Law of Cosines menu.
600-630	Law of Cosines drill.
640-690	Law of Cosines challenge.
700-800	Common subroutines to all levels.
810-820	Draw angle and draw enter symbol.
830-880	Draw graphics on screen subroutines.
890-1020	Equations for getting sides and angles.
1030-1040	Laws of Sines and Cosine equations.
1050-1090	Random-number generators.
1100-1510	Parser for input.
1520-1650	Input subroutine.
1660-1830	Initialization subroutine.
1840-1880	DATA statements for menus.
1890-1910	DATA for problem strings.

DESIGN FOCUS

CREATING A CURSOR WHEN USING SCREEN 1



DIRECTORY OF VARIABLES

Variables	Functions
A, A2	Values of angles
A\$, A2\$	Names of angles.
S1, S2, SL, SH	Values of sides.
S1\$, S2\$, S3\$	Names of sides.
TI\$	Menu title.
OP	Number of menu options.
PI	Constant for 3.14159...
PR	Problem number.
PR\$	String for possible problems.
BS, BH\$, BH	Strings & pointers for parser status.
FUNCT\$, OP\$	Strings used by parser.
FNEQ\$	Function used by parser.
ST	Parser status.
CP	Current position in input string.
DP	Decimal flag for parser.
JP	Syntax-error flag for parser.
SK1, SK2	Control variables for parser.
SK\$(), SK()	Arrays to track parser input.
PC\$	String holding parsed input.
IN\$	Input string for parser.
PT	Pointer for input.
MAXLEN	Length of input.
SELECT\$	Legal-input string.
SELECTLOS	Lowercase legal input.
ROW, COL	Row and column for input.
N\$	Storage for input numeric values.
V, VR	Value of parsed input.
TR	Number of tries.
CR\$, ESC\$, SP\$	ASCII character constants.
T, I, X, K, P, CH\$, K\$	Utility variables.
TI	Temporary timer for delay.
FNDEG, FNRAD	Degree and radian functions.
FNASN, FNACS	Arcsine and arccosine functions.
XP, YP	X and Y locations for graphics.
RESP\$	Response string.
ENT\$, ENT%	String & array for graphic.
ANG\$, ANG%	String & array for graphic.
CRS%	Array for cursor graphic.
RIGHTTR\$	Right-triangle graphic.
OBLIQUETR\$	Oblique-triangle graphic.
HEIGHT\$	Height-diagram graphic.
PARALLEL\$	Parallelogram graphic.

REMARKS

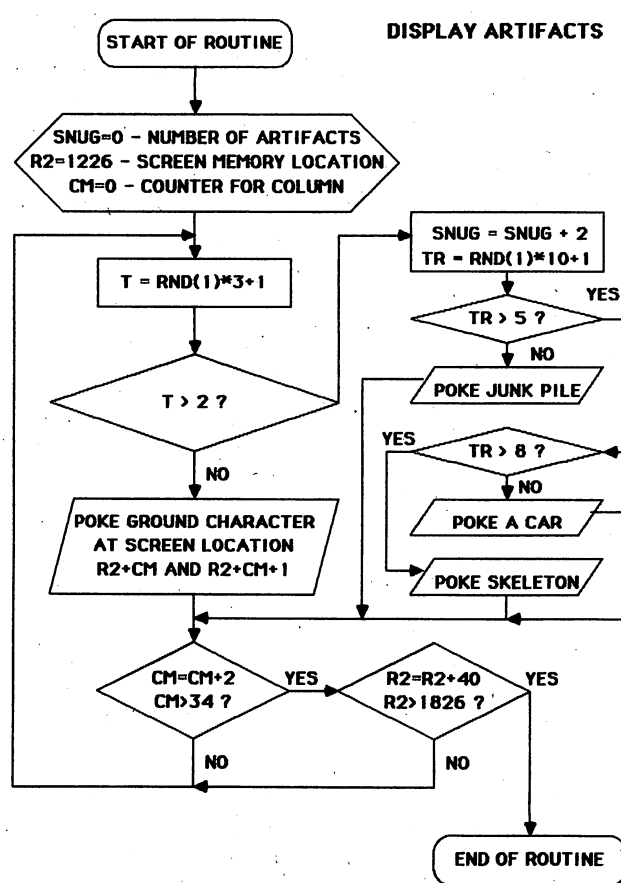
The Commodore 64 provides two methods of displaying characters on the screen: you can either print to it, or POKE to it. Although the PRINT statement is much quicker and more convenient, it doesn't offer as much flexibility as the POKE command. For instance, have you ever tried printing something to the very last screen location (row 40, column 25)? Well, if you try to use the PRINT statement, it won't work because the screen will always scroll. The only way to get a character at that last position is to POKE it there.

How, you may ask, do I POKE the screen? To begin with, this has nothing to do with jabbing your finger at the TV monitor. Instead, you use a method described in pages 60-66 of the *Commodore 64 User's Guide*. Basically, remember two things: screen memory and color memory. Memory locations 1024 to 2023 make up the screen memory; color memory is located at 55296 to 56295. Notice that both color and screen memory each consume 1000 bytes. This is because there are exactly 1000 character locations on the C-64's display screen (25 rows X 40 columns = 1000). Both of these memory locations are set up in a row-by-row format: The first 40 bytes contain the characters for the first screen row, the second 40 bytes contain the second screen row, and so on. Once you've picked a screen location, you POKE the screen value of a character (see Appendix E of the *Commodore 64 Users' Guide*) into screen memory and the color of that character into the corresponding location in color memory. For instance, to put a black A in the middle of the screen, enter the following:

POKE 1483,1: POKE 55755,0

The routine used to create a mining field on the C-64 version of *Archeodroid*—lines 540-660—uses this same method of poking characters onto the screen. For a more detailed look into this routine, see the accompanying Design Focus.

DESIGN FOCUS



DIRECTORY OF VARIABLES

Variables

DM0
JVS
K5,K15,K25
BLAST
R,C
C1,C5,C6
C7,C8,C9
CH
CL
DNUG
FL
FM
I,CM,LL
R2,T,T9
JV,FR
M1,M2
MINER
MNUG
PO
QK
R1
RR,RC
RS
S9
SCREEN
SNUG
SP
SX
TM
TR
TTL
A,B,W,X

Functions

Death song data.
Character from key scan.
Input strings.
Number of charges remaining.
Miner's row and column location.
Screen contents during blast.
Screen contents during blast.
Screen contents when moving.
Start of SID memory area 54272.
Artifacts held by dead robot.
Indicates miner is out of charges.
Flag that miner has moved.
Loop counters.
Loop counters.
Used to read keyboard and joystick.
Used when blasting.
Number of miners remaining.
Artifacts held by current miner.
Direction miner is facing.
Time remaining after depleting charges.
Blast center row.
Random row and column.
Character at location RR,RC.
Point to SID chip memory 54272.
Number of screens completed.
Number of artifacts left on screen.
Screen address pointer.
Random number for screen location.
Robot timer.
Determines types of artifacts.
Score.
Utility variables.

LISTING ANNOTATIONS

Line Nos.

100-190 Program header.
200-240 Title screen.
250-300 Initialize characters and sound.
310-420 Choose keyboard or joystick.
430-440 Set variables for beginning of game.
450-530 Display initial playing screen.
540-660 Display artifacts on screen.
670-690 Display mother ship.
700-810 Thunder and lightning.
820-870 Display robot.
880-890 Display score.
900-1150 Detonate blast.
1160-1420 Blast subroutines.
1430-1530 Return to ship with artifacts.
1540-1590 Replay option.
1600-1610 Count moves when robot is out of blasts.
1620-1700 Robot meets its maker.
1710-1880 Read keyboard and joystick.
1890-1950 Move up.
1960-2020 Move left.
2030-2090 Move down.
2100-2160 Move right.
2170-2180 Is obstacle a wall or a dead robot?
2190-2300 Add to artifacts.
2310-2380 Print robot.
2390-2550 Character graphics.
2560-2650 Death song routine and data.
2660-2740 Sound routines.



REMARKS

The TI-99/4A without Extended BASIC has some fairly good features—user definable characters, IF-THEN-ELSE structure, color graphics, and sound. In fact, the unexpanded TI does very well without the aid of added cartridges. But I know many of you bare-bones TI users wish you had **DISPLAY AT**. This one statement can really make a difference in your programming, for without any real control over where character strings are printed, screen formatting is next to impossible. In a game like *Archeodroid*, the ability to print the score and number of blasts at a certain location without scrolling the screen is vital. Because of this, a short subroutine is used to simulate the **DISPLAY AT** command. This subroutine, located in lines 2740-2790, is shown in the Design Focus.

The algorithm is really quite straight-forward. By using the **HCHAR** command, a character string is plotted onto the screen, one letter at a time. To cut down on code, a **FOR** loop is used to cycle through the string. As the loop is executed, the variable **A\$** is separated into single characters with the **SEG\$** function. These characters are placed into another variable, **B\$**, and then positioned onto the screen with **HCHAR**. Two other variables are passed to this subroutine in order to determine the actual screen location for the output. The column is placed in the **C1** variable, and the row is passed by **R1**.

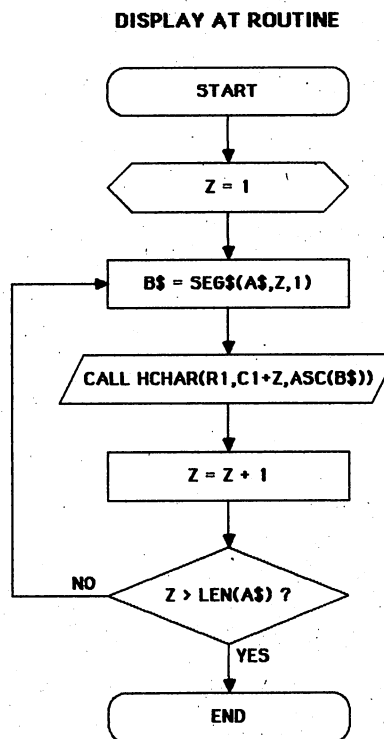
Using this subroutine in your own programs is easy. First, copy lines 2740-2790 from *Archeodroid* into your own program. Now, put the string that you want to print into **A\$**. For example, you could use **A\$ = "Home Computer Magazine"**. Next, to print this message in the middle of the screen, set the column to 5 (**C1 = 5**), the row to 12 (**R1 = 12**), and **GOSUB 2740**. (The line number of this subroutine may be changed to suit your needs.)

As useful as this routine is, it has one major drawback—it's slow. But, because *Archeodroid* relies on strategy, and not speed, the execution time of this routine is of little importance.

LISTING ANNOTATIONS

Line Nos.	
100-210	Program header.
220-230	Display title screen. Initialize program.
240-300	Display initial playing screen.
310-490	Display artifacts on the screen.
500-630	Display mother ship.
640-870	Display the remaining miners.
880-960	Scan keyboard.
970-990	Scan joystick.
1000-1080	Move left.
1090-1170	Move up.
1180-1260	Move right.
1270-1340	Move down.
1350-1530	Check if robot picked up an artifact.
1540-1600	Increment score.
1610-2110	Detonate blast.
2120-2480	Cave-in routine.
2490-2700	Robot returns to ship (ET phone home).
2710-2720	Start next screen.
2730-2790	Routine to print at any location.
2800-2870	End-of-the-game routine.
2880-3430	Out-of-charges routine.
3440-3480	Title screen.
3490-3500	Prompt for (ALPHALOCK) key.
3510-3730	Initialization routine.
3740	Sound data.
3750-3810	Character-graphics data.
3820	Initial-variable-value data.

DESIGN FOCUS



DIRECTORY OF VARIABLES

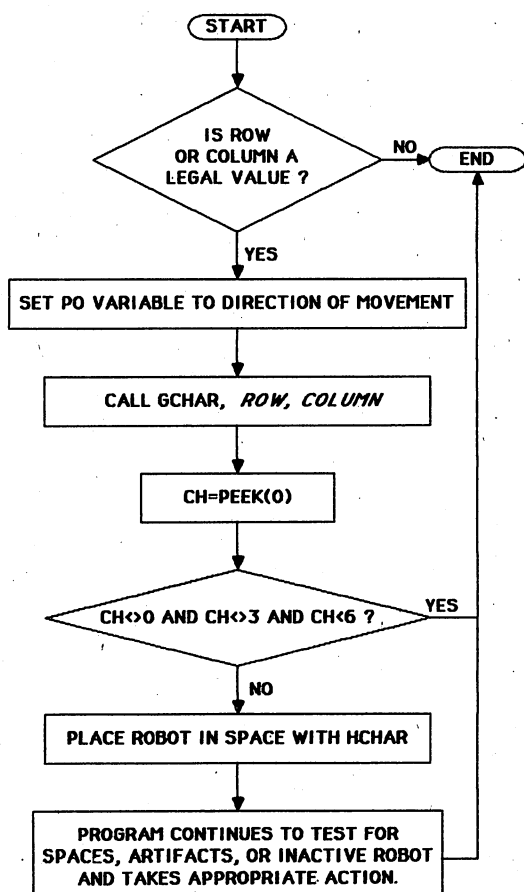
Variables	Functions
A\$	String containing output for print at routine.
BS	Used to separate A\$ into single characters.
SH1\$,SH\$	Spaceship graphics.
A,B	Used to read data.
C	Screen column that miner is on.
C1	Column position for print at routine.
C2	Loop counter for displaying game screen.
CH,CH1	Used to read characters from screen.
CH2,CH3	Used to read characters from screen.
CH4,CH5	Used to read characters from screen.
CHARGES	Number of charges left.
DNUG	Artifacts held by dead miner.
I	Loop counter.
K	ASCII of keyboard input.
KEY	ASCII of keyboard input.
LD	Loop counter for lightning sound.
MINER	Number of miners left alive.
MNUG	Artifacts held by current miner.
PO	Direction miner is facing.
R	Screen row that miner is on.
R1	Blast center row.
R2	Loop counter.
S	Status of keyboard input.
SCREEN	Number of screens completed.
SND	Loop counter for sound routine.
SNUG	Artifacts left on screen.
ST	Status of keyboard input.
T	Random number.
TB	Random number.
TD	Loop counter.
TIME	Robot timer.
TOTAL	Score.
X	Used to read the joystick.
Y	Used to read the joystick.
Z	Loop counter.



Archeodroid

DESIGN FOCUS

Using GCHAR to PEEK Hi-Res



REMARKS

The Apple version of *Archeodroid* relies heavily upon the Character Graphics routine published in *HCM* Vol. 5, No. 3. It is especially interesting how this program uses the **GCHAR** command of the utility to find what is on the screen. Normally, finding a specific bit pattern on the high-resolution (hi-res) screen takes some rather tricky math—but with the Character Graphics system it's easy.

A particularly good illustration of how this routine is used occurs when the program checks a robot's movement. Here the **GCHAR** routine checks not only whether the move is legal, but what artifact is on the screen at the given location. This saves the programmer from having to keep track of each screen location in a large array.

The accompanying Design Focus represents a general case for a robot's move. First, the Row (in the case of a horizontal move) or the Column (in the case of a vertical move) is checked against the limit for that move to be sure it is legal. If the move is illegal, then the routine jumps back. The **PO** variable is set to the direction of the move—1 for left, 2 for up, 3 for down, and 4 for right. Then the **GCHAR** routine is called. Here's the format for the **GCHAR** instruction:

CALL GCHAR, row, col

After the routine is called, zero page location 0 will contain the character number of the item at the location. We **PEEK** this location and place the value in the **CH** variable. Then the **CH** variable is used in a series of tests to determine what is at the location. If **CH** is not equal to 0 or 3, or is less than 6, the robot cannot legally move into the square, so we return to the main loop. Otherwise, there is either a blank (0), a cross marking an inactive robot (3), or an artifact (6-11). If you study lines 830-880, you will see how this information is used to pick up artifacts.

LISTING ANNOTATIONS

Line Nos.	
100-190	Program header.
200-210	Relocate program above hi-res graphics.
220-280	Initialize program.
290	Ask if using the keyboard or joystick.
300	Ask if user wants sound effects.
310	Turn on or off speaker.
320-410	Set up the screen.
420-490	Display the men.
500-510	Display scoring information.
520-640	Main game loop—get player input.
650-680	Move robot left.
690-720	Move robot up.
730-760	Move robot down.
770-790	Move robot right.
800-810	Replace robot with space character.
820-880	Pick up an artifact.
890-910	Set a charge.
920-960	Check what is being destroyed.
970-1020	Replace with blanks.
1030-1050	Legal-blast-check subroutine.
1060-1180	Cave-in routine.
1190-1200	Robot is back at ship.
1210-1240	Update score.
1250-1290	End of game—play again?
1300-1330	Spaceship table data.
1340-1410	Data for machine-language routines.

DIRECTORY OF VARIABLES

Variables	Functions
AS	Used as input variable.
KJS	Keyboard or joystick flag.
SNS	Sound effects flag.
XS	Used to read DATA statements.
A	Used to PEEK the keyboard.
C	Screen-column indicator.
CI	Column of blast center.
CG	Number of charges remaining.
CH	Character returned by GCHAR.
DNUG	Artifacts on dead robot.
FLAG	Indicates when robot is out of charges.
GCHAR	Screen PEEK routine, 2051.
HCHAR	Screen POKE routine, 2054.
I	Loop counter.
K	Utility, loop counter, etc.
MINER	Number of miners left.
MNUG	Number of artifacts on miner.
NOISE	Sound routine, 2048.
P	Used for reading DATA.
PO	Direction miner is facing.
R	Screen-row indicator.
RI	Row of blast center.
SCREEN	Number of screens completed.
SNUG	Number of artifacts on the screen.
SPKR	Speaker address.
TIME	Time remaining to return to ship.
TTL	Score.
X, XX, YY	Loop counters.

REMARKS

When writing a game such as the IBM PC and PCjr version of *Archeodroid*, it is desirable to make both the keyboard and the joystick responses quick-acting, but similar to each other. When both routines return identical values for the same command, then such response is assured. This is how the IBM version is designed.

Before we look at how the program makes the joystick routine return identical values to the keyboard, let's look at the keyboard input itself. In *Archeodroid*, the arrow keys are used to control the robots' movements. Using the `INKEY$` function to scan the keyboard returns string values that are not 1, but 2 characters in length. The first character is an ASCII zero; the second is a different character. By first testing the length of a string returned by the `INKEY$` function, we determine whether one of these keys could have been pressed. If the `LENGTH` is 2, we then check the second character against the following list to see whether it is one of the following:

K Left
M Right
H Up
P Down

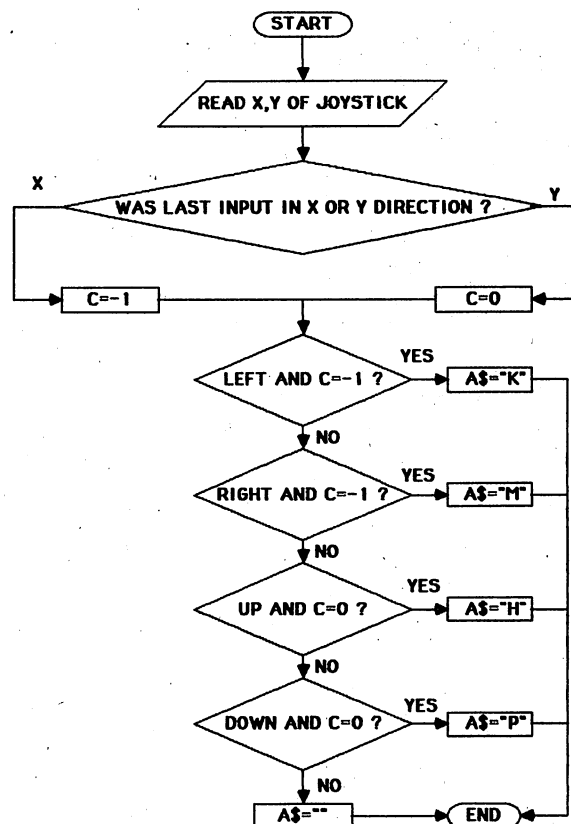
If it is none of the above, we know that no movement key was pressed, and check to see whether the Spacebar was pressed, indicating that the blaster was fired.

To approximate the timing of this keyboard routine, we actually check the joystick twice for each move. If the joystick returns two of the same type of motions (either horizontal or vertical) in a row, then the `C` flag is set for that kind of motion. The code that does this is in line 790. If this check is not included, joystick response is so rapid that the robots tend to move an extra space before the motion can be halted.

The accompanying flow chart shows how this `C` flag is set, and the appropriate character returned from the routine. The condition of the `C` flag is used to screen the joystick input, and only when the direction of movement indicated is on the same axis (x or y) will `A$` be set to the character indicating that direction of movement. A similar method makes the fire button respond identically to the Spacebar in keyboard mode.

DESIGN FOCUS

SEEING JOYSTICK AS KEYBOARD INPUT



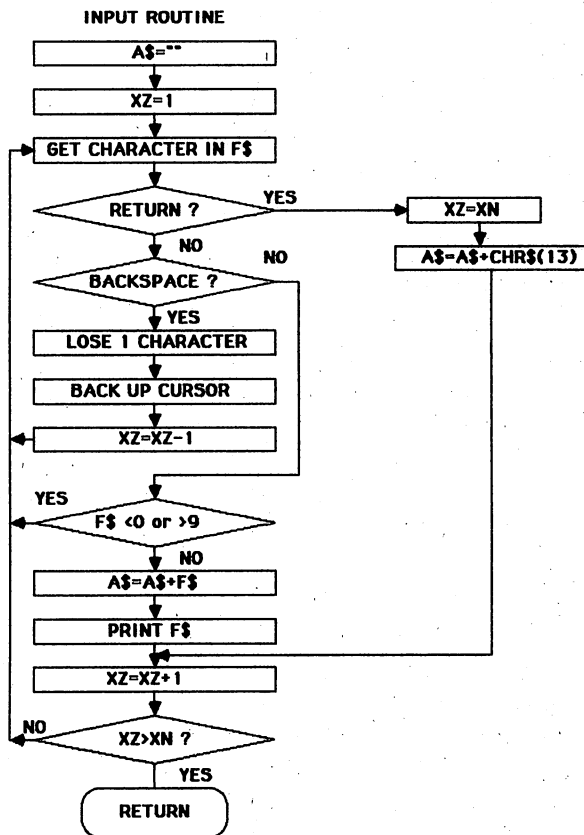
DIRECTORY OF VARIABLES

Variables	Functions
A\$,B\$	Key-input variables.
A,B,AM, BM	Joystick-input variables.
ART()	Array of what is on the screen.
BIGFILL	Graphic array to fill screen.
BLNK	Graphic array for blank area.
BX,BY	Temporary storage for screen coordinates.
C	Last joystick input flag.
CAR	Graphic array for car body.
CHG	Number of charges.
CIX,CIY	Coordinates of random cave-ins.
CNT	Time counter when out of charges.
GARB	Graphic array for garbage.
KJ	Keyboard or joystick flag.
MNUG	Artifact points collected.
NROB	Number of robots left.
ONR	Temporary storage of number of robots.
PCHARGE	Number of charges.
PDIR, DIR	Direction robot is pointing.
R	Variable for random number.
RA	Storage for artifacts held by robot caught in cave-in.
ROB	Graphic array for robot.
ROBX,ROBY	Current X,Y coordinate of robot.
SCN	Value of article at current location.
SCN1,SCN2	Values of article at location to be blasted.
SCN3,SCN4	Values of article at location to be blasted.
SCN5	Value of article at location to be blasted.
SCORE	Player's score.
SKEL	Graphic array for skeletons.
SNUG	Total number of artifacts on screen.
TD	Delay loop counter.
X,Y,Z	Loop counter variables

LISTING ANNOTATIONS

Line Nos.	Program header.
100-240	Title Screen.
250	Program initialization.
260	Choose keyboard or joystick.
270-280	Initialize for new game.
290	Main control loop.
300-330	Get graphics arrays.
340-410	Paint playing screen.
420-530	Beam down robots.
540-650	Robot control routine.
660-740	Scan keyboard.
750-780	Scan joystick.
790-810	Set robot direction.
820-850	Move robot up.
860-880	Move robot left.
890-910	Move robot right.
920-940	Move robot down.
950-970	Figure affect of robot's movement.
980-1020	Blaster routine.
1030-1220	Robot returns to ship.
1230-1300	Robot-out-of-power routine.
1310-1400	End-game-and-replay routine.
1410-1450	

DESIGN FOCUS



REMARKS

A common problem with writing BASIC programs on the Commodore 64 is the programmer's inability to control information being entered into the computer through the keyboard.

The INPUT statement has many drawbacks: it allows users to enter graphics characters, cursor control characters, and even system control characters such as Clear Screen, Home, and Backspace. Putting these characters into a string that is to be displayed can have disastrous effects, especially if you are concerned about screen formatting.

The only solution to this dilemma in *Mine Over Matter* was to write a separate input routine; one that would allow only the user to enter numbers and would limit the entry to a certain number of characters to prevent wrapping. You can see such a routine in the listing at lines 3290 through 3360.

The routine used here has been diagrammed in the Design Focus. As you can see, the only built-in editing feature is the backspace key—for short entries it is often adequate. A check is done in the routine to limit entry to the numbers 0 to 9 and to a maximum length, which is passed in the variable XN.

DIRECTORY OF VARIABLES

Variables

M\$()
 PL\$()
 B()
 MP()
 T()
 U()
 V()
 W()
 A\$
 DP\$, DW\$, DZ\$
 E\$
 F\$
 K\$, S\$
 B
 BP, BZ
 C
 CL
 D
 E
 ER
 F
 H
 HR
 HX
 HY
 I
 J
 K
 L, LL
 LX
 M
 N
 P
 PS
 QM
 QS
 R
 RW
 S
 SA, SB, SC
 SP
 W
 XN
 XS
 T, X, XP
 XZ, Z, Z1

Functions

Messages for mine status.
 Players' names.
 Coordinates for ripple centers.
 Mine data and status.
 Players' totals.
 ASCII character to change color.
 Contains the survey parameters.
 POKE code to change color.
 Return value from input routine.
 Format strings.
 Error message from error routine.
 Character returned from pressing key.
 Used in file-name-input routine.
 Temporary in getting V() values.
 Loops to display mines.
 Return screen contents.
 Pointer to SID chip.
 Cost of an operation.
 Length of game in years.
 Error condition from disk.
 Value of key pressed.
 Horizontal screen position.
 Horizontal sprite MSB.
 Horizontal sprite coordinate.
 Vertical sprite coordinate.
 Vertical screen position.
 Number of players.
 ASCII of key pressed.
 Utility.
 Sprite horizontal position.
 Value of one barrel of "Yellow Cake."
 Utility.
 Current player.
 Used to restart a game from storage.
 Mine status.
 Screen memory location.
 Starting year for the game.
 Row pointer for clearing the screen.
 Production rate in barrels/year.
 Load/Save loop counters.
 Pointer to sprite memory area.
 Temporary in getting V() values.
 Input routine's maximum characters.
 Used to restart a game from storage.
 Loop counters.
 Loop counters.

LISTING ANNOTATIONS

Line Nos.	
100-200	Program header.
210-300	Initialize program variables.
310-340	Load-previous-game option.
350-490	Get options.
500-520	Set up for a new game.
530-580	Main control loop.
590-660	Players' main menu.
670-1020	End-of-the-year routine.
1030-1220	Print a report to the screen.
1230-1330	Production main menu.
1340-1490	Production—set production rate.
1500-1630	Production—start reclamation.
1640-1880	Production—start construction.
1890-2280	Survey routine.
2290-2320	Routine to scan the keyboard.
2330-2450	End-of-game routine.
2460-2480	Calculate survey parameters.
2490-2520	Build format strings.
2530-2580	Sprite data.
2590-2610	Message data.
2620-2630	Machine language data for clearing text portion of the screen.
2640-2880	Save this game.
2890-3110	Load previous game.
3120-3280	Load/Save screen.
3290-3360	Input routine for numeric data.
3370-3380	Plot cursor.
3390-3490	Input routine for file name.
3500-3520	Read disk status.



Mine Over Matter

REMARKS

We used just about every trick in the book to allow *Mine Over Matter* to run on the unexpanded TI-99/4A home computer. The TI-99/4A is a great computer, but in "minimum-memory configuration" (TI Extended BASIC with one or more disk drives attached, and no memory expansion), it severely limits the size of program that can be run. If you do the **SIZE** command with such a configuration, you will find that you get to use only 11,840 bytes of the 99/4A's 16K of memory. This includes program space, variable space, and any scratch-pad memory that the interpreter requires while the program is running. However, a BASIC program may be compressed to run under limited memory.

By converting commonly used constants, such as the numbers 0 through 9, into single-letter variables, a big savings is realized. Each constant in your program eats up 1 byte of memory, plus the number of characters in the constant (e.g., the constant 1 requires 2 bytes, while 100 requires 4 bytes). A variable uses up 9 bytes of memory when it is first defined. Then, each time it is placed within the program as in-line code, it uses only 1 more byte. Of course, you must use a 1-character variable (such as A) so the variable takes up only one byte. Thus, we can gain substantial memory savings if we replace a constant used several times in a program with a variable.

Another method of conserving memory has its trade-offs with execution speed. You can store integer numbers with values between 0 and 255 as ASCII characters in a string. In doing this you only use up 1 byte of memory for each number (character) in the string, plus 4 bytes of overhead for the whole string. A string which contains 10 integer values would use $10 + 4 = 14$ bytes of memory. However, the string manipulations necessary to insert or extract a number tend to slow a program down. When you want to make a program fit, trade-offs such as this are often necessary.

LISTING ANNOTATIONS

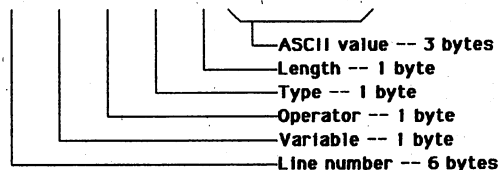
Line Nos.	
100-190	Program header.
200-270	Initialize the program and display title.
280-300	Get number of players and their names.
310-340	Get game-length and starting year.
350	Set up for start of a game.
360-410	Main control and sequencing loop.
420-470	Main menu for each player.
480-640	End-of-the-year routine.
650-670	Display all players' status screens.
680-810	Print report for one mine on the screen.
820-900	Production main menu.
910-980	Production option—set production rate for a mine.
990-1030	Production option—start the reclamation phase for a mine.
1040-1130	Production option—Start construction.
1140-1360	Survey routine. Select a survey site and determine whether to start operation on it or not.
1370-1400	Calculate the production rate.
1410	Calculate reclamation costs.
1420	Scan the keyboard.
1430	Time-delay routine.
1440-1460	End-of-the-game routine.
1470-1480	IMAGE format for the USING clause
1490-1510	Program data.
1520-1540	Subroutine to return the four survey parameters. This routine uses the "ripple" algorithm discussed in "Algorithm-A-Tricks" in this issue of HCM.

DESIGN FOCUS

CONSTANT TO VARIABLE CONVERSION

Internal format for the BASIC line: 100 A=123

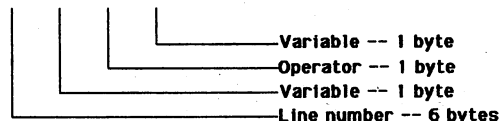
Line Number	A	=	T	3	49	50	51
-------------	---	---	---	---	----	----	----



Total = 13 bytes

Internal format for the BASIC line: 100 A=B

Line Number	A	=	B
-------------	---	---	---



Total = 9 bytes

Memory saved from 1 conversion = 4 bytes

DIRECTORY OF VARIABLES

Variables	Functions
PLS()	Players' names.
MP(,)	Mine information.
T()	Players' totals.
V()	Survey results for depth of mine, quality of ore, environmental impact factor, and quantity of ore.
AS\$	Utility string for inputs and character definitions.
BS\$	Contains the center coordinates for the four survey parameters.
CS\$	Utility string for character definitions.
A	Constant for 1.
B	Constant for 2.
C	Character returned from looking at the screen with the GCHAR statement.
D	Cost of an operation.
E	Length of the game in years.
F	Key returned from scanning keyboard.
G	Constant for 0.
H	Horizontal screen coordinate for a mine.
I	Vertical screen coordinate for a mine.
J	Number of players.
K	Constant for 3.
L	Constant for 4.
M	Value of one barrel of "Yellow Cake" ore.
N	Utility variable. Loop counters, etc.
P	Current player number.
Q	Constant for 5.
R	Current game year.
S	Current mine's production in barrels per year.
X	Game-turn counter.
Z	Utility variable. Loop counter.



REMARKS

Applesoft BASIC has no built-in functions or utilities, such as PRINT USING, for formatting numeric output. When printing large dollar amounts, it is traditional to place commas in the number every three digits in order to make it easier to read. A number like 1653427645 is much more difficult to read than 1,653,427,645. For this reason, we included a routine in lines 1700-1730 which formats a number by placing a comma between every 3 digits.

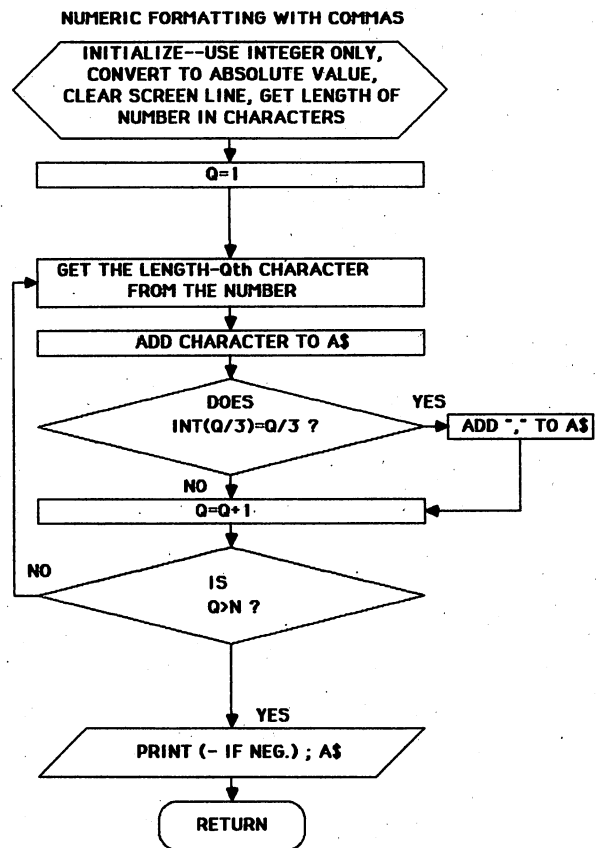
To do this, we first need to truncate the number into an integer. The sign (+ -) must be preserved by saving the current working value before converting it to an absolute value (positive number). The number is stored in the C variable. Next, we convert the number to a positive value in order to simplify the comma-insertion loop. We also need to convert the number to a string (N\$) before inserting the commas so that its individual characters can be manipulated.

Next, the screen is prepared for formatted numbers by locating the cursor and clearing the screen line. The length of N\$ is placed in the N variable, which is used as the terminating value of the comma-insertion loop.

We now use this loop to search through each character stored in N\$, starting at the right side. The variable Q is used as a counter in this routine. On each pass through the loop, a character is pulled from N\$ and tacked onto the front of A\$ (newchar + A\$). By using the test IF INT(Q/3) = Q/3 we can easily identify every third character. In this case we add a comma to A\$, otherwise we continue the loop until the entire string has been converted.

When complete, the formatted number will be in A\$ (missing any sign which may have been stripped off earlier). If the number is negative, the sign is printed along with A\$ when the number is output.

DESIGN FOCUS



LISTING ANNOTATIONS

Line Nos.	
100-200	Program header.
210-230	Protect hi-res screen.
240-260	Print title screen.
260	Set ProDOS flag.
270	Initialize character graphics.
280	First menu.
290-300	Get initial input.
310-360	Get number of players and names.
370-390	Get number of years.
400-410	Get current year.
420-430	Print first game screen.
440-500	Main control loop.
510-600	Players' main menu.
610-850	End-of-the-year routine.
860-980	Print reports.
990-1070	Production main menu.
1080-1180	Production--set production rate.
1190-1240	Production--start reclamation.
1250-1360	Production--start construction.
1370-1610	Survey routine.
1620-1660	Calculate production rate.
1670-1680	Calculate reclamation fee.
1690	Time-delay loop.
1700-1740	Format output.
1750-1790	Exit-program routine.
1800-2110	Load-and-save routines.
2120-2230	Error routine.
2240-2280	Mine-status display-message data.
2290-2360	Machine-language-routine data.

DIRECTORY OF VARIABLES

Variables	Functions
B%()	Coordinates of the centers of each of the four ripples used in calculating the survey results.
MP(,)	Status on each player's mines.
MSG\$()	Message for each mine's status.
PL\$()	Players' names.
S%()	Contents of mine fields for redisplaying graphics.
T(,)	Totals for each player.
V()	Survey results for one mine.
C	Character read from the screen.
E	Length of game in years.
F	Key pressed when keyboard is scanned.
FLAG	Fatal-error flag.
HCHAR	Address of HCHAR routine 2048.
J	Number of players.
K	Used in locating the player's mine characters for display.
M	Value for one barrel of "Yellow Cake."
A\$,D,N	Utility variables.
P	Current player.
PD	ProDOS flag.
Q	Loop counter for number formatting.
N\$	Used for number formatting.
R	Year at the start of the game.
S	Production rate for one mine per year.
S1,S2	Used for restarting the game from a disk file.
H	Horizontal screen position.
I	Vertical screen position.
V1	Vertical screen coordinate.
V2	Horizontal screen coordinate.
Y1,Y2	Temporary variables used in calculating the survey results.
X,Z,Z1	Loop counters.

REMARKS

After running the IBM PC/PCjr version of *Mine Over Matter*, you may notice that two display screens are used. One screen is used for the overhead view of the survey area, while the other is the menu screen. The tricky part about doing this is that the contents of each screen must not be destroyed when the screens are switched in and out. Fear not, because all of this can be, and is, accomplished through the use of the **SCREEN** statement. The **SCREEN** command allows a program to select different pages for one to work with. Each page, a separate screen of text, can be edited and displayed.

In 40-column text mode, you can have up to 8 different pages. In this program, it's sufficient to use just 2 pages—one for the colorful display of the mines and the survey locations, and the other for printing reports and getting information from the players.

You can specify four parameters with the **SCREEN** statement. These parameters allow you to change the page that is being displayed and redirect the output of the **PRINT** statements to each different page. It is possible to switch from one page to another within the blink of an eye—even when the new page needs to be completely redisplayed with different text. When you first start the transfer to the new page use: **SCREEN 0,0,1,0**

This allows the program to print text to the new page while still displaying the old page. When you are finished outputting to the new page, use this command to display it: **SCREEN 0,0,1,1**

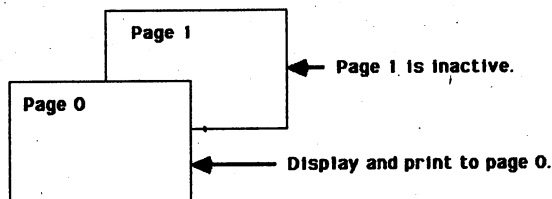
The illustration here shows how the **SCREEN** statement can alter what you and the computer see as the output page. Notice that the first parameter determines the screen's mode. The mode can be any number between 0 and 6—0 is a text screen and can be either 40 or 80 columns in width. All modes above 0 indicate a hi-res screen. With a little planning and a bit of imagination, a lot can be done with the **SCREEN** command—try it.

LISTING ANNOTATIONS

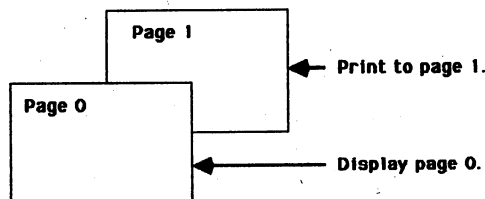
Line Nos.	
100-220	Program header.
230-280	Initialization.
290	Display title screen.
300-320	Option to start new or load old game.
330-340	Enter the number of players.
350-380	Select game length and starting year.
390-430	Main control loop.
440-500	Players' main menu.
510-730	Survey routine.
740-810	Production routine. Main menu.
820-910	Set production rate.
920-990	Start reclamation.
1000-1100	Start construction.
1110-1200	Print a report for one mine on the screen.
1210-1270	Load-game routine.
1280-1360	Save-game routine.
1370-1540	End-of-year routine.
1550-1620	End-of-game routine.
1630-1650	Option to play again.
1660-1730	Numeric-input routine.
1740-1810	Alpha-character-input routine.
1820-1850	Scan for a key press.
1860	Display prompt to continue & scan keyboard.
1870-1880	Mine-status-report routine.
1890	Routine to calculate the quality of ore, depth of a mine, environmental impact, and quantity of ore for a mine site.
1900	Calculate the production rate for a mine in barrels per year.
1910-1940	Program data.

DESIGN FOCUS

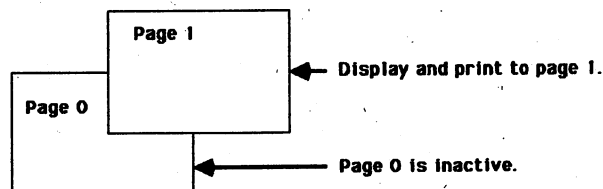
SCREEN 0,0,0,0 produces:



SCREEN 0,0,1,0 produces:



SCREEN 0,0,1,1 produces:



SCREEN 0,0,0,0

- Visual page (display this page)
- Active page (PRINT to this page)
- Burst (enable/disable color)
- Mode (indicates screen mode; 0=text)

DIRECTORY OF VARIABLES

Variables	Functions
MP(,)	Status on each of the players' mines.
PCOL()	The color for each of the players.
PL\$()	Players' names.
STAT\$()	Mine-status display messages.
T(,)	Players' totals.
AS	Used to return the results from input.
F1\$	Display format #1 (text and dollars).
F2\$	Display format #2 (text and value).
F3\$	Display format #3 (text and text)
FILE\$	File name.
KS	Value of key scanned.
CCOL	The color of a character on the screen.
CHAR	The ASCII of a character on the screen.
COST	Cost of an operation.
CP	Character position of cursor.
CREC	Cost of reclamation.
CURX,CURY	Coordinates of a mine.
DIR	Indicates direction of cursor.
DLN	Starting line for the status report.
E	Length of game in years.
IL	Maximum length for the input routines.
IX,IV	Starting position for the input routines.
J	Number of players.
M	Value of one barrel of "Yellow Cake."
P	Indicates the current player.
R	Year at the start of the game.
S	Mine's production rate in barrels per year.
SC	Current screen number.
YEAR	Current game year.
TD,Y,Z	Utility variable and loop counter.

LOGO SYSTEMS LOGO — IBM PC & PCjr

[illegible][illegible]

HOW

TERRAPIN LOGO — C-64

[illegible][illegible]

E I F M A N L O : M J - H J A R E
A V E . N E I T H ? O F B E S = T H E N K P U
. I N : H E R E F I E S I E S I E S I E S I E S
M I N E : M I N E E 1 1 1 1 1 1 1 1

Continued

[illegible][illegible][illegible]

T	O	L	C	A	K	E	S	E	M	A	K	E	"	S	E	?
I	N	F	O	R	M	E	E	C	A	M	E	T	H	E	F	O
L	L	I	O	O	O	O	O	I	S	S	:P	R	E	N	O	T
I	N	G	P	D	D	D	D	D	0	S	M	S	I	R	E	T
E	N	D						=								

[illegible][illegible][illegible][illegible][illegible][illegible]

T	L	O	K	A	:	L	S	:	N	P	
I	E	M	P	T	?	L	S	T	S	T	=
F	A	P	R	F	F	R	F	L	T	L	
I	P	R	E	S	? F	1					
:N	MAK	:	PRE	:	LIST	:	LIST	:	LIST	:	ST
END	LOOK	:	KAT	:	LIST	:	LIST	:	LIST	:	ST

TO I CAL " SEE? MAKE " SEE?
LOO C PR [YOU ARE CARRYING:]
LO OK AT : ITEMS (1)
N EN D : SEE? = 0 PR [NO TH I

[illegible][illegible][illegible][illegible]

HOM

RUN-DAY-VIEW

APPLE II Family

```

100 REM
110 REM * RUN-DAY-VIEW *
120 REM *****
130 REM COPYRIGHT 1985
140 REM EMERALD VALLEY PUBLISHING CO
150 REM BY RANDY THOMPSON
160 REM AND THE HCM STAFF
170 REM HOME COMPUTER MAGAZINE
180 REM VERSION 5.4.1
190 REM APPLE II FAMILY APPLESOFT
200 PD = 0: IF PEEK (48905) = 76 AND
    PEEK (48911) = 0 THEN PD = 1
210 ONERR GOTO 3820
220 GOSUB 330
230 GOSUB 640
240 IF CH (1) = 6 OR CH (1) = 0 THEN 270
250 ON CH (1) GOSUB 690,1490,1990,3210,3
    450
260 GOTO 230
270 HOME
280 VTAB 10: HTAB 1: PRINT "ARE YOU SUR
    E YOU WISH TO QUIT? (Y/N)"
290 VC = 10: HC = 32: GOSUB 3760
300 IF INS < > "Y" THEN 230
310 END
320 REM INITIALIZATION
330 HOME: VTAB 10: HTAB 5: PRINT "INIT
    IALIZING. PLEASE WAIT. . ."

```

```

340 CR$ = "CHR$(13):ESC$ = CHR$(27)"
350 LF$ = "CHR$(8):RT$ = CHR$(21):AS$ = CHR$(25)"
360 UP$ = "CHR$(Z):BL$ = CHR$(J) - 64:DN$ = CHR$(Q) - 64"
$ (ASC ("P")) - 64:PG$ = CHR$(ASC ("A")) - 64
$ (ASC ("C")) - 64:PF$ = CHR$(ASC ("F")) - 64
370 QAS$ = CHR$(ASC ("Z") - 64)
380 M$ = 0:YR$ = 0:BT$ = 9
390 DIM ET$(30): DIM LOC(23)
400 DIM CH(10): DIM CX(3)
410 DIM LIM(4): DIM WA(28,2)
420 DIM MENU$(30): DIM MNS(12): DIM DWS(7): DIM ND(12)
430 FOR IT = 1 TO 14: READ MENU$(IT): N
440 FOR IT = 0 TO 23: READ LOC(IT): NEX
450 BT$ = " "
460 DIM AP$(31,18): DIM MK(31,18): FOR IT = 1 TO 31: FOR JI = 1 TO 18: AP$(IT,JI) = BK$:MK(IT,JI) = 0: NEXT IT: NEXT JI
470 DIM PNs(18): FOR IT = 1 TO 18: PNs(IT) = BK$: NEXT IT

```

Continued.


```

480 FOR IT = 1 TO 12: READ MNS(IT): REA
D ND(IT): NEXT FOR IT = 1 TO 7: R
490 FOR IT = 1 TO 4: READ LIM(IT): NEXT
500 RETURN
510 DATA "EDIT APPOINTMENTS", "EDIT PHO
NE NUMBERS", "PRINT ROUTINES", "RI
520 DATA "LOAD APPOINTMENT FILE", "SAVE
APPOINTMENT FILE", "EXIT PROGRAM", "PRI
530 DATA "SET DATE AND TIME", "EDIT AP
POINTMENT BOOK", "RETURN TO MAIN MEN
U"
540 DATA "PRINT ONE PAGE FROM BOOK", "
PRINT APPOINTMENT BOOK", "PRINT WEEK
LY SUMMARY"
550 DATA "PRINT PHONE NUMBERS", "RETURN
TO MAIN MENU"
560 DATA 1024, 1152, 1280, 1408, 1536, 1664
, 1792, 1920
570 DATA 1064, 1192, 1320, 1448, 1576, 1704
, 1832, 1960, 1104, 1232, 1360, 1488, 1616
, 1744, 1872, 2000
580 DATA "JANUARY", 31, "FEBRUARY", 28, "M
ARCH", 31, "APRIL", 30, "MAY", 31, "JUNE"
, 30
590 DATA "JULY", 31, "AUGUST", 31, "SEPTEM
BER", 30, "OCTOBER", 31, "NOVEMBER", 30,
"DECEMBER", 31
600 DATA "SUNDAY", "MONDAY", "TUESDAY", "
WEDNESDAY", "THURSDAY", "FRIDAY", "SAT
URDAY"
610 DATA 4, 21, 8, 29
620 RETURN
630 REM MAIN MENU
640 HOME: INVERSE: FOR IT = 1 TO 3: V
TAB IT: HTAB 10: PRINT SPC(20): N
EXT: NORMAL
650 INVERSE: VTAB 2: HTAB 14: PRINT "R
UN-DAY-VIEW": NORMAL
660 CX(1) = 1: CX(2) = 6: CX(3) = 1: GOSU
B 1700
670 RETURN
680 REM EDIT APPOINTMENTS
690 HOME: VTAB 3: HTAB 10: PRINT "EDIT
APPOINTMENTS"
700 CX(1) = 2: CX(2) = 3: CX(3) = 7: GOSU
B 1700
710 IF CH(2) = 0 OR CH(2) = 3 THEN RET
URN
720 ON CH(2) GOSUB 750, 990: GOTO 690
730 RETURN
740 REM SET DATE AND TIME
750 IF M = 0 THEN M = 1: YR = 1985: BT =
9: GOTO 840
760 HOME: VTAB 3: HTAB 9: PRINT "SET D
ATE AND TIME"
770 VTAB 10: HTAB 1: PRINT "ERASE ALL C
URRENT APPOINTMENTS? (Y/N)"
780 VC = 10: HC = 33: GOSUB 3760: IF IN$
< "Y" AND IN$ < "N" THEN 780
790 IF IN$ = "N" THEN RETURN
800 VTAB 16: HTAB 1: PRINT "ERASE ALL P
HONE NUMBERS? (Y/N)"
810 VC = 16: HC = 27: GOSUB 3760: IF IN$
< "Y" THEN 830
820 FOR IT = 1 TO 18: PMS(IT) = BKS: NEX
T
830 HOME: PRINT "ERASING...": FOR IT
= 1 TO 31: FOR JI = 1 TO 18: PRINT
= 1: AP$(IT, JI) = BKS: MK(IT, JI) = 0
: NEXT JI: NEXT IT
840 TM = M: TY = YR: TB = BT
850 HOME: VTAB 3: HTAB 9: PRINT "SET D
ATE AND TIME"
860 VTAB 7: HTAB 5: PRINT "ENTER MONTH
(1-12)": CN = TM: N1 = 7: N2 = 25: N3 =
2: GOSUB 3040
870 IF IN$ = ESC$ THEN RETURN
880 TM = VAL (CN$): IF TM < 1 OR TM >
12 THEN PRINT BLS: GOTO 860
890 VTAB 9: HTAB 10: PRINT "ENTER YEAR"
: CN = TY: N1 = 9: N2 = 25: N3 = 4: GOS
UB 3040
900 IF IN$ = ESC$ THEN RETURN
910 TY = VAL (CN$): IF TY < 1900 OR TY
> 2100 THEN PRINT BLS: GOTO 890
920 VTAB 11: HTAB 5: PRINT "STARTING HO
UR": PRINT "FOR APPOINTMENTS (1-12)"
930 VTAB 12: HTAB 1: PRINT "FOR APPOINT
MENTS": CN = TB: N1 = 12: N2 = 25: N3 =
2: GOSUB 3040
940 IF IN$ = ESC$ THEN RETURN
950 TB = VAL (CN$): IF TB < 1 OR TB >
12 THEN PRINT BLS: GOTO 930
960 M = TM: YR = TY: BT = TB
970 RETURN
980 REM EDIT APPOINTMENT BOOK
990 IF M THEN 1010
1000 HOME: VTAB 10: HTAB 2: PRINT "...
FOR DI = 1 TO 750: NEXT: RETURN
1010 D = 1: GOSUB 2950: GOSUB 3000
1020 GOSUB 1050
1030 RETURN
1040 REM EDIT AN APPOINTMENT PAGE

```

```

1050 HOME: GOSUB 3000: VTAB 2: PRINT D"
, DWS(DW)", MNS(M)", YR
1060 MX = 0
1070 FOR IT = 0 TO 17: X = INT (BT + IT
/ VTAB LIM(1) + 1): HTAB 1: PRINT RI
GHTS ((X) / 2) THEN PRI
1080 IF INT (IT / 2) = IT / 2 THEN PRI
NT "00": GOTO 1110
1090 PRINT "30"
1100 HTAB LIM(3): PRINT APS(D, IT + 1):
HTAB 38: PRINT CHRS(32 + (MK(D, IT
+ 1) * 10) = MX + (MK(D, IT
) * 1)): NEXT
1110 VX = LIM(1): HX = LIM(3)
1120 VC = VX: HC = HX: GOSUB 3760
1130 IF IN$ > "H" THEN GOSUB 1910: G
OTO 1130
1140 IF IN$ = ESC$ THEN GOSUB 1940: RET
URN
1150 ON (IN$ = PFS) + 2 * (IN$ = PBS) +
3 * (IN$ = PGS) + 4 * (IN$ = QCS) +
5 * (IN$ = QAS) GOTO 1190, 1210, 123
0, 1280, 1360
1160 ON (IN$ = UPS) + 2 * (IN$ = DNS OR
IN$ = CRS) + 3 * (IN$ = LFS) + 4 *
(IN$ = RTS) GOSUB 1810, 1840, 1880, 19
10
1170 GOTO 1130
1180 GOSUB 1940: D = D + 1: IF D > ND(M)
THEN D = 1
1190 GOTO 1050
1200 GOSUB 1940: D = D - 1: IF D < 1 THEN
D = ND(M)
1210 GOTO 1050
1220 GOSUB 1940: VTAB 23: HTAB 1: CALL
- 868: HTAB 4: INVERSE: PRINT "EN
TER A PAGE NUMBER (1-ND(M)): "
1230 NORMAL
1240 CN = 1: N1 = 23: N2 = 34: N3 = 2
1250 GOSUB 3040: IF IN$ = ESC$ THEN 1050
1260 IF CN < 1 OR CN > ND(M) THEN PRINT
BLS: GOTO 1250
1270 D = CN: GOTO 1050
1280 FOR IT = 0 TO 17: VTAB LIM(1) + IT:
HTAB 36: PRINT CHRS(65 + IT): N
EXT
1290 VTAB 23: HTAB 1: CALL - 868: HTAB
4: INVERSE: PRINT "SET APPOINTMEN
T MARKERS": NORMAL
1300 VC = 23: HC = 31: GOSUB 3760: IF IN$
= ESC$ THEN 1340
1310 IF IN$ < "A" OR IN$ > "R" THEN PRI
NT BLS: GOTO 1300
1320 X = ASC (IN$) - 64: MK(D, X) = (MK(D
, X) = 1) * MX + 2 * (MK(D, X) = 1
) * 0: IF MX > 4 THEN MX = 4: MK(D, X
) = 0
1330 VTAB LIM(1) + X - 1: HTAB 38: PRINT
CHRS(32 + 10 * (MK(D, X) = 1)): G
OTO 1300
1340 FOR IT = 0 TO 17: VTAB LIM(1) + IT:
HTAB 36: PRINT "": NEXT
1350 VTAB 23: HTAB 1: CALL - 868: GOTO
1130
1360 VTAB 23: HTAB 1: CALL - 868: INVER
SE: PRINT "PHONE #": NORMAL
1370 IF PMS(18) < BKS THEN VTAB 23:
HTAB 1: CALL - 868: INVERSE: PRIN
T "PHONE LIST IS FULL": NORMAL:
FOR DI = 1 TO 750: NEXT: GOTO 146
0
1380 VC = 23: HZ = 14
1390 HC = HZ: GOSUB 3760: IF IN$ = ESC$
THEN 1460
1400 IF IN$ = LFS THEN HZ = HZ - (HZ > 1
4): GOTO 1390
1410 IF IN$ = RTS OR IN$ > " " THEN H
Z = HZ + (HZ < 35): GOTO 1390
1420 IF IN$ = CRS THEN 1440
1430 GOTO 1390
1440 PMS(18) = "": Y = LOC(22) + 13: FOR
IT = 0 TO 21: PMS(18) = PMS(18) + C
HRS(PEEK(Y + IT) - 128): NEXT
1450 GOSUB 1590
1460 VTAB 23: HTAB 1: CALL - 868: GOTO
1130
1470 RETURN
1480 REM EDIT PHONE NUMBERS
1490 HOME: INVERSE: VTAB 2: HTAB 10: P
RINT "EDIT PHONE NUMBERS": NORM
AL
1500 FOR IT = 0 TO 17: VTAB LIM(1) + IT:
HTAB 1: PRINT "": RIGHTS ((X) / 2) +
STRS (IT + 1) / 2):
1510 HTAB LIM(3): PRINT PMS(IT + 1): NEX
T
1520 VX = LIM(1): HX = LIM(3)
1530 VC = VX: HC = HX: GOSUB 3760
1540 IF IN$ > "H" THEN GOSUB 1910: G
OTO 1530
1550 IF IN$ = ESC$ THEN GOSUB 1940: GOT
O 1590
1560 ON (IN$ = UPS) + 2 * (IN$ = DNS OR
IN$ = CRS) + 3 * (IN$ = LFS) + 4 *
(IN$ = RTS) GOSUB 1810, 1840, 1880, 19
10

```

Continued


```

1570 GOT TO 1530
1580 REM SORT PHONE NUMBERS
1590 VRTAB 23: HTAB 1: CALL 868: PRINT
1600 D = 1: FOR IT = 1 TO 18: IF MK(D, IT) = 1 THEN
1610 FOR JI = 1 TO 18: IF MK(D, JI) = 1 THEN
1620 IF (PNS(JI) < PNS(JI + D)) OR (PNS(JI) > PNS(JI + D)) THEN
1630 S$ = THEN PNS(JI) = PNS(JI + D):
1640 PN$ = (JI + D) = S$:
1650 NEXT JI
1660 NEXT IT
1670 INT (D / 2): IF D > 0 THEN 161
1680 RETURN
1690 REM GENERAL MENU ROUTINE
1700 FOR IT = 1 TO CX(2): VRTAB 4 + 2 * I
1710 T: HTAB 1: PRINT IT: MENUS(CX(3)
1720 + IT - 1): NEXT
1730 VRTAB 20: HTAB 8
1740 PRINT: ENTER A CHOICE: "; SPC(5); "(
1750 VQ = CX(2): HC = 25: GOSUB 3760
1760 IF IN$ = ESC$ THEN CH(CX(1)) = 0: R
1770 RETURN
1780 IF VAL(IN$) < 1 OR VAL(IN$) > C
1790 X(2) THEN PRINT BL$: GOTO 1730
1800 IF CH(CX(1)) = VAL(IN$)
1810 CH(CX(1)) > 0 THEN VRTAB 4 + 2 *
1820 CH(CX(1)): HTAB 1: PRINT "=>": F
1830 OR DI = 1 TO 500: NEXT
1840 RETURN
1850 REM CURSOR MOVEMENT ROUTINES
1860 REM CURSOR UP
1870 GOSUB 1940: VX = VX - 1: IF VX < LIM
1880 (1) THEN VX = LIM(2)
1890 RETURN
1900 REM CURSOR DOWN
1910 GOSUB 1940: VX = VX + 1: IF VX > LIM
1920 (2) THEN VX = LIM(1)
1930 IF IN$ = CR$ THEN HX = LIM(3)
1940 RETURN
1950 REM CURSOR LEFT
1960 HX = HX - 1: IF HX < LIM(3) THEN HX
1970 = LIM(4): GOSUB 1810
1980 RETURN
1990 REM CURSOR RIGHT
2000 HX = HX + 1: IF HX > LIM(4) THEN HX
2010 = LIM(3): GOSUB 1840
2020 RETURN
2030 REM STORE A LINE
2040 X$ = "": Y = 0: LOG(VX - 1) + LIM(3) -
2050 1: FOR JI = 0 TO 24: X$ = X$ + CHR$
2060 (PEEK(Y + JI) - 128): NEXT
2070 IF CH(1) = 2 THEN PNS(VX - LIM(1) +
2080 1) = X$: RETURN
2090 AP$(D, VX - LIM(1) + 1) = X$
2100 RETURN
2110 REM PRINT ROUTINES
2120 IF M THEN 2010
2130 HOME: VRTAB 10: HTAB 2: PRINT "****
2140 YOU MUST FIRST SET THE DATE ****":
2150 FOR DI = 1 TO 750: NEXT: RETURN
2160 HOME: VRTAB 3: HTAB 10: PRINT "PRIN
2170 T ROUTINES"
2180 CX(1) = 3: CX(2) = 5: CX(3) = 10: GOS
2190 UB 1700
2200 IF IN$ = ESC$ OR IN$ = "5" THEN RE
2210 TURN
2220 TB = BT: TY = YR: TM = M
2230 ON VAL(IN$) GOSUB 2090, 2190, 2260,
2240 2540:
2250 BT = TB: YR = TY: M = TM: GOSUB 2920:
2260 GOTO 1990
2270 RETURN
2280 REM PRINT ONE PAGE FROM BOOK
2290 GOSUB 2800: IF IN$ = ESC$ THEN RET
2300 URN
2310 D = CN
2320 GOSUB 2840: IF IN$ < > CR$ THEN R
2330 ETURN
2340 IF D / 2 = INT(D / 2) THEN D = D
2350 1
2360 DL = D: FOR DX = DL TO DL + 1: D = D
2370 X
2380 IF D > ND(M) THEN D = 1: M = M + 1
2390 IF M > 12 THEN M = 1: YR = YR + 1
2400 GOSUB 2630: NEXT
2410 RETURN
2420 REM PRINT APPOINTMENT BOOK
2430 GOSUB 2840: IF IN$ < > CR$ THEN R
2440 ETURN
2450 DL = ND(M): IF DL / 2 < > INT(DL
2460 / 2) THEN DL = DL + 1
2470 FOR DX = 1 TO DL: D = DX
2480 IF D > ND(M) THEN D = 1: M = M + 1:
2490 IF M > 12 THEN M = 1: YR = YR + 1
2500 GOSUB 2630: NEXT
2510 RETURN
2520 REM PRINT WEEKLY SUMMARY
2530 GOSUB 2800: IF IN$ = ESC$ THEN RET
2540 URN
2550 D = CN
2560 GOSUB 2840: IF IN$ < > CR$ THEN R
2570 ETURN
2580 PRINT
2590 LY SUMMARY SPC(32): "I": PRINT "WEEK
2600 + BKS), 28): "I" + STR$(M) + "
2610 + STR$(YR)
2620 PRINT SPC(32): "I": CR$:
2630 GOSUB 2950: DL TO DL + 6
2640 D = D + 1: IF D > ND(M) THEN D = 1:
2650 M = M + 1: IF M > 12 THEN M = 1: YR
2660 = YR + 1
2670 IF M < TM AND D = 1 THEN GOSUB
2680 2950
2690 GOSUB 3000
2700 PRINT "RIGHTS (( " + STR$(D)),
2710 3): "I": CR$: SPC(32): "I":
2720 IF M < TM THEN PRINT SPC(32):
2730 GOTO 2420
2740 X$ =
2750 FOR IT = 1 TO 18: IF MK(D, IT) = 1 T
2760 HEN WZ = WZ + 1: WA(WZ, 1) = D: WA(WZ,
2770 2) = IT: X$ = X$ + " " + STR$(WZ)
2780 NEXT: PRINT LEFT$(X$ + BKS + BKS
2790 32):
2800 PRINT
2810 NEXT DX: YR = TY: M = TM
2820 PRINT "APPOINTMENTS FOR WEEK START
2830 ING I"
2840 PRINT SPC(4): LEFT$( (MNS(M) + "
2850 + BKS), 28): "I" + STR$(YR)
2860 PRINT SPC(32): "I": CR$:
2870 FOR IT = 1 TO 28: IF IT > WZ THEN
2880 PRINT SPC(32): "I": GOTO 2500
2890 PRINT "RIGHTS (( " + STR$(
2900 IT), 2): "I": APS(WA(IT, 1), WA(IT, 2))
2910 NEXT
2920 PRINT
2930 RETURN
2940 REM PRINT PHONE NUMBERS
2950 GOSUB 2840: IF IN$ < > CR$ THEN R
2960 ETURN
2970 PRINT SPC(32): "I": CR$: "PHONE NUMB
2980 ERS FOR: SPC(13): "I":
2990 PRINT LEFT$( (MNS(M) + "
3000 + BKS), 32): "I": CR$:
3010 SPC(32): "I": CR$: SPC(32): "
3020 I"
3030 FOR IT = 1 TO 27: IF IT > 18 THEN
3040 PRINT SPC(32): "I": GOTO 2590
3050 PRINT SPC(3): PNS(IT): SPC(7): "I"
3060 NEXT
3070 PRINT
3080 RETURN
3090 REM PRINT ONE DAY IN BOOK
3100 GOSUB 2950: GOSUB 3000
3110 PRINT SPC(32): "I": SPC(10): "NOTE
3120 S.
3130 PRINT "APPOINTMENTS FOR: "MNS(M);
3140 PRINT LEFT$(BKS, 13 - LEN(MNS(M)
3150 )): PRINT "I"
3160 PRINT "D: "DWS(DW): SPC(2): L
3170 EFTS(BKS, 26 - LEN(DWS(DW))) - (D
3180 > 9)
3190 PRINT
3200 SPC(32): "I"
3210 PRINT SPC(32): "I"
3220 FOR IT = 0 TO 17: X = INT(BT + IT
3230 / 2): IF X > 12 THEN X = X - 12
3240 PRINT RIGHT$( ( " + STR$(X)), 3
3250 ): "I": IF IT / 2 < > INT(IT / 2)
3260 THEN PRINT "30 "; GOTO 2730
3270 PRINT "00"
3280 IF M < TM THEN PRINT SPC(25):
3290 GOTO 2750
3300 PRINT SPC(2): APS(D, IT + 1): SPC(
3310 1): "I"
3320 IF IT / 2 < > INT(IT / 2) THEN
3330 PRINT SPC(32): "I"
3340 NEXT
3350 PRINT
3360 RETURN
3370 REM PRINTER PROMPT SUBROUTINES
3380 VRTAB 20: HTAB 1: CALL 868: HTAB
3390 4: INVERSE: PRINT "ENTER DESIRED
3400 DAY (1 - ND(M)): "I": NORMAL: CN = 1
3410 N1 = 20: N2 = 32: N3 = 2
3420 GOSUB 3040: IF IN$ = ESC$ THEN RET
3430 URN
3440 IF CN < 1 OR CN > ND(M) THEN PRINT
3450 BLS: GOTO 2810
3460 RETURN
3470 VRTAB 22: HTAB 1: CALL 868: PRINT
3480 "POSITION PRINTER PAPER, PRESS RET
3490 URN:

```

Continued


```

2850 VC<=22:HC=17:GOSUB 3760:IF IN$
2860 VTAB<=23:HTAB 10:PRINT "PRINTING..
2870 RETURN
2880 REM TURN ON PRINTER
2890 PRINT CHR$(4);"PR#1"
2900 RETURN
2910 REM TURN OFF PRINTER
2920 PRINT CHR$(4);"PR#0"
2930 RETURN
2940 REM FIND FIRST DAY OF A MONTH
2950 A=INT(.6+(1/M)):B=YR-A:
A=M+INT(13*(A+B)/100)+INT(
((5*B)/4)-(7*INT(A/7))+
1
2970 ND(2)=28:IF YR/4=INT(YR/4)
AND (YR/100<INT(YR/100))
OR YR/400=INT(YR/400)) T
HEN ND(2)=29
2980 RETURN
2990 REM FIND DAY OF THE WEEK BASED ON
FIRST DAY OF MONTH
3000 DW=D+(7*INT(D/7))+FD-
1:IF DW>7 THEN DW=DW-7
3010 RETURN
3020 REM
3030 CN$=
3040 LEFT$((STR$(CN)+"
N3))
VTAB N1:HTAB N2:PRINT CN$
3050 FOR IT=1 TO N3:ETS(IT)=MID$(C
N$,IT,1):NEXT
3060 HX=N2
3070 IF HX<N2 THEN HX=N2
3080 IF HX>N2+1 THEN HX=N2+
1
3090 HX=HX:VC=N1:GOSUB 3760
3100 IF IN$=ESC$ THEN RETURN
3110 IF IN$=LF$ THEN HX=HX-1:GOTO
3120 3080
3130 IF IN$=RT$ THEN HX=HX+1:GOTO
3140 3080
3150 IF IN$=CR$ THEN 3170
IF (IN$>"0" AND IN$<"9")
OR IN$=" " THEN ETS(HX)=N2+1
=IN$:HX=HX+1:GOTO 3080
3160 PRINT BL$:GOTO 3080
3170 IN$+ETS(IT):NEXT CN:VAL(IN$)
3180 CN$:LEFT$(STR$(CN)+
N3)):VTAB N1:HTAB N2:PRINT CN
3190 RETURN
3200 REM LOAD APPOINTMENT FILE
3210 HOME:VTAB 2:HTAB 2
3220 PRINT "LOAD AN APPOINTMENT FILE"
3230 GOSUB 3650
3240 IF IN$=ESC$ OR T$=" " THEN RETU
RN
3250 HOME:VTAB 8:HTAB 1:PRINT "ARE Y
OU SURE YOU WISH TO LOAD AN
3260 PRINT "APPOINTMENT FILE FROM THE DI
SK DRIVE?"
3270 PRINT "ALL DATA IN MEMORY WILL B
E
3280 PRINT "OVERWRITTEN. (Y/N)"
3290 VC=12:HC=17:GOSUB 3760
3300 IF IN$<"Y" AND IN$<"N" TH
EN PRINT BL$:GOTO 3290
3310 IF IN$="N" THEN RETURN
3320 IF PD THEN GOSUB 4180
3330 PRINT CHR$(4)"VERIFY":T$
3340 HOME:VTAB 10:HTAB 1:PRINT "LOAD
ING
3350 PRINT CHR$(4)"OPEN";FL$;"D";DR
$
3360 PRINT CHR$(4)"READ";FL$
3370 INPUT X$:M=VAL(X$):INPUT X$:YR
=VAL(X$):INPUT X$:BT=VAL(X
$)
3380 FOR IT=1 TO 31:FOR JI=1 TO 18:
INPUT APS(IT,JI):INPUT X$:MK(IT,J
I)=VAL(X$):IF APS(IT,JI)=
THEN APS(IT,JI)=BK$
3390 NEXT
3400 FOR IT=1 TO 18:INPUT PNS(IT):IF
PNS(IT)=" " THEN PNS(IT)=BK$
3410 NEXT
3420 PRINT CHR$(4)"CLOSE"
3430 RETURN
3440 REM SAVE APPOINTMENT FILE
3450 HOME:VTAB 2:HTAB 2
3460 PRINT "SAVE AN APPOINTMENT FILE"
3470 GOSUB 3650
3480 IF IN$=ESC$ OR T$=" " THEN RETU
RN
3490 HOME:VTAB 10:HTAB 1:PRINT "SAVI
NG
3500 IF PD THEN GOSUB 4180
3510 PRINT CHR$(4)"OPEN";FL$;"D";DR
$
3520 PRINT CHR$(4)"CLOSE";FL$
3530 PRINT CHR$(4)"DELETE";FL$;"D";

```

```

3540 PRINT CHR$(4)"OPEN";FL$;"D";DR
$
3550 PRINT CHR$(4)"WRITE";FL$
3560 PRINT STR$(M):PRINT STR$(YR):
PRINT STR$(BT)
3570 FOR IT=1 TO 31:FOR JI=1 TO 18:
X$=AP$(IT,JI):IF X$=BK$ THEN X
$=
3580 PRINT X$:PRINT STR$(MK(IT,JI)):
NEXT
3590 FOR IT=1 TO 18:IF PNS(IT)=BK$
THEN PRINT PNS(IT):GOTO 3610
3600 PRINT CHR$(4)"CLOSE"
3610 NEXT
3620 PRINT
3630 RETURN
3640 REM FILE NAME ENTRY
3650 VTAB 10:HTAB 1
3660 PRINT "ENTER FILE NAME:"
3670 T$=VTAB 12:HTAB 4:GOSUB 4010
3680 IF IN$=ESC$ OR T$=" " THEN RETU
RN
3690 FL$=T$
3700 DR$="1"
3710 VTAB 14:HTAB 10:PRINT "DRIVE:"DR
$
3720 VC=14:HC=17:GOSUB 3760:IF IN$
<"1" AND IN$<"2" AND IN$<"PR
">ESC$ THEN PR
INT BL$:GOTO 3720
3730 IF IN$<">CR$ THEN DR$=IN$
3740 RETURN
3750 REM GET CHARACTER
3760 VTAB VC:HTAB HC
3770 GET IN$:IF IN$=" " OR IN$="." CHR$
(34) OR IN$=";" OR IN$=":" CHR$
OR IN$="|" THEN 3780
3790 IF IN$=ASC(">31 THEN VTAB VC:H
TAB HC:PRINT IN$:
3800 RETURN
3810 REM ERROR HANDLER
3820 X=PEEK(222)
3830 IF X=6 OR X=7 THEN HOME:VTAB
8:HTAB 1:PRINT "FILE NOT ON DISK
":GOTO 3890
3840 PRINT CHR$(4)"CLOSE":HOME:VTA
B 7:HTAB 1
3850 IF X=22 THEN 230
3860 ON ((X=5)+2*(X=9))+3*(X
=8)) GOTO 3950,3980,3990
3870 PRINT "ERROR NUMBER:"
3880 PRINT "AT LINE NUMBER:"PEEK(218)+PEE
K(219)
3890 VTAB 12:HTAB 1
3900 PRINT "PRESS ANY KEY TO CONTINUE:"
3910 GET IN$:IF IN$=" " THEN 3910
3920 POP:POP:POP:POP:POP:POP:POP:
POP:POP:POP:POP:POP:POP:POP:
3930 GOTO 230
3940 RETURN
3950 PRINT "FL$:"
3960 PRINT "IS NOT ON DRIVE" RIGHTS
(DR$,1)
3970 GOTO 3890
3980 PRINT "DISK IS FULL":GOTO 3890
3990 PRINT "IS THE DRIVE DOOR OPEN?":GO
TO 3890
4000 REM GET FILENAME
4010 IF PD THEN MX=15:GOTO 4030
4020 MX=30
4030 VTAB VT:HTAB HT:PRINT T$:SPC(MX
LEN(T$)):HTAB LEN(T$)+HT
4040 IF LEN(T$)=MX THEN HTAB MX
4050 IF T$=" " THEN GOSUB 4120:GOTO 4
070
4060 GOSUB 4140
4070 IF IN$=CR$ OR IN$=ESC$ THEN 417
0
4080 IF IN$=" " AND LEN(T$)<2 THEN
T$=
4090 IF IN$=LF$ THEN T$=LEFT$(T$,
LEN(T$)-1):GOTO 4030
4100 IF LEN(T$)<MX THEN T$=T$+IN
$:GOTO 4030
4110 T$=LEFT$(T$,MX-1)+IN$:GOTO
4030
4120 GET IN$:IF NOT (IN$=LF$ OR IN$
=CR$ OR (IN$>"A" AND IN$<="Z"))
THEN PRINT CHR$(7):GOTO
4120
4130 RETURN
4140 IF PD THEN GET IN$:IF NOT (IN$=
LF$ OR IN$=" " OR (IN$>"A"
AND IN$<="Z") OR (IN$>"a"
AND IN$<="z")) THEN
PRINT CHR$(7):GOTO 4140
4150 IF PD THEN GET IN$:IF NOT (IN
$=LF$ OR IN$=CR$ OR (IN$>"A"
AND IN$<="Z") OR (IN$>"a"
AND IN$<="z")) THEN PRINT
CHR$(7):GOTO 4150
4160 RETURN
4170 PRINT CHR$(4)"PREFIX,D"DR$:RETU
RN
4180

```

SONIC TYPE-IN LISTINGS


```

100 REM *** RUN-DAY-VIEW ***
110 REM *****
120 REM COPYRIGHT 1985
130 REM EMERALD VALLEY PUBLISHING CO.
140 REM BY RANDY THOMPSON
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 5.4.1
170 REM C-64 BASIC
180 REM
190 REM
200 REM DISPLAY TITLE SCREEN
210 POKE 53281,0:POKE 53280,0:POKE 646,15:POKE 53272,23:POKE 657,128:POKE 65
0,128
220 TS=SHIFT R--SHIFT D--SHIFT
VIEW:CLS=
230 PRINT"SHIFT CLR--10CRSRDOWN"
240 PRINT"13CRSRDOWN"SHIFT P
SHIFT R--SHIFT E--2SHIFT S--CTRL R
VSON--SHIFT R--SHIFT E--SHIFT T--SH
IFT U--SHIFT R--SHIFT N--CTRL RVSO
FT--SHIFT T--SHIFT O--SHIFT C--SHI
FT O--SHIFT N--SHIFT T--SHIFT I--SH
IFT N--SHIFT U--SHIFT E--X=13:Y=1
0:S=1
250 GET K$:GOSUB3810:S=ABS(S-1):POKE 19
9,S
260 FOR I=1 TO 50:NEXT:PRINTT$:IF K$=""
THEN250
270 IF ASC(K$)<>13 THEN250
280 DIM M$(12),ND(12),DWS(7),AP$(18,31)
290 MK(4,31),PNS(18)
300 RESTORE:FOR I=1 TO 12:READ M$(I),ND
(I):NEXT:FOR I=1 TO 7:READ DWS(I):N
EXT
310 REM MAIN MENU
320 PRINT"SHIFT CLR--CTRL RVSON"
TS--:GOSUB37
20
330 PRINTTAB(5)"CTRL RVSON1"CTRL RVSO
FF--SHIFT EDIT APPOINTMENTS--CRS
RDOWN"
340 PRINTTAB(5)"CTRL RVSON2"CTRL RVSO
FF--SHIFT EDIT PHONE NUMBERS--CR
SRDOWN"
350 PRINTTAB(5)"CTRL RVSON3"CTRL RVSO
FF--SHIFT PRINT ROUTINES--CRSRDO
WN"
360 PRINTTAB(5)"CTRL RVSON4"CTRL RVSO
FF--SHIFT LOAD APPOINTMENT FILE
--CRSRDOWN"
370 PRINTTAB(5)"CTRL RVSON5"CTRL RVSO
FF--SHIFT SAVE APPOINTMENT FILE
--CRSRDOWN"
380 PRINTTAB(5)"CTRL RVSON6"CTRL RVSO
FF--SHIFT EXIT PROGRAM--4CRSRDOW
N"
390 PRINT"SHIFT SELECT ONE:"
L=1:B=49:T=54:GOSUB3230:IF S$="" TH
EN390
400 ON VAL(S$) GOSUB410,2990,1120,2540,
2330,3830:GOTO310
410 REM EDIT APPOINTMENTS
420 PRINT"SHIFT CLR--CTRL RVSON"
TS--SHIFT EDIT APPOINTME
NTS--:GOSUB3720
430 PRINTTAB(5)"CTRL RVSON1"CTRL RVSO
FF--SHIFT SET/SHIFT CHANG DA
TE AND TIME--CRSRDOWN"
440 PRINTTAB(5)"CTRL RVSON2"CTRL RVSO
FF--SHIFT EDIT APPOINTMENT BOOK
--CRSRDOWN"
450 PRINTTAB(5)"CTRL RVSON3"CTRL RVSO
FF--SHIFT RETURN TO MAIN MENU--4
CRSRDOWN"
460 PRINT"SHIFT SELECT ONE:"
L=1:B=49:T=51:GOSUB3230:IF S$="" TH
EN470
480 ON VAL(S$) GOTO490,780:RETURN
490 REM SET DATE AND TIME SCHEDULE
500 IF M=0 THEN610
510 GOSUB3790:PRINT"HOME--2CRSRDOWN"
SHIFT S--SHIFT I--ERASE ALL CU
RRENT APPOINTMENTS"
520 PRINTTAB(16)"(Y/N)"
530 GET K$:IF K$<>"Y" AND K$<>"N" THEN5
30
540 IF K$="N" THEN410
550 PRINT"SHIFT ERASING":FOR I=1TO18:
FOR J=1TO31:AP$(I,J)="":NEXT:NEXT
560 FOR I=1 TO ND(M):FOR J=1 TO 4:MK(J,
I)=0:NEXT:NEXT:PRINT
570 PRINT"CRSRDOWN"SHIFT D--DO YOU WANT
TO ERASE THE PHONE NUMBERS":PRINTT
AB(16)"(Y/N)"
580 GET K$:IF K$<>"Y" AND K$<>"N" THEN5
80
590 IF K$="N" THEN610
600 FOR I=1 TO 18:PNS(I)="":NEXT
610 GOSUB3790:X=0:Y=4:GOSUB3810:PRINT"
SHIFT ENTER SHIFT MONTH (1-12):
2SHIFT CRSRLEFT"
L=2:B=48:T=57:GOSUB3230:IF S$="" TH
EN410

```

```

630 M=VAL(S$):IF M<1 OR M>12 THEN610
640 PRINTTAB(30)"CTRL RVSON--M$(M) E
SS="1985":PRINT"CRSRDOWN"SHIFT E
NTER"SHIFT YEAR:"S$--4SHIFT CRSR
LEFT"::L=4:GOSUB3250:YR=VAL(S$)
650 PRINTTAB(30)"CTRL RVSON"YR"SHIFT
CRSRLEFT"
660 X=0:Y=8:GOSUB3810:PRINT"SHIFT I
THIS CORRECT (Y/N)? :Y--2SHIFT CRSR
LEFT"
670 S$="Y":L=1:B=78:T=89:GOSUB3250:IF S
$<>"Y" AND S$<>"N" THEN670
680 IF S$="N" THEN610
690 POKE 781,8:SYS59903:GOSUB3720
700 X=0:Y=8:GOSUB3810:PRINT"SHIFT S
RTING HOUR":PRINT"FOR APPOINTMENTS
:9--2SHIFT CRSRLEFT"
710 S$="9":L=2:B=48:T=57:GOSUB3250:BT=V
AL(S$):IF BT<1 OR BT>12 THEN710
720 PRINTTAB(30)"CTRL RVSON"SS""0"
730 X=0:Y=11:GOSUB3810:PRINT"SHIFT I
THIS CORRECT (Y/N)? :Y--2SHIFT CRS
RLEFT"
740 S$="Y":L=1:B=78:T=89:GOSUB3250:IF S
$<>"Y" AND S$<>"N" THEN740
750 IF S$="N" THENFOR I=8 TO 16:POKE 7
81,I:SYS59903:NEXT:GOTO710
760 GOTO410
770 REM EDIT APPOINTMENTS
780 IF M THEN810
790 PRINT"SHIFT CLR--**SHIFT Y--YOU
MUST FIRST SET THE DATE ***":GOSUB2
910:GOTO410
800 D=1:P=1:E=1
810 GOSUB2820
820 X=7:Y=P+3:GOSUB3810:S$=AP$(P,D):L=2
830 B=32:T=90
840 GOSUB3250:AP$(P,D)=S$
850 IF K=137 AND D>1 THEN D=D-1:P=1:GOT
O820
860 IF K=133 AND D<ND(M) THEN D=D+1:P=1
:GOTO820
870 IF K=134 THEN GOSUB920:GOTO830
880 IF K=136 THEN GOSUB2940:GOTO820
890 IF K=135 THEN GOSUB3150:GOTO820
900 IF K<>95 THEN840
910 E=0:GOTO410
920 REM SET MARKERS
930 X=37:FOR Y=4 TO 21:GOSUB3810:PRINTC
HRS(61+Y):NEXT:POKE 198,0
940 X=0:Y=22:GOSUB3810:PRINT"CTRL RVSO
N"CTRL WHT"SHIFT P--PRESS THE K
EY NEXT TO THE MARKER
950 PRINT"THAT YOU WISH TO SET OR
RESET.":POKE 646,15:POKE 199,0
960 GET K$:IF K$="" THEN960
970 K=ASC(K$):IF K<>134 THEN1000
980 GOSUB1080:X=37:FOR Y=4 TO 21:GOSUB3
810:PRINT"NEXT
990 POKE 781,22:SYS 59903:POKE 781,23:S
YS 59903:RETURN
1000 IF K<65 OR K>82 THEN960
1010 Z=K-64:J=0:I=1
1020 IF Z=MK(I,D) THEN1050
1030 IF MK(I,D)=0 THEN J=I
1040 I=I+1:IF I<5 THEN1020
1050 IF I<5 THEN X=35:Y=3+Z:GOSUB3810:PR
INT"MK(I,D)=0:GOTO960
1060 IF J THEN X=35:Y=3+Z:GOSUB3810:PRIN
T"SHIFT @":MK(J,D)=Z
1070 GOTO960
1080 REM SORT MARKERS
1090 FOR I=1 TO 3:FOR J=I+1 TO 4
1100 IF MK(J,D)<MK(I,D) THENZ=MK(I,D):MK
(I,D)=MK(J,D):MK(J,D)=Z
1110 NEXT:NEXT:RETURN
1120 REM PRINT ROUTINES
1130 IF M THEN1150
1140 PRINT"SHIFT CLR--**SHIFT Y--YOU
MUST FIRST SET THE DATE ***":GOSUB2
910:RETURN
1150 PRINT"SHIFT CLR--CTRL RVSON"
TS--SHIFT PRINT"SHIFT SPACE--S
HIFT ROUTINES--:GOSUB3
720
1160 PRINTTAB(5)"CTRL RVSON1"CTRL RVSO
FF--SHIFT PRINT ONE PAGE FROM B
OOK--CRSRDOWN"
1170 PRINTTAB(5)"CTRL RVSON2"CTRL RVSO
FF--SHIFT PRINT APPOINTMENT BOO
K--CRSRDOWN"
1180 PRINTTAB(5)"CTRL RVSON3"CTRL RVSO
FF--SHIFT PRINT WEEKLY SUMMARY"
CRSRDOWN"
1190 PRINTTAB(5)"CTRL RVSON4"CTRL RVSO
FF--SHIFT PRINT PHONE NUMBERS--C
RSRDOWN"
1200 PRINTTAB(5)"CTRL RVSON5"CTRL RVSO
FF--SHIFT RETURN TO MAIN MENU--3
CRSRDOWN"
1210 PRINT"SHIFT SELECT ONE:"
L=1:B=49:T=53:GOSUB3230:IF S$="" TH
EN1220
1220 GOSUB1440,1570,1670,2080
1230 ON VAL(S$)

```

Continued


```

1240 REM PRINT ONE DAY IN BOOK
1250 GOSUB 3630:GOSUB 3680
1260 OPEN 4,4,7
1270 PRINT#4,"
1280 PRINT#4," "SHIFT APPOINTMENTS FOR:
1290 PRINT#4," LEFT$(BL$,13-LEN(M$(M)))";:P
1300 PRINT#4," "DW$(DW);
1310 PRINT#4," LEFT$(BL$,26-LEN(DW$(DW)))"
1320 PRINT#4,"
1330 PRINT#4,"
1340 FOR I=0 TO 17:X=INT(BT+I/2):IF X>12
1350 THEN X=X-12
1360 IF X<10 THEN PRINT#4," ";
1370 S$=STR$(X)+":":PRINT#4,CHR$(27)"SH
1380 IF I AND 1 THEN PRINT#4,"30 ";:GOTO 139
1390 PRINT#4,"00 ";
1400 PRINT#4,CHR$(27)"SHIFT F"AP$(I+1,
1410 D);:PRINT#4,LEFT$(BL$,25-LEN(AP$(I+
1420 1,D)))
1430 NEXT I
1440 CLOSE 4:RETURN
1450 REM PRINT A SELECTED PAGE
1460 GOSUB 3790:PRINT#4,"HOME"2CRSRDOWN"
1470 SHIFT ENTER DESIRED DAY :":L=2:B=4
1480 S$=T=57:GOSUB 3230
1490 IF S$="Y" THEN RETURN
1500 D=VAL(S$):IF D<1 OR D>ND(M) THEN
1510 POKE 198,0:PRINT:PRINT"CRSRDOWN"
1520 SHIFT POSITION PRINTER PAPER AND P
1530 RESS CTRL RVSON"SHIFT R"SHIFT E
1540 SHIFT CTRL RVSON"SHIFT R"SHIFT E
1550 GET K$:IF K$<>CHR$(13) THEN
1560 PRINT"CRSRDOWN"SHIFT PRINTING...
1570 :IF D/2=INT(D/2) THEN D=D-1
1580 FOR D=D TO D+1:GOSUB 1240:IF D>ND(M)
1590 THEN
1600 NEXT
1610 S$="N":X=0:Y=8:GOSUB 3810:PRINT"SHI
1620 FT P"PRINT ANOTHER (Y/N)? :N"2SHIFT
1630 CRSRLEFT":L=1:B=78:T=89
1640 GOSUB 3250:IF S$<>"Y" AND S$<>"N" TH
1650 EN1530
1660 IF S$="Y" THEN 1450
1670 RETURN
1680 REM PRINT WEEKLY SUMMARY
1690 GOSUB 3790:PRINT#4,"HOME"2CRSRDOWN"
1700 SHIFT ENTER DAY FOR WEEK TO BEGIN :
1710 :L=2:B=48:T=57:GOSUB 3230
1720 IF S$="Y" THEN RETURN
1730 D=VAL(S$):IF D<1 OR D>ND(M) THEN
1740 POKE 198,0:PRINT:PRINT"2CRSRDOWN"
1750 SHIFT POSITION PRINTER PAPER AND P
1760 RESS CTRL RVSON"SHIFT R"SHIFT E
1770 SHIFT CTRL RVSON"SHIFT R"SHIFT E
1780 GET K$:IF K$<>CHR$(13) THEN
1790 PRINT"CRSRDOWN"SHIFT PRINTING...
1800 :K=D
1810 OPEN 4,4,7:GOSUB 2060:PRINT#4,"SHIF
1820 T WEEKLY SUMMARY STARTING
1830 PRINT#4," "M$(M)STR$(K)LEFT$(BL$,9-
1840 LEN(M$(M)))";:YR"
1850 IF K<10 THEN PRINT#4," ";
1860 PRINT#4,"":GOSUB 2060:PRINT#4,"
1870 X=0:B=D:T=B+6:IF T>ND(M) THEN T=ND(
1880 M)
1890 FOR D=B TO T:GOSUB 3680:IF D<10 THEN
1900 PRINT#4,"D";
1910 (DW$(DW))";

```

```

1920 PRINT#4,"
1930 PRINT#4," "SHIFT APPOIN
1940 TMENTS FOR WEEK STARTING
1950 PRINT#4," "M$(M)STR$(K)LEFT$(BL$,9-
1960 LEN(M$(M)))";:YR"
1970 IF K<10 THEN PRINT#4," ";
1980 PRINT#4,"":PRINT#4,"
1990 X=0
2000 FOR I=B TO T:IF MK(J,I)=0 THEN
2010 X=X+1:S$=MID$(STR$(X),2):PRINT#4,"
2020 "S$";:IF X<10 THEN PRINT#4," ";
2030 PRINT#4,"AP$(MK(J,I),I)LEFT$(BL$,
2040 24-LEN(AP$(MK(J,I),I)))";
2050 PRINT#4,"
2060 NEXT I
2070 NEXT:IF X=28 THEN
2080 FOR I=1 TO 28-X:GOSUB 2060:NEXT
2090 PRINT#4,"
2100 CLOSE 4
2110 S$="N":X=0:Y=9:GOSUB 3810:PRINT"SHI
2120 FT P"PRINT ANOTHER (Y/N)? :N"2SHIFT
2130 CRSRLEFT":L=1:B=78:T=89
2140 GOSUB 3250:IF S$<>"Y" AND S$<>"N" TH
2150 EN2020
2160 IF S$="Y" THEN 1670
2170 RETURN
2180 PRINT#4,":RETURN
2190 REM PRINT PHONE NUMBERS
2200 GOSUB 3790
2210 POKE 198,0:PRINT"HOME"2CRSRDOWN"
2220 SHIFT POSITION PRINTER PAPER AND P
2230 RESS CTRL RVSON"SHIFT R"SHIFT E
2240 SHIFT CTRL RVSON"SHIFT R"SHIFT E
2250 GET K$:IF K$<>CHR$(13) THEN
2260 PRINT"CRSRDOWN"SHIFT PRINTING...
2270 OPEN 4,4,7
2280 PRINT#4,"SHIFT PPHONE NUMBERS FOR
2290 :PRINT#4," "M$(M)LEFT$(BL$,9-LEN(M$(
2300 M)))";:YR"
2310 PRINT#4,"
2320 FOR I=1 TO 18:IF I AND 1 THEN GOSUB
2330 2060
2340 PRINT#4," "PN$(I)LEFT$(BL$,24-LEN(P
2350 N$(I)))";:NEXT:GOSUB 2060
2360 PRINT#4,"
2370 CLOSE 4
2380 S$="N":X=0:Y=6:GOSUB 3810:PRINT"SHI
2390 FT P"PRINT ANOTHER (Y/N)? :N"2SHIFT
2400 CRSRLEFT":L=1:B=78:T=89
2410 GOSUB 3250:IF S$<>"Y" AND S$<>"N" TH
2420 EN2210
2430 IF S$="Y" THEN 2090
2440 RETURN
2450 REM INPUT FILE NAME
2460 PRINT#4,"HOME"3CRSRDOWN"SHIFT I"INPU
2470 T FILE NAME :":
2480 L=10:B=32:T=90:GOSUB 3230:FL$=S$:IF
2490 S$=" " THEN RETURN
2500 PRINT:PRINT"2CRSRDOWN"TAPE OR DISK
2510 (T/D)"4SHIFT CRSRLEFT":
2520 GET K$:IF K$<>"T" AND K$<>"D" THEN
2530 IF K$="D" THEN PRINT"2CRSRRIGHT"
2540 CTRL RVSON"
2550 IF K$="T" THEN PRINT"CTRL RVSON"
2560 RETURN
2570 REM SAVE ROUTINE
2580 IF M THEN
2590 PRINT"SHIFT CLR"***SHIFT Y"YOU
2600 MUST FIRST SET THE DATE ***":GOSUB 2
2610 910:RETURN
2620 PRINT"SHIFT CLR"CTRL RVSON"
2630 "
2640 :GOSUB 3720:GOSUB 2250
2650 IF S$=" " THEN RETURN
2660 IF K$="D" THEN OPEN 1,8,8,"@:"+FL$+
2670 ".APP,S,W":GOSUB 2780:IF ER THEN
2680 2530
2690 IF K$="T" THEN OPEN 1,1,1,FL$+
2700 ".APP"
2710 PRINT"CRSRDOWN"SAVING";
2720 PRINT#1,M
2730 PRINT#1,YR
2740 PRINT#1,BT
2750 FOR I=1 TO 4:FOR J=1 TO 31
2760 PRINT#1,MK(I,J)
2770 NEXT:

```

Continued


```

2470 FOR I=1 TO 18:FOR J=1 TO 31:PRINT".
      :SS=APS(I,J):IF SS=" THEN SS="C
      TRL WHT#1
2480 PRINT#1,SS
2490 NEXT:NEXT
2500 FOR I=1 TO 18:SS=PNS(I):IF SS=" TH
      EN SS="CTRL WHT#1
2510 PRINT#1,SS
2520 NEXT
2530 CLOSE1:CLOSE15:RETURN
2540 REM LOAD ROUTINE
2550 IF M=0 THEN2600
2560 PRINT"SHIFT CLR+SHIFT T THIS WILL
      REPLACE ALL DATA IN MEMORY."
2570 PRINT"SHIFT D TO YOU REALLY WANT TO
      DO THIS (Y/N)?"
2580 GET K$:IF K$<>"Y" AND K$<>"N" THEN2
      580
2590 IF K$="N" THEN RETURN
2600 PRINT"SHIFT CLR+CTRL RVSON#1
      :LOAD"
      :GOSUB3720:GOSUB2250
2610 IF SS=" THEN FL$=":RETURN
2620 IF K$="D" THEN OPEN1,8,8,"@":"+FL$+
      :APP$R":GOSUB2780:IF ER THEN2770
2630 IF K$="I" THEN OPEN1,1,0,FL$+
      :APP$R"
2640 PRINT"CRSRDOWN+LOADING"
2650 INPUT#1,M
2660 INPUT#1,YR
2670 INPUT#1,BT
2680 FOR I=1 TO 4:FOR J=1 TO 31
2690 INPUT#1,MK(I,J)
2700 NEXT:NEXT
2710 FOR I=1 TO 18:FOR J=1 TO 31:PRINT".
      :SS=APS(I,J):IF APS(I,J)="CTRL
      WHT#1 THEN APS(I,J)="
2720 NEXT:NEXT
2730 FOR I=1 TO 18
2740 INPUT#1,PNS(I):IF PNS(I)="CTRL WHT
      #1 THEN PNS(I)="
2750 NEXT
2760 CLOSE1:CLOSE15:RETURN
2770 OPEN15,8,15
2780 INPUT#15,ER,ES:IF ER THEN PRINT"CR
      SRDOWN+
2790 RETURN
2800 PRINT"SHIFT CLR+CTRL RVSON#1
      :SHIFT EDIT PHONE NUMBERS--
      :GOSUB3720
2820 REM PRINT A PAGE TO THE SCREEN
2830 GOSUB3790:GOSUB3630:GOSUB3680:PRINT
      "HOME+2CRSRDOWN+D+SHIFT CRSRLEF
      T+DWS(DW),MS(M),YR:PRINT
2840 FOR I=0 TO 17:X=INT(BT+I/2):IF X>12
      THEN X=X-12
2850 PRINTLEFT$(BL$-(X<10)):PRINTX"SH
      IFT CRSRLEFT:
2860 IF I AND 1 THEN PRINT"30 ";:GOTO288
      0
2870 PRINT"00 "
2880 PRINTAPS(I+1,D):NEXT:X=35
2890 FOR I=1 TO 4:Z=MK(I,D):IF Z THEN Y=
      Z+3:GOSUB3810:PRINT"SHIFT @+
2900 NEXT:RETURN
2910 REM BUZZ AND PAUSE
2920 FOR I=1 TO20:POKE 54296,15:FOR J=1
      TO 5:NEXT:POKE 54296,0:FOR Z=1 TO 5
      NEXT:NEXT:FOR I=1 TO 2000:NEXT:RETU
      RN
2940 REM JUMP PAGES IN APPOINTMENT BOOK
2950 E=0:X=0:Y=22:GOSUB3810:PRINT"SHIFT
      D+WAY TO JUMP TO :3SHIFT CRSRLE
      FT#1
2960 L=2:B=48:T=57:GOSUB3230:IF SS=" TH
      EN E=1:RETURN
2970 D=VAL(SS):IF D<1 OR D>ND(M) THEN295
      0
2980 E=1:RETURN
2990 REM EDIT PHONE NUMBERS
3000 PRINT"SHIFT CLR+CTRL RVSON#1
      :SHIFT EDIT PHONE NUMBERS--
      :GOSUB3720:PRINT"HOME+3CR
      SRDOWN#1
3010 FOR I=1 TO 18:PRINT"#I:;IF I<10 TH
      EN PRINT"
3020 PRINTTAB(7)PNS(I):NEXT
3030 D=0:E=1:P=1:X=7:Y=P+3:GOSUB3810
3040 SS=PNS(P):L=24:B=32:T=90
3050 GOSUB3250:PNS(P)=SS:IF K<>95 THEN30
      50
3060 X=0:Y=22:GOSUB3810:PRINT"SHIFT S+O
      RTING#1
3070 E=0
3080 L=17
3090 S=0
3100 FOR I=1 TO L:IF PNS(I)=" AND PNS(I
      +1)<>" THEN3120
3110 IF PNS(I)<=PNS(I+1) OR PNS(I+1)="
      THEN3130
3120 SS=PNS(I):PNS(I)=PNS(I+1):PNS(I+1)=
      SS:S=1:L=L-1
3130 NEXT:IF S=1 THEN3090
3140 RETURN
3150 REM ADD A PHONE NUMBER FROM APPOINT
      MENT EDITING SCREEN

```

```

3160 FOR I=1 TO 18:IF PNS(I)=" THEN3190
      NEXT:X=0:Y=22:GOSUB3810:PRINT"SORRY
      ,PHONE NUMBERS ARE FULL UP"
3180 FOR I=1 TO 1000:NEXT:RETURN
3190 J=1:X=0:Y=22:GOSUB3810:PRINT"ENTER
      NAME AND NUMBER:"
3200 PRINT"HOME#1:X=0:Y=22:GOSUB3810:PO
      KE204,PEEK(204)OR128:PRINT
3210 E=0:GOSUB3250:E=1:PNS(J)=SS:POKE 78
      1,22:SYS 59903:POKE 781,23:SYS59902
      :GOSUB3060:POKE 781,22:SYS59902:E=1:
      RETURN
3230 REM INPUT ROUTINE
3240 SS="
3250 Z=0:S=PEEK(214)+40+PEEK(211)+1024:P
      OKE 213,L+PEEK(211)
3260 GOSUB3590:IF K=13 AND E=0 THEN POKE
      213,40:GOTO3570
3270 IF E=1 AND D<>0 THEN3300
3280 IF E=1 THEN IF K=95 THEN POKE 213,4
      0:GOTO3570
3290 GOTO3310
3300 IF E=1AND(K=95ORK=133ORK=137ORK=134O
      RK=136ORK=135)THEN POKE213,40:GOTO3
      570
3310 GOSUB3320:GOTO3260
3320 IF E=0 OR (K<>13 AND K<>145 AND K<>
      17) THEN3420
3330 IF D=0 THEN3380
3340 POKE204,1:GOSUB3570:APS(P,D)=SS
3350 IF (K=13 ORK=17)ANDP<18 THEN P=P+1:
      S=S+40:GOTO3370
3360 IF K=145 AND P>1 THEN P=P-1:S=S-40
3370 Z=0:X=7:Y=P+3:GOSUB3810:SS=APS(P,D)
      :POKE 213,L+PEEK(211):RETURN
3380 POKE204,1:GOSUB3570:PNS(P)=SS
3390 IF (K=13 ORK=17)ANDP<18 THEN P=P+1:
      S=S+40:GOTO3410
3400 IF K=145 AND P>1 THEN P=P-1:S=S-40
3410 Z=0:X=7:Y=P+3:GOSUB3810:SS=PNS(P):P
      OKE 213,L+PEEK(211):RETURN
3420 IF Z=0 THEN3450
3430 IF K=20 THEN K$=":I=-1:GOTO3530
3440 IF K=157 THEN I=-1:GOTO3560
3450 IF Z=L THEN3580
3460 IF Z=LEN(SS) THEN3490
3470 IF K=29 THEN I=1:GOTO3560
3480 IF K=148 AND LEN(SS)<L THEN K$="
      +MID$(SS,Z+1,1):GOTO3520
3490 IF K<B OR K>T THEN3580
3500 IF K=34 OR K=36 OR K=42 OR K=44 OR
      K=58 OR K=63 OR K=64 THEN3580
3510 I=0
3520 Z=Z+1
3530 SS=LEFT$(SS,Z-1)+K$+MID$(SS,Z+1)
3540 IF K$=" THEN K$=CHR$(20)
3550 IF LEN(K$)=2 THEN K$=CHR$(148):I=-1
3560 PRINT K$:POKE 216,0
3570 POKE S+Z,PEEK(S+Z) AND 127:Z=Z+I
3580 POKE 204,1:RETURN
3590 REM INPUT ONE CHARACTER
3600 POKE 204,0:POKE 207,0:GET K$:IF K$=
      " THEN3600
3610 K=ASC(K$):IF K>192 AND K<258 THEN K
      =K-128
3620 RETURN
3630 REM FIND FIRST DAY OF A MONTH
3640 A=INT((0.6+(1/M))):B=YR-A:A=M+12*A:C=
      B/100
3650 A=INT((13*(A+1)/5)+INT((5*B)/4)-INT(
      C)+INT(C/4)):FD=A-(7*INT(A/7))+1
3660 IF M=2THEN ND(2)=28:IF (YR/4=INT(YR
      /4))OR(YR/400=INT(YR/400))THEN ND(2
      )=29
3670 RETURN
3680 REM FIND DAY OF THE WEEK BASED ON F
      IRST DAY OF MONTH
3690 DW=D-(7*INT(D/7))+FD-1:IF DW>7 THEN
      DW=DW-7
3700 IF DW=0 THEN DW=7
3710 RETURN
3720 REM PRINT THE STATUS LINE
3730 X=0:Y=24:GOSUB3810
3740 IF M THEN PRINT"CTRL RVSON+SHIFT
      D+ATE:"M$(M),"YR"SHIFT CRSRLEFT#1
      :LEFT$(BL$,9-LEN(M$(M))):GOTO3760
3750 PRINT"CTRL RVSON#1
3760 IFFLS<>" THEN PRINT"SHIFT FILE
      :FL$LEFT$(BL$,10-LEN(FL$)):GOTO3
      780
3770 PRINT"CTRL RVSON#1
3780 POKE 2023,160:POKE 56295,15:PRINT"
      HOME+5CRSRDOWN#1:RETURN
3790 REM CLEAR MIDDLE OF SCREEN
3800 FOR I=1 TO 23:POKE 781,1:SYS59903:N
      EXT:RETURN
3810 REM PLACE CURSOR AT X,Y
3820 POKE 781,Y:POKE 782,X:POKE 783,0:SY
      S 65520:RETURN
3830 REM EXIT PROGRAM
3840 PRINT"SHIFT CLR+2CRSRDOWN+SHIFT
      D+O YOU REALLY WANT TO EXIT THE PRO
      GRAM?":PRINTTAB(15)"(Y/N)"
3850 GET K$:IF K$<>"Y" AND K$<>"N" THEN3
      850

```

Continued


```

1240 REM PRINT ONE DAY IN BOOK
1250 GOSUB 3630:GOSUB 3680
1260 OPEN 4,4,7
1270 PRINT#4,"
1280 PRINT#4," "SHIFT APPOINTMENTS FOR:
1290 PRINT#4," MS(M); LEFT$(BL$,13-LEN(M$(M)));:P
1300 PRINT#4," D"; DW$(DW);
1310 PRINT#4," LEFT$(BL$,26-LEN(DW$(DW)))"
1320 PRINT#4,"
1330 PRINT#4,"
1340 FOR I=0 TO 17:X=INT(BT+I/2):IF X>12
1350 THEN X=X-12
1360 IF X<10 THEN PRINT#4," ";
1370 S$=STR$(X)+":PRINT#4,CHR$(27)"SHIFT
1380 IF I AND 1 THEN PRINT#4,"30";:GOTO 139
1390 PRINT#4,"00";
1400 PRINT#4,CHR$(27)"SHIFT F"AP$(I+1,
1410 D);:PRINT#4,LEFT$(BL$,25-LEN(AP$(I+
1420 1,D)))
1430 IF I AND 1 THEN PRINT#4,"
1440 NEXT
1450 PRINT#4,"
1460 CLOSE 4:RETURN
1470 REM PRINT A SELECTED PAGE
1480 GOSUB 3790:PRINT#4,"HOME"2CRSRDOWN"
1490 SHIFT ENTER DESIRED DAY :";:L=2:B=4
1500 8:T=57:GOSUB 3230
1510 IF S$="" THEN RETURN
1520 D=VAL(S$):IF D<1 OR D>ND(M) THEN
1530 POKE 198,0:PRINT:PRINT"CRSRDOWN"
1540 SHIFT POSITION PRINTER PAPER AND P
1550 RESS CTRL RVSON"SHIFT R"SHIFT E
1560 SHIFT T"SHIFT U"SHIFT R"SHIFT N
1570 CTRL RVSON"
1580 GET K$:IF K$<>CHR$(13) THEN
1590 PRINT"CRSRDOWN"SHIFT PRINTING...
1600 :IF D/2=INT(D/2) THEN D=D-1
1610 FOR D=D TO D+1:GOSUB 1240:IF D>ND(M)
1620 THEN
1630 NEXT
1640 S$="N":X=0:Y=8:GOSUB 3810:PRINT"SHI
1650 FT PRINT ANOTHER (Y/N):N"2SHIFT
1660 CRSRLEFT":L=1:B=78:T=89
1670 GOSUB 3250:IF S$<>"Y" AND S$<>"N" TH
1680 EN1530
1690 IF S$="Y" THEN
1700 RETURN
1710 REM PRINT APPOINTMENT BOOK
1720 GOSUB 3790
1730 POKE 198,0:PRINT"HOME"2CRSRDOWN"
1740 SHIFT POSITION PRINTER PAPER AND P
1750 RESS CTRL RVSON"SHIFT R"SHIFT E
1760 SHIFT T"SHIFT U"SHIFT R"SHIFT N
1770 CTRL RVSON"
1780 GET K$:IF K$<>CHR$(13) THEN
1790 PRINT"CRSRDOWN"SHIFT PRINTING...
1800 FOR D=1 TO ND(M):GOSUB 1240:NEXT
1810 S$="N":X=0:Y=6:GOSUB 3810:PRINT"SHI
1820 FT PRINT ANOTHER (Y/N):N"2SHIFT
1830 CRSRLEFT":L=1:B=78:T=89
1840 GOSUB 3250:IF S$<>"Y" AND S$<>"N" TH
1850 EN1630
1860 IF S$="Y" THEN
1870 RETURN
1880 REM PRINT WEEKLY SUMMARY
1890 GOSUB 3790:PRINT#4,"HOME"2CRSRDOWN"
1900 SHIFT ENTER DAY FOR WEEK TO BEGIN :
1910 :L=2:B=48:T=57:GOSUB 3230
1920 IF S$="" THEN RETURN
1930 D=VAL(S$):IF D<1 OR D>ND(M) THEN
1940 POKE 198,0:PRINT:PRINT"2CRSRDOWN"
1950 SHIFT POSITION PRINTER PAPER AND P
1960 RESS CTRL RVSON"SHIFT R"SHIFT E
1970 SHIFT T"SHIFT U"SHIFT R"SHIFT N
1980 CTRL RVSON"
1990 GET K$:IF K$<>CHR$(13) THEN
2000 PRINT"CRSRDOWN"SHIFT PRINTING...
2010 :K=D
2020 OPEN 4,4,7:GOSUB 2060:PRINT#4,"SHIF
2030 T WEEKLY SUMMARY STARTING
2040 PRINT#4,"
2050 MS(M)STR$(K)LEFT$(BL$,9-
2060 LEN(M$(M)))";:YR
2070 IF K<10 THEN PRINT#4," ";
2080 PRINT#4,"I":GOSUB 2060:PRINT#4,"
2090 X=0:B=D:T=B+6:IF T>ND(M) THEN T=ND(
2100 M)
2110 FOR D=B TO T:GOSUB 3680:IF D<10 THEN
2120 PRINT#4,"D";
2130 DW$(DW)LEFT$(BL$,9-LEN
2140 (DW$(DW)))

```

```

2150 PRINT#4,"
2160 FOR J=1 TO 4:FOR I=1 TO 31
2170 PRINT#1,MK(I,J)
2180 NEXT I:NEXT
2190 GOSUB 3720:GOSUB 2250
2200 IF S$="" THEN RETURN
2210 IF K$="D" THEN OPEN 1,8,8,"@:"+FL$+
2220 ".APP,S,W":GOSUB 2780:IF ER THEN
2230 2530
2240 IF K$="T" THEN OPEN 1,1,1,FL$+
2250 ".APP"
2260 PRINT#1,CRSRDOWN"SAVING";
2270 PRINT#1,M
2280 PRINT#1,YR
2290 PRINT#1,BT
2300 FOR I=1 TO 4:FOR J=1 TO 31
2310 PRINT#1,MK(I,J)
2320 NEXT I:NEXT
2330 GOSUB 3720:GOSUB 2250
2340 IF S$="" THEN RETURN
2350 IF K$="D" THEN OPEN 1,8,8,"@:"+FL$+
2360 ".APP,S,W":GOSUB 2780:IF ER THEN
2370 2530
2380 IF K$="T" THEN OPEN 1,1,1,FL$+
2390 ".APP"
2400 PRINT#1,CRSRDOWN"SAVING";
2410 PRINT#1,M
2420 PRINT#1,YR
2430 PRINT#1,BT
2440 FOR I=1 TO 4:FOR J=1 TO 31
2450 PRINT#1,MK(I,J)
2460 NEXT I:NEXT

```



```

2470 FOR I=1 TO 18:FOR J=1 TO 31:PRINT".
      :SS=APS(I,J):IF SS=" THEN SS="FC
      TRL WHT
2480 PRINT#1,SS
2490 NEXT:NEXT
2500 FOR I=1 TO 18:SS=PNS(I):IF SS=" TH
      EN SS="CTRL WHT
2510 PRINT#1,SS
2520 NEXT
2530 CLOSE1:CLOSE15:RETURN
2540 REM LOAD ROUTINE
2550 IF M=0 THEN2600
      PRINT#1,SHIFT CLR SHIFT THIS WILL
      REPLACE ALL DATA IN MEMORY.
2570 PRINT#1,SHIFT T DO YOU REALLY WANT TO
      DO THIS (Y/N)?
2580 GET KS:IF KS<>"Y" AND KS<>"N" THEN2
      580
2590 IF KS="N" THEN RETURN
2600 PRINT#1,SHIFT CLR CTRL RVSON
      :GOSUB3720:GOSUB2250
2610 IF KS=" THEN FLS=":RETURN
2620 IF KS="D" THEN OPEN1,8,8,"@0:"+FLS+
      ".APP,S,R":GOSUB2780:IF ER THEN2770
2630 IF KS="T" THEN OPEN1,1,0,FLS+".APP"
2640 PRINT#1,CRSRDOWN"LOADING";
2650 INPUT#1,M
2660 INPUT#1,YR
2670 INPUT#1,BT
2680 FOR I=1 TO 4:FOR J=1 TO 31
      INPUT#1,MK(I,J)
2690 NEXT:NEXT
2700 FOR I=1 TO 18:FOR J=1 TO 31:PRINT".
      :
2720 INPUT#1,APS(I,J):IF APS(I,J)="CTRL
      WHT" THEN APS(I,J)="
2730 NEXT:NEXT
2740 FOR I=1 TO 18
      INPUT#1,PNS(I):IF PNS(I)="CTRL WHT
      " THEN PNS(I)="
2760 NEXT
2770 CLOSE1:CLOSE15:RETURN
2780 OPEN15,8,15
2790 INPUT#15,ER,ES:IF ER THEN PRINT#1,CR
      SRDOWN"ES":GOSUB2910
2800 RETURN
2810 PRINT#1,SHIFT CLR CTRL RVSON
      :SHIFT EDIT PHONE NUMBERS
      :GOSUB3720
2820 REM PRINT A PAGE TO THE SCREEN
2830 GOSUB3790:GOSUB3630:GOSUB3680:PRINT
      "HOME"2CRSRDOWN"D"SHIFT CRSRLEF
      T"DWS(DW),MS(M),YR:PRINT
2840 FOR I=0 TO 17:X=INT(BT+I/2):IF X>12
      THEN X=X-12
2850 PRINTLEFT$(BL$(X<10));:PRINTX"SH
      IFT CRSRLEFT";
2860 IF I AND 1 THEN PRINT"30 ";:GOTO288
      0
2870 PRINT"00 ";
2880 PRINTAPS(I+1,D):NEXT:X=35
2890 FOR I=1 TO 4:Z=MK(I,D):IF Z THEN Y=
      Z+3:GOSUB3810:PRINT#1,SHIFT @
2900 NEXT:RETURN
2910 REM BUZZ AND PAUSE
2920 FOR I=1 TO20:POKE 54296,15:FOR J=1
      TO 5:NEXT:POKE 54296,0:FOR Z=1 TO 5
      NEXT:NEXT:FOR I=1 TO 2000:NEXT:RETU
      RN
2940 REM JUMP PAGES IN APPOINTMENT BOOK
2950 E=0:X=0:Y=22:GOSUB3810:PRINT#1,SHIFT
      D"DAY TO JUMP TO :3SHIFT CRSRLE
      FT";
2960 L=2:B=48:T=57:GOSUB3230:IF SS=" TH
      EN E=1:RETURN
2970 D=VAL(SS):IF D<1 OR D>ND(M) THEN295
      0
2980 E=1:RETURN
2990 REM EDIT PHONE NUMBERS
3000 PRINT#1,SHIFT CLR CTRL RVSON
      :SHIFT EDIT PHONE NUMBERS
      :GOSUB3720:PRINT#1,HOME"3CR
      SRDOWN"
3010 FOR I=1 TO 18:PRINT#"I::IF I<10 TH
      EN PRINT"
3020 PRINTTAB(7)PNS(I):NEXT
3030 D=0:E=1:P=1:X=7:Y=P+3:GOSUB3810
3040 SS=PNS(P):L=24:B=32:T=90
3050 GOSUB3250:PNS(P)=SS:IF K<>95 THEN30
      50
3060 X=0:Y=22:GOSUB3810:PRINT#1,SHIFT S"O
      RTING..
3070 E=0
3080 L=17
3090 S=0
3100 FOR I=1 TO L:IF PNS(I)=" AND PNS(I
      +1)<>" THEN3120
3110 IF PNS(I)<=PNS(I+1) OR PNS(I+1)="
      THEN3130
3120 SS=PNS(I):PNS(I)=PNS(I+1):PNS(I+1)=
      SS:S=1:L=L-1
3130 NEXT:IF S=1 THEN3090
3140 RETURN
3150 REM ADD A PHONE NUMBER FROM APPOINT
      MENT EDITING SCREEN
3160 FOR I=1 TO 18:IF PNS(I)=" THEN3190
      NEXT:X=0:Y=22:GOSUB3810:PRINT"SORRY
      ,PHONE NUMBERS ARE FULL UP"
3180 FOR I=1 TO 1000:NEXT:RETURN
3190 J=1:X=0:Y=22:GOSUB3810:PRINT"ENTER
      NAME AND NUMBER:"
3200 PRINT#1,HOME"X=0:Y=22:GOSUB3810:PO
      KE240,PEEK(240)OR128:PRINT
3210 E=0:GOSUB3230:E=1:PNS(J)=SS:POKE 78
      1,22:SYS 59903:POKE 781,23:SYS59902:E=1:
      GOSUB3060:POKE 781,22:SYS59902:E=1:
      RETURN
3220 REM INPUT ROUTINE
3230 SS="
3240 Z=0:S=PEEK(214)*40+PEEK(211)+1024:P
      OKE 213,L+PEEK(211)
3250 GOSUB3590:IF K=13 AND E=0 THEN POKE
      213,40:GOTO3570
3260 IF E=1 AND D<>0 THEN3300
3270 IF E=1 THEN IF K=95 THEN POKE 213,4
      0:GOTO3570
3280 GOTO3310
3290 IF E=1AND(K=95ORK=133ORK=137ORK=134O
      RK=136ORK=135)THEN POKE213,40:GOTO3
      570
3310 GOSUB3320:GOTO3260
3320 IF E=0 OR (K<>13 AND K<>145 AND K<>
      17) THEN3420
3330 IF D=0 THEN3380
3340 POKE204,1:GOSUB3570:APS(P,D)=SS
3350 IF (K=13 ORK=17)ANDP<18 THEN P=P+1:
      S=S+40:GOTO3370
3360 IF K=145 AND P>1 THEN P=P-1:S=S-40
3370 Z=0:X=7:Y=P+3:GOSUB3810:SS=APS(P,D)
      :POKE 213,L+PEEK(211):RETURN
3380 POKE204,1:GOSUB3570:PNS(P)=SS
3390 IF (K=13 ORK=17)ANDP<18 THEN P=P+1:
      S=S+40:GOTO3410
3400 IF K=145 AND P>1 THEN P=P-1:S=S-40
3410 Z=0:X=7:Y=P+3:GOSUB3810:SS=PNS(P):P
      OKE 213,L+PEEK(211):RETURN
3420 IF Z=0 THEN3450
3430 IF K=20 THEN KS="":I=-1:GOTO3530
3440 IF K=157 THEN I=-1:GOTO3560
3450 IF Z=L THEN3580
3460 IF Z=LEN(SS) THEN3490
3470 IF K=29 THEN I=1:GOTO3560
3480 IF K=148 AND LEN(SS)<L THEN KS=" "+
      MIDS(SS,Z+1,1):GOTO3520
3490 IF K<B OR K>T THEN3580
3500 IF K=34 OR K=36 OR K=42 OR K=44 OR
      K=58 OR K=63 OR K=64 THEN3580
3510 I=0
3520 Z=Z+1
3530 SS=LEFT$(SS,Z-1)+KS+MIDS(SS,Z+1)
3540 IF KS=" THEN KS=CHR$(20)
3550 IF LEN(KS)=2 THEN KS=CHR$(148):I=-1
3560 PRINT KS:POKE 216,0
3570 POKE S+Z,PEEK(S+Z) AND 127:Z=Z+1
3580 POKE 204,1:RETURN
3590 REM INPUT ONE CHARACTER
3600 POKE 204,0:POKE 207,0:GET KS:IF KS=
      " THEN3600
3610 K=ASC(KS):IF K>192 AND K<258 THEN K
      =K-128
3620 RETURN
3630 REM FIND FIRST DAY OF A MONTH
3640 A=INT(.6+(1/M)):B=YR-A:A=M+12*A:C=
      B/100
3650 A=INT(13*(A+1)/5)+INT((5*B)/4)-INT(
      C)+INT(C/4):FD=A-(7*INT(A/7))+1
3660 IF M=2THEN ND(2)=28:IF (YR/4=INT(YR
      /4)OR(YR/400=INT(YR/400)))THEN ND(2
      )=29
3670 RETURN
3680 REM FIND DAY OF THE WEEK BASED ON F
      IRST DAY OF MONTH
3690 DW=D-(7*INT(D/7))+FD-1:IF DW>7 THEN
      DW=DW-7
3700 IF DW=0 THEN DW=7
3710 RETURN
3720 REM PRINT THE STATUS LINE
3730 X=0:Y=24:GOSUB3810
3740 IF M THEN PRINT#1,CTRL RVSON
      :DATE:MS(M),YR"SHIFT CRSRLEFT
      :LEFT$(BL$,9-LEN(MS(M))):GOTO3760
      PRINT#1,CTRL RVSON
3750 :
3760 IF FL<>" THEN PRINT#1,SHIFT FILE
      :FL$LEFT$(BL$,10-LEN(FL$)):GOTO3
      780
3770 PRINT#1,CTRL RVSON
3780 POKE 2023,160:POKE 56295,15:PRINT#1,
      HOME"5CRSRDOWN":RETURN
3790 REM CLEAR MIDDLE OF SCREEN
3800 FOR I=1 TO 23:POKE 781,I:SYS59903:N
      EXT:RETURN
3810 REM PLACE CURSOR AT X,Y
3820 POKE 781,Y:POKE 782,X:POKE 783,0:SY
      S 65520:RETURN
3830 REM EXIT PROGRAM
3840 PRINT#1,SHIFT CLR"2CRSRDOWN"SHIFT
      D"DO YOU REALLY WANT TO EXIT THE PRO
      GRAM?":PRINTTAB(15)"(Y/N)"
3850 GET KS:IF KS<>"Y" AND KS<>"N" THEN3
      850

```

Continued

RUN-DAY-VIEW *Continued*

COMMODORE 64

```

3860 IF KE="N" THEN RETURN
3870 POKE 657,0:PRINT"SHIFT CLR BYE..."
3880 DATA "SHIFT JANUARY",31,"SHIFT F
SHIFT FEBRUARY",28,"SHIFT MARCH",31,"S
SHIFT APRIL",30,"SHIFT MAY",31,"S
SHIFT JUNE",30,"SHIFT

```

```

3890 DATA "SHIFT JULY",31,"SHIFT AUG
SHIFT SEPTEMBER",30,"SH
SHIFT OCTOBER",31,"SHIFT NOVEMBER"
3900 DATA "SHIFT DECEMBER",31
3910 DATA "SHIFT SUNDAY",1,"SHIFT MOND
SHIFT TUESDAY",2,"SHIFT WEDN
SHIFT THURSDAY",3,"SHIFT F
SHIFT FRIDAY",4,"SHIFT
3920 DATA "SHIFT SATURDAY"

```

HCM

RUN-DAY-VIEW

IBM PC & IBM PCjr

```

1000 ** RUN-DAY-VIEW **
1100 ** **
1200 ** **
1300 ** COPYRIGHT 1985 **
1400 ** EMERALD VALLEY PUBLISHING CO **
1500 ** BY RANDY THOMPSON **
1600 ** HOME COMPUTER MAGAZINE **
1700 ** VERSION 5.4.1 **
1800 ** IBM PC, WITH CARTRIDGE BASIC **
1900 ** IBM PC WITH BASICA **
2000 ** **
2100 ** INITIALIZE VARIABLES **
2200 DIM DWS(7),MS(12),ND(12),PN$(18),AP
$(18,31),MK(4,31):FOR I=1 TO 20:CLS
=CLS+CHR$(29):NEXT
ILS=CHR$(34)+":TS="Run-Day-View
230
240 FOR I=1 TO 5:ES=ES+CHR$(1):KEY I:CH
RS(I):NEXT:FOR I=6 TO 10:KEY I,:N
EXT
250 KEY OFF:FOR I=1 TO 7:READ DWS(I):NE
XT:FOR I=1 TO 12:READ MS(I):READ ND
(I):NEXT
260 TITLE SCREEN 0:CLS:LOCATE 12,14:
270 PRINT TS:GOSUB 2430
280 MAIN MENU
290 CLS:PRINT TAB(15)"Main Menu":LOCATE
6,1
300 PRINT "1) Edit appointments":PRINT
PRINT "2) Edit phone numbers":PRIN
T:PRINT "3) Print routines":PRINT
310 PRINT "4) Load appointments":PRINT:
PRINT "5) Save appointments":PRINT:
PRINT "6) Exit program":PRINT:
320 T=54:GOSUB 1340:IF K=0 THEN 320
330 ON K GOSUB 340,870,400,2200,2110,24
60:GOTO 280
340 EDIT APPOINTMENTS MENU
350 CLS:PRINT TAB(12)"Edit Appointments
":LOCATE 6,1
360 PRINT "1) Set date":PRINT:PRINT "2
) Edit appointment book":PRINT:PRIN
T "3) Return to the Main Menu":PRIN
T
370 T=51:GOSUB 1340:IF K=0 THEN 370
380 ON K GOTO 480,680,390:GOTO 370
390 RETURN
400 PRINT ROUTINES MENU
410 IF M=0 THEN GOSUB 1450:RETURN
420 CLS:PRINT TAB(12)"Print Routines":L
OCATE 6,1
430 PRINT "1) Print one page":PRINT:PR
INT "2) Print appointment book":PRI
NT:PRINT "3) Print weekly summary":
PRINT
440 PRINT "4) Print phone numbers":PRIN
T:PRINT "5) Return to the Main Menu"
450 T=53:GOSUB 1340:IF K=0 THEN 450
460 ON K GOTO 1640,1700,1740,1930,390
470 GOTO 340
480 SET TIME AND DATE
490 CLS:LOCATE 1,15:PRINT "Set Date"
500 IF M=0 THEN 570
510 LOCATE 5,1:PRINT "Do you wish to er
ase the":PRINT "appointment data? (
Y/N) the":GOSUB 2520:IF S$="N" THEN 3
40
520 LOCATE 9,1:PRINT "Erasing":M=0
530 FOR I=1 TO 18:PRINT "":FOR J=1 TO
31:AP$(J,J)=NEXT J:FOR I=1 TO 1
4:FOR J=1 TO 31:MK(I,J)=0:NEXT J,I
540 LOCATE 9,1:PRINT "":LOCATE 9,1
550 PRINT "Do you wish to erase the":PR
INT "phone numbers? (Y/N)":GOS
UB 2520:IF S$="N" THEN 480
560 LOCATE 13,1:PRINT "Erasing":FOR I=
1 TO 18:PRINT "":PN$(I)=:NEXT:G
OTO 480
570 S$="1":LOCATE 5,1:PRINT "Enter month
(1-12)":S$:LOCATE 5,1
21:L=2:B=48:T=57:GOSUB 1070:IF S$=
THEN 340 ELSE M=VAL(S$)
580 IF M<1 OR M>12 THEN 570 ELSE LOCATE
5,21:PRINT M$(M)
590 S$="1985"
600 LOCATE 8,1:PRINT "Enter year":S$LEF
TS(CLS,4):L=4:B=48:T=57:GOSUB 1070
YR=VAL(S$):IF LEN(S$)<4 THEN 590
610 LOCATE 11,1:PRINT "Is this correct (
Y/N)":CHR$(29):GOSUB 2520:IF S$=
"N" THEN M=0:GOTO 480

```

```

620 GOSUB 2590:LOCATE 11,1:PRINT"
630 LOCATE 11,1:PRINT"Enter starting ti
me"
640 S$="9":LOCATE 12,1:PRINT "for appoi
nments (1-12)":S$:LOCATE
12,26:L=2:B=48:T=57:GOSUB 1070:BT=V
AL(S$)
650 IF BT<1 OR BT>12 THEN 640 ELSE LOCA
TE 12,26:PRINT STR$(BT):00"
660 LOCATE 15,1:PRINT "Is this correct (
Y/N)":CHR$(29):GOSUB 2520
670 IF S$="N" THEN LOCATE 15,1:PRINT"
":GOTO 640:E
LSE 340
680 EDIT APPOINTMENT BOOK
690 IF M=0 THEN GOSUB 1450:GOTO 340
700 D=1:F=1
710 GOSUB 970:LOCATE 4+P,7:X=0
720 S$=AP$(P,D):E=1:L=24:B=32:T=122:GOS
UB 1080:AP$(P,D)=S$:IF K=13 THEN 34
0
730 ON ASC(K$) GOTO 740,750,760,770,780
740 IF D>1 THEN D=D-1:GOTO 710 ELSE 720
750 IF D<ND(M) THEN D=D+1:GOTO 710 ELSE
720
760 LOCATE 25,1:GOSUB 1410:GOTO 710
770 C=CSRLIN:Y=POS(0):GOSUB 1480:GOSUB
1380:LOCATE C,Y:GOTO 720
780 INPUT A:PHONE NUMBER FROM THE EDI
TING SCREEN
790 F=CSRLIN:G=POS(0):H=X:LOCATE 24,1
800 FOR J=1 TO 18:IF PN$(J)= THEN 830
810 NEXT:PRINT "Sorry, call phone number
care full up":GOSUB 1470:LOCATE 2
4,1
820 PRINT "":GOTO 860
830 PRINT "Enter phone number...":LOCA
TE 25,1:PRINT
840 LOCATE 25,1:PRINT ">":L=24:B=32:T=
122:GOSUB 1050:PN$(J)=S$:LOCATE 24,
1
850 PRINT "":GOSU
B 920:GOSUB 1380
860 LOCATE F,G:X=H:GOTO 720
870 EDIT PHONE NUMBERS
880 CLS:PRINT TAB(12)"Edit Phone Number
s"
890 FOR I=1 TO 18:LOCATE I+4,2:PRINT "#
":IF I<10 THEN PRINT
900 PRINT I:PN$(I):NEXT:GOSUB 1380
910 E=2:L=24:B=32:T=122:F=1:S$=PN$(F):L
OCATE 5,7:GOSUB 1070:PN$(F)=S$
920 SORT PHONE NUMBERS
930 LOCATE 25,1:PRINT "Sor
ting..."
940 FOR I=1 TO 17:FOR J=I+1 TO 18
950 IF (PN$(J)<PN$(I) AND PN$(J)<>"") O
R (PN$(I)= AND PN$(J)<>"") THEN S
WAP PN$(J),PN$(I)
960 NEXT J:RETURN
970 PRINT A PAGE TO THE SCREEN
980 CLS:PRINT TAB(10)"Edit Appointment
Book"
990 LOCATE 3,1:GOSUB 2640:PRINT STR$(D)
":DWS(DW):LOCATE 3,16:PRINT M$(M)
":LOCATE 3,32:PRINT STR$(YR)
1000 FOR I=1 TO 18:LOCATE 4+I,1:A=BT+INT
((I-1)/2):IF A>12 THEN A=A-12
1010 PRINT USING "##";A:IF I=AND 1 THEN
PRINT "00":ELSE PRINT "30":
1020 PRINT AP$(I,D):NEXT
1030 FOR I=1 TO 4:J=MX(I,D):IF J THEN LO
CATE 4+J,32:PRINT CHR$(17)
1040 NEXT:GOSUB 1380:RETURN
1050 INPUT ROUTINE
1060 S$="":E=0
1070 X=0
1080 C=CSRLIN:Y=POS(0)
1090 LOCATE C,Y,1
1100 GOSUB 1300:IF K=13 THEN LOCATE C,Y,
0:E=0:RETURN
1110 I=0
1120 IF X>0 AND A=75 THEN X=X-1:PRINT CH
RS(29):GOTO 1100
1130 IF X<LEN(S$) AND A=77 THEN X=X+1:PR
INT CHR$(28):GOTO 1100
1140 IF X=0 AND E=8 THEN X=X-1:A=LEN(S$)
-X:PRINT CHR$(29)RIGHT$(S$,A-1)":
:FOR J=1 TO A:PRINT CHR$(29):NEXT:
E$="":GOTO 1280

```

Continued

SONSIST IN LISTINGS


```

1150 IF X<LEN(S$) AND A=83 THEN A=LEN(S$)
1160 IF X=0 THEN PRINT "RIGHTS(S$,A-1)";:NEXT K$="":GOTO 1280
1170 IF E=1 AND INSTR(E$,K$) THEN E=0:RE
1180 IF A=72 OR A=80 THEN ON E GOTO 1180
1190 IF A=72 AND P<1 THEN P=P-1:C=C-1
1200 IF A=80 AND P<18 THEN P=P+1:C=C+1
1210 S$=AP$(P,D):X=0:Y=7:GOTO 1090
1220 S$=PNS$(P):S$=S$
1230 IF A=72 AND P<1 THEN P=P-1:C=C-1
1240 IF A=80 AND P<18 THEN P=P+1:C=C+1
1250 S$=PNS$(P):X=0:Y=7:GOTO 1090
1260 IF K<B OR K>T OR X=L OR INSTR(IL$,K$) THEN 1100
1270 I=1
1280 IF LEN(S$)=X THEN S$=S$+K$ ELSE S$=LEFT$(S$,X)+K$+RIGHT$(S$,LEN(S$)-X-1)
1290 X=X+1:PRINT K$;:GOTO 1100
1300 S$=INKEY$:IF K$=" " THEN 1300
1310 K=ASC(K$):IF K=0 THEN A=ASC(RIGHT$(K$,1)) ELSE A=0
1320 IF E>0 AND K=27 THEN K=13 ELSE IF E>0 AND K=13 THEN K=0:A=80
1330 RETURN
1340 INPUT A MENU SELECTION
1350 LOCATE 25,1:PRINT "Select one :";:B=49:L=1:GOSUB 1050
1360 K=VAL(S$)
1370 RETURN
1380 PRESS ESC TO EXIT
1390 LOCATE 25,1:PRINT "
1400 LOCATE 25,12:PRINT "Press ESC to ex
1410 INPUT A DAY OF THE MONTH
1420 C=CSRLIN
1430 LOCATE C,1:PRINT "ENTER DESIRED DAY
1440 (1)";:PRINT USING "##";ND(M);:PRIN
1450 T(1)";:LEFT$(CL$,2);:L=2:B=48:T=5
1460 7:GOSUB 1050:IF S$=" " THEN RETURN
1470 D=VAL(S$):IF D<1 OR D>ND(M) THEN 14
1480 30 ELSE RETURN
1490 BEEP AND PAUSE
1500 CLS:LOCATE 12,7:PRINT "YOU MUST FIR
1510 ST SET THE DATE!"
1520 BEEP:FOR I=1 TO 2200:NEXT:RETURN
1530 SET MARKERS
1540 FOR I=65 TO 82:LOCATE I-60,33,0:PRI
1550 NT CHR$(I):NEXT:LOCATE 25,1
1560 PRINT "PRESS KEYS A-R TO SET/RESET
1570 THE MARKERS:"
1580 K$=INKEY$:IF K$=" " THEN 1510 ELSE K
1590 =ASC(K$):IF K=4 THEN 1600
1600 IF (K<65 OR K>82) AND (K<97 OR K>12
1610 2) THEN 1510 ELSE K=K-64:IF K>18 TH
1620 EN K=K-32
1630 J=0:FOR I=1 TO 4
1640 IF MK(I,D)=K THEN 1570
1650 IF MK(I,D)=0 THEN J=I
1660 NEXT
1670 IF I<5 THEN LOCATE 4+K,32:PRINT "
1680 :MK(I,D)=0:GOTO 1510
1690 IF J THEN LOCATE 4+K,32:PRINT CHR$(
1700 17):MK(J,D)=K
1710 GOTO 1510
1720 SORT MARKERS
1730 FOR I=1 TO 3:FOR J=I+1 TO 4:IF MK(J
1740 D)<MK(I,D) THEN SWAP MK(J,D),MK(I,
1750 D)
1760 NEXT J,I
1770 FOR I=5 TO 22:LOCATE I,33:PRINT "
1780 :NEXT:RETURN
1790 PRINT A PAGE TO THE PRINTER
1800 CLS:PRINT TAB(13)"PRINT A PAGE":LOC
1810 ATE 5,1:GOSUB 1410:IF S$=" " THEN RE
1820 TURN
1830 B=D:IF B/2=INT(B/2) THEN B=B-1
1840 T=B+1:IF T>ND(M) THEN T=B
1850 GOSUB 2410:LOCATE 7,15:PRINT "PRINT I
1860 NG...":OPEN "LPT1:" FOR OUTPUT AS #
1870 1
1880 FOR D=B TO T:GOSUB 2000:NEXT:CLOSE
1890 #1:GOSUB 2500:IF S$="Y" THEN 1640 E
1900 LSE RETURN
1910 PRINT APPOINTMENT BOOK
1920 CLS:PRINT TAB(13)"PRINT BOOK":GOSUB
1930 2410:LOCATE 7,15:PRINT "PRINTING..
1940
1950 OPEN "LPT1:" FOR OUTPUT AS #1
1960 FOR D=1 TO ND(M):GOSUB 2000:NEXT:CL
1970 OSE #1:GOSUB 2500:IF S$="Y" THEN 17
1980 00 ELSE RETURN
1990 PRINT WEEKLY SUMMARY
2000 CLS:PRINT TAB(13)"WEEKLY SUMMARY":L
2010 OCATE 5,1:GOSUB 1410:IF S$=" " THEN
2020 RETURN
2030 A=D:GOSUB 2410:LOCATE 7,15:PRINT "P
2040 RINTING...":OPEN "LPT1:" FOR OUTPUT
2050 AS #1
2060 GOSUB 2090:PRINT #1," WEEKLY SUMMAR
2070 Y STARTING" TAB(33)"I"

```

```

1780 PRINT #1,"";:PRINT #1,USING"##";A;
1790 :PRINT #1,"";:MS(M) TAB(24)YR TAB(3
1800 3)"I"
1810 GOSUB 2100:GOSUB 2090:K=0:B=D:T=B+6
1820 :IF T>ND(M) THEN T=ND(M)
1830 FOR D=B TO T:GOSUB 2640:PRINT #1,"
1840 :DWS(DW) TAB(33)"I"
1850 PRINT #1,TAB(33)"I"
1860 FOR I=1 TO 4
1870 IF MK(I,D) THEN K=K+1:PRINT #1," *
1880 :PRINT #1,USING"##";K;:ELSE PRINT
1890 #1," "
1900 PRINT #1,"";:IF I/2=INT
1910 (I/2) THEN PRINT #1,TAB(33)"I"
1920 NEXT:GOSUB 2100:NEXT
1930 IF B+6>ND(M) THEN FOR I=D+1 TO B+7:
1940 FOR J=1 TO 3:GOSUB 2090:NEXT:GOSUB
1950 2100:NEXT
1960 GOSUB 2090:PRINT #1," *APPOINTMENTS
1970 FOR WEEK STARTING"
1980 PRINT #1,USING"##";A:PRINT #1," "
1990 MS(M) TAB(24)YR TAB(33)"I":GOSUB 21
2000 00
2010 K=0:FOR D=B TO T:FOR I=1 TO 4:C=MK(
2020 I,D)
2030 IF C THEN K=K+1:PRINT #1," *";:PRIN
2040 T #1,USING"##";K;:PRINT #1," "
2050 :AP$(C
2060 ,D) TAB(33)"I"
2070 NEXT I,D
2080 IF K<28 THEN FOR I=1 TO 28-K:GOSUB
2090 2090:NEXT
2010 GOSUB 2100:CLOSE #1:GOSUB 2500:IF S
2020 $="Y" THEN 1740 ELSE RETURN
2030 PRINT PHONE NUMBERS
2040 CLS:PRINT TAB(10)"PRINT PHONE NUMBE
2050 RS":LOCATE 5,1:GOSUB 2410:LOCATE 7,
2060 15:PRINT "PRINTING..
2070 OPEN "LPT1:" FOR OUTPUT AS #1:GOSUB
2080 2090
2090 PRINT #1," PHONE NUMBERS FOR:" TAB(
2100 33)"I"
2110 PRINT #1,"MS(M) TAB(24)YR TAB(33)
2120 "I":GOSUB 2100
2130 FOR I=1 TO 18:IF I AND 1 THEN GOSUB
2140 2090
2150 PRINT #1,"PNS(I) TAB(33)"I":NEXT:
2160 GOSUB 2090:GOSUB 2100:CLOSE #1:GOSU
2170 B 2500:IF S$="Y" THEN 1930 ELSE RET
2180 URN
2190 PRINT ONE DAY TO THE PRINTER
2200 GOSUB 2640:GOSUB 2090
2210 PRINT #1,"APPOINTMENTS FOR:"MS(M)
2220 TAB(33)"I"
2230 PRINT #1,USING"##";D::PRINT #1,"
2240 :DWS(DW) TAB(33)"I"
2250 GOSUB 2100:GOSUB 2090
2260 FOR I=1 TO 18:A=INT((I-1)/2):IF
2270 A>12 THEN A=A-12
2280 PRINT #1,USING"##";A::PRINT #1," "
2290 :A=(INT(I/2)=I/2):IF A THEN PRINT
2300 #1," "
2310 :ELSE PRINT #1," "
2320 PRINT #1,AP$(I,D) TAB(33)"I":IF A T
2330 HEN GOSUB 2090
2340 NEXT:GOSUB 2100:RETURN
2350 PRINT #1,":RETURN
2360 PRINT #1,":RETURN
2370 SAVE APPOINTMENTS
2380 IF M=0 THEN GOSUB 1450:RETURN
2390 CLS:PRINT TAB(13)"SAVE APPOINTMENTS
2400 :GOSUB 2350:IF S$=" " THEN RETURN
2410 ON ERROR GOTO 2310:OPEN FL$ FOR OUT
2420 PUT AS #1
2430 PRINT #1,M:PRINT #1,YR:PRINT #1,BT
2440 FOR I=1 TO 4:FOR J=1 TO 31:PRINT #1
2450 ,MK(I,J):NEXT J,I
2460 FOR I=1 TO 18:FOR J=1 TO 31:PRINT #
2470 1,AP$(I,J):NEXT J,I
2480 FOR I=1 TO 18:PRINT #1,PNS(I):NEXT
2490 CLOSE #1:ON ERROR GOTO 0:RETURN
2500 LOAD APPOINTMENTS
2510 CLS:PRINT TAB(13)"LOAD APPOINTMENTS
2520 :IF M=0 THEN 2240
2530 LOCATE 5,1:PRINT "This will replace
2540 all current data."
2550 :PRINT "Is this
2560 OK (Y/N)";:GOSUB 2520:IF S$="N" TH
2570 EN RETURN
2580 LOCATE 5,1:PRINT "
2590 :PRINT "
2600
2610 GOSUB 2350:IF S$=" " THEN RETURN
2620 ON ERROR GOTO 2330:OPEN FL$ FOR INP
2630 UT AS #1
2640 INPUT #1,M:INPUT #1,YR:INPUT #1,BT
2650 FOR I=1 TO 4:FOR J=1 TO 31:INPUT #1
2660 ,MK(I,J):NEXT J,I
2670 FOR I=1 TO 18:FOR J=1 TO 31:INPUT #
2680 1,AP$(I,J):NEXT J,I
2690 FOR I=1 TO 18:INPUT #1,PNS(I):NEXT
2700 CLOSE #1:ON ERROR GOTO 0:GOSUB 2590
2710 :RETURN
2720 ERROR ROUTINE FOR SAVE
2730 BEEP:CLS:LOCATE 11,1:PRINT "ERROR I
2740 N SAVING "MID$(FL$,3,LEN(FL$)-6)"
2750 :LOCATE 13,1:PRINT "ERR# "ERR:GOSUB
2760 2430:RESUME 2110
2770 ERROR ROUTINE FOR LOAD

```

Continued

Continued

IBM PC & IBM PCjr

[illegible]

2510	LOCATE	9,1:PRINT	"PRINT ANOTHER (Y/N)	
2520	'INPUT	GOTO 2540		
2530	S\$="Y"	A GOTO 2550	RESPONSE	
2540	S\$="N"			
2550	PRINT	CHR\$(29)::LOCATE	CSRLIN,PO	
	S(0),1::L=1:B=65:T=121:GOSUB	1070		
2560	IF S\$<"Y"	AND S\$<"N"		
	AND S\$<"Y"	AND S\$<"N"		
2570	IF S\$<"Y"	THEN S\$="N"	ELSE IF S\$="Y"	
	THEN S\$="Y"			
2580	RETURN			
2590	'FIND	FIRST DAY OF THE MONTH		
2600	A=INT(.6+(1/M)):B=YR-A:A=M+12*A:C=B			
	/100			
2610	A=INT(13*(A+1)/5)+INT((5*B)/4)-INT(C)+INT(C/4):FD=A-(7*INT(A/7))+1			
2620	IF M=2 AND YR/4=INT(YR/4) THEN ND(M)			
	=29			
2630	RETURN			
2640	'FIND	THE DAY OF THE WEEK BASED ON		
	FIRST DAY OF THE MONTH			
2650	DW=D-(7*INT(D/7))+FD-1:IF DW>7 THEN			
	DW=DW-7			
2660	IF DW=0 THEN DW=7			
2670	RETURN			
2680	DATA SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY,SATURDAY			
2690	DATA JANUARY,31,FEBRUARY,28,MARCH,31,APRIL,30,MAY,31,JUNE,30			
2700	DATA JULY,31,AUGUST,31,SEPTEMBER,30,OCTOBER,31,NOVEMBER,30,DECEMBER,31			

RUN-DAY-VIEW

TI-99/4A

```

100 REM ** * * * * *
110 REM ** RUN-DAY-VIEW **
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 5.4.1
180 OPTI BASIC OR EXTENDED BASIC
190 DIM ON BASE 1
200 DIM DW$(7), PN$(18), AP$(18,31), MK$(4)
210 GOSUB 4980
220 BLS=" "
230 FOR I=1 TO 7
240 READ NEXT I DW$(I)
250 CALL CLEAR
260 PRINT:;
270 :; ** RUN-DAY-VIEW **:;
280 GOSUB 4160
290 CALL CLEAR
300 PRINT MAIN MENU:; "1) EDIT APP
OINTMENTS:; "2) EDIT PHONE NUMBERS"
:; "3) PRINT ROUTINES":;
310 PRINT "4) LOAD APPOINTMENTS":; "5) S
AVE APPOINTMENTS":; "6) EXIT PROGRAM
":;
320 T=54
330 GOSUB 4200
340 ON K-48 GOSUB 360,4280,430,2040,204
ON 1980
350 GOTO 290
360 CALL CLEAR
370 PRINT EDIT AND APPOINTMENTS:; "1)
SET DATE BOOK AND TIME":; "2) EDIT APPOIN
TMENT:;
380 PRINT "3) RETURN TO MAIN MENU":;
:;
390 T=51
400 GOSUB 4200
410 ON K-48 GOTO 510,1100,420
420 RETURN
430 IF M$=" " THEN 4090
440 CALL CLEAR
450 PRINT "PRINT ROUTINES":; "1) PRI
NT ONE PAGE":; "2) PRINT APPOINTMENT
BOOK:;
460 PRINT "3) PRINT WEEKLY SUMMARY":; "4
") PRINT PHONE NUMBERS":; "5) RETURN
TO MAIN MENU":;
470 T=53
480 GOSUB 4200
490 ON K-48 GOTO 3300,3190,2450,4650,50
0
500 RETURN
510 CALL CLEAR
520 IF M$=" " THEN 700
530 PRINT DO YOU WISH TO ERASE THE
APPOINTMENT DATA (Y/N)?
540 GOSUB 4250
550 IF K=78 THEN 360
560 PRINT PLEASE WAIT...
570 GOSUB 4980
580 FOR I=1 TO 18
590 FOR J=1 TO 31
600 AP$(I,J)=
610 NEXT J
620 NEXT I

```

```

640 PRINT "DO YOU WISH TO ERASE THE  

650 GOSUB 4250  

660 IF K=78 THEN 690  

670 FOR I=1 TO 18  

680 PNT$(I)=" "  

690 NEXT I  

700 CALL CLEAR  

710 INPUT S$;ENTER MONTH (1-12) :";S$  

720 IF S$="" THEN 730  

730 GOTOSUB 980  

740 IF B=1 THEN 700  

750 M=VAL(S$)  

760 IF (M>0)*(M<13) THEN 800  

770 M=0  

780 GOSUB 3880  

790 GOTORE 700  

800 RESTORE 5100  

810 FOR I=1 TO M  

820 READ M$,ND  

830 NEXT I  

840 INPUT S$;ENTER YEAR :";S$  

850 IF (S$^"V")*(LEN(S$)=4) THEN 880  

860 GOSUB 980  

870 GOTOSUB 980  

880 GOSUB 980  

890 IF B=1 THEN 840  

900 YR=VAL(S$)  

910 PRINT M$,YR  

920 GOSUB 4240  

930 IF K=78 THEN 690  

940 PRINT "STARTING HOUR"  

950 INPUT S$;FOR APPOINTMENTS (1-12) :";S$  

960 IF (S$^"")*(LEN(S$)<3) THEN 990  

970 GOSUB 980  

980 GOTOSUB 980  

990 GOSUB 3810  

1000 IF B=1 THEN 940  

1010 BT=VAL(S$)  

1020 IF (BT^"")*(BT<13) THEN 1050  

1030 GOSUB 980  

1040 GOTOSUB 980  

1050 PRINT "STARTING TIME";BT;":00"  

1060 GOSUB 4240  

1070 IF K=78 THEN 940  

1080 GOSUB 3890  

1090 GOTOSUB 3890  

1100 IF M$^" THEN 1130  

1110 GOSUB 4090  

1120 GOTOSUB 4090  

1130 D=1  

1140 GOSUB 1420  

1150 PRINT  

1160 GOSUB 1520  

1170 T=53  

1180 GOSUB 5210  

1190 IF K=55 THEN 360  

1200 ON T TO 1200 GOSUB 1710,1810,1220,4830  

1210  

1220 GOSUB 1370  

1230 CALL CLEAR  

1240 GOSUB 1580  

1250 PRINT  

1260 INPUT S$;ENTER APPOINTMENT.."  

1270 GOSUB 1510  

1280 IF B=1 THEN 1250  

1290 AP$(K)=S$  

1300 RETURN  

1310 R=0

```

Continued


```

1330 A=LEN(S$)
1330 IF A<21 THEN 1360
1340 PRINT: "ENTRY MUST BE UNDER 21 CHARACTERS LONG":
1350 B=1
1360 RETURN
1370 PRINT "PRESS LETTER (A-R)":
1380 GOSUB 4060
1390 IF (K<65)+(K>82) THEN 1380
1400 I=K-64
1410 RETURN
1420 GOSUB 4000
1430 CALL CLEAR
1440 S$=STR$(D)
1450 PRINT S$: " "; DW$(DW); " "; MS$: " "; YR
1460 PRINT
1470 FOR I=1 TO 18
1480 PRINT CHR$(64+I); " ";
1490 GOSUB 1580
1500 NEXT I
1510 RETURN
1520 FOR I=1 TO 4
1530 GOSUB 5050
1540 IF T=0 THEN 1560
1550 CALL HCHAR(T+4,2,42,1)
1560 NEXT I
1570 RETURN
1580 T=BT+INT((I-1)/2)
1590 IF T<13 THEN 1610
1600 T=T-12
1610 IF T>9 THEN 1630
1620 PRINT " ":
1630 S$=STR$(T):
1640 PRINT S$: " ";
1650 IF I/2=INT(I/2) THEN 1680
1660 PRINT "00":
1670 GOTO 1690
1680 PRINT "30":
1690 PRINT " "; AP$(I,D)
1700 RETURN
1710 PRINT "ENTER DESIRED DAY(1-"; STR$(N
D); "): "
1720 INPUT S$
1730 IF S$=" " THEN 1780
1740 GOSUB 3810
1750 IF B=1 THEN 1710
1760 D=VAL(S$)
1770 IF (D<1)+(D>ND) THEN 1790
1780 RETURN
1790 PRINT "NO SUCH DAY EXISTS IN "; MS$:
1800 GOTO 1710
1810 GOSUB 1370
1820 B=K-64
1830 J=0
1840 FOR I=1 TO 4
1850 GOSUB 5050
1860 IF B=1 THEN 1950
1870 IF T<>0 THEN 1890
1880 J=1
1890 NEXT I
1900 IF J=0 THEN 1940
1910 T=B
1920 I=J
1930 GOSUB 5070
1940 RETURN
1950 T=0
1960 GOSUB 5070
1970 RETURN
1980 CALL CLEAR
1990 PRINT "ARE YOU SURE YOU WISH TO EXIT THE PROGRAM (Y/N)?"
2000 GOSUB 4250
2010 IF K=78 THEN 1410
2020 CALL CLEAR
2030 END
2040 IF (M$="")*(K=53) THEN 4090
2050 CALL CLEAR
2060 PRINT "DEVICE & FILE NAME:"
2070 INPUT ">": FL$
2080 IF FL$="CS1" THEN 2130
2090 IF SEG$(FL$,1,3)<"DSK" THEN 2330
2100 IF (SEG$(FL$,4,1)<"1")+(SEG$(FL$,4,1)>"3") THEN 2330
2110 IF SEG$(FL$,5,1)<"." THEN 2330
2120 FL$=SEG$(FL$,1,15)
2130 IF K=53 THEN 2340
2140 PRINT: "THIS WILL REPLACE ALL DATA IN MEMORY DO YOU WISH TO DO THIS (Y/N)"
2150 GOSUB 4250
2160 IF K=78 THEN 2330
2170 OPEN #1: FL$, INTERNAL, INPUT, FIXED 1
2180 INPUT #1: MS$, ND, YR, BT
2190 INPUT #1: MK$(1), MK$(2), MK$(3), MK$(4)
2200 FOR I=0 TO 9 STEP 9
2210 FOR J=1 TO 31
2220 INPUT #1: AP$(I+1,J), AP$(I+2,J), AP$(I+3,J), AP$(I+4,J), AP$(I+5,J), AP$(I+6,J), AP$(I+7,J), AP$(I+8,J), AP$(I+9,J)
2230 NEXT J
2240 INPUT #1: PNs$(I+1), PNs$(I+2), PNs$(I+3), PNs$(I+4), PNs$(I+5), PNs$(I+6), PNs$(I+7), PNs$(I+8), PNs$(I+9)
2250 NEXT I
2260 RESTORE 5100
2270 FOR M=1 TO 12
2280 READ S$, T
2290 IF M=S$ THEN 2310
2300 NEXT M
2310 GOSUB 3900
2320 CLOSE #1
2330 RETURN
2340 OPEN #1: FL$, INTERNAL, OUTPUT, FIXED 1
2350 PRINT #1: MS$, ND, YR, BT
2360 PRINT #1: MK$(1), MK$(2), MK$(3), MK$(4)
2370 FOR I=0 TO 9 STEP 9
2380 FOR J=1 TO 31
2390 PRINT #1: AP$(I+1,J), AP$(I+2,J), AP$(I+3,J), AP$(I+4,J), AP$(I+5,J), AP$(I+6,J), AP$(I+7,J), AP$(I+8,J), AP$(I+9,J)
2400 NEXT J
2410 PRINT #1: PNs$(I+1), PNs$(I+2), PNs$(I+3), PNs$(I+4), PNs$(I+5), PNs$(I+6), PNs$(I+7), PNs$(I+8), PNs$(I+9)
2420 NEXT I
2430 CLOSE #1
2440 RETURN
2450 CALL CLEAR
2460 PRINT "WEEKLY SUMMARY":
2470 GOSUB 1710
2480 IF S$=" " THEN 3180
2490 A=D
2500 GOSUB 3760
2510 GOSUB 3740
2520 PRINT #1: "WEEKLY SUMMARY STARTING
2530 PRINT #1: " "; MS$: " "; STR$(A); SEG$(BL$,1,9-LEN(M$)); " "; YR:
2540 IF A>9 THEN 2560
2550 PRINT #1: " ";
2560 PRINT #1: "1"
2570 GOSUB 3740
2580 GOSUB 3720
2590 K=0
2600 B=D
2610 M=B+6
2620 IF M<=ND THEN 2640
2630 M=ND
2640 FOR D=B TO M
2650 GOSUB 4000
2660 IF D>9 THEN 2680
2670 PRINT #1: " ";
2680 PRINT #1: D; " "; DW$(DW); SEG$(BL$,1,26-LEN(DW$(DW))); "1"
2690 FOR I=1 TO 4
2700 GOSUB 5050
2710 IF T=0 THEN 2770
2720 K=K+1
2730 PRINT #1: " "; STR$(K);
2740 IF K>9 THEN 2760
2750 PRINT #1: " ";
2760 GOTO 2780
2770 PRINT #1: " ";
2780 IF I/2=INT(I/2) THEN 2800
2790 PRINT #1: "1"
2800 NEXT I
2810 GOSUB 3720
2820 NEXT D
2830 IF B+7<ND THEN 2900
2840 FOR I=D+1 TO B+7
2850 FOR J=1 TO 3
2860 GOSUB 3740
2870 NEXT J
2880 GOSUB 3720
2890 NEXT I
2900 GOSUB 3740
2910 PRINT #1: " *APPOINTMENTS FOR WEEK S
TARTING I
2920 PRINT #1: " "; MS$: " "; STR$(A); SEG$(BL$,1,9-LEN(M$)); " "; YR:
2930 IF A>9 THEN 2950
2940 PRINT #1: " ";
2950 PRINT #1: "1"
2960 GOSUB 3720
2970 K=0
2980 FOR D=B TO M
2990 FOR I=1 TO 4
3000 GOSUB 5050
3010 IF T=0 THEN 3070
3020 K=K+1
3030 PRINT #1: " "; STR$(K);
3040 IF K>9 THEN 3060
3050 PRINT #1: " ";
3060 PRINT #1: " "; AP$(T,D); SEG$(BL$,1,27-LEN(AP$(T,D))); "1"
3070 NEXT I
3080 NEXT D
3090 IF K=28 THEN 3130
3100 FOR I=1 TO 28-K
3110 GOSUB 3740
3120 NEXT I
3130 GOSUB 3720
3140 CLOSE #1
3150 PRINT: "PRINT ANOTHER (Y/N)"
3160 GOSUB 4250
3170 IF K=89 THEN 2450

```

Continued


```

3180 RETURN
3190 CALL CLEAR
3200 PRINT "PRINT BOOK"::
3210 GOSUB 3760
3220 FOR D=1 TO ND
3230 GOSUB 3460
3240 NEXT D
3250 CLOSE #1
3260 PRINT "PRINT ANOTHER (Y/N)"
3270 GOSUB 4250
3280 IF K=89 THEN 3190
3290 RETURN
3300 CALL CLEAR
3310 PRINT "PRINT PAGE..."::
3320 GOSUB 1710
3330 IF S=1 THEN 3450
3340 GOSUB 3760
3350 IF D/2<>INT(D/2) THEN 3370
3360 D=D-1
3370 FOR D=D TO D+1
3380 GOSUB 3460
3390 IF D=ND THEN 3410
3400 NEXT D
3410 CLOSE #1
3420 PRINT "PRINT ANOTHER (Y/N)"
3430 GOSUB 4250
3440 IF K=89 THEN 3300
3450 RETURN
3460 GOSUB 4000
3470 GOSUB 3740
3480 PRINT #1:"APPOINTMENTS FOR: ";MS;S
      EG$(BLS,1,13-LEN(MS));";"
3490 IF D>9 THEN 3510
3500 PRINT #1:"";
3510 PRINT #1:D;";";DW$(DW);SEG$(BLS,1,
      26-LEN(DW$));";"
3520 GOSUB 3720
3530 GOSUB 3740
3540 FOR I=1 TO 18
3550 T=BT+INT((I-1)/2)
3560 IF T<13 THEN 3580
3570 T=T-12
3580 IF T>9 THEN 3600
3590 PRINT #1:"";
3600 PRINT #1:STR$(T);";";
3610 T=(INT(I/2)=1/2)
3620 IF T=1 THEN 3650
3630 PRINT #1:"";
3640 GOTO 3660
3650 PRINT #1:"30";
3660 PRINT #1:AP$(I,D);SEG$(BLS,1,26-LEN
      (AP$(I,D)));";"
3670 IF T=0 THEN 3690
3680 GOSUB 3740
3690 NEXT I
3700 GOSUB 3720
3710 RETURN
3720 PRINT #1:""
3730 RETURN
3740 PRINT #1:""
3750 RETURN
3760 PRINT
3770 INPUT "ENTER PRINTER PARAMETER:";SS
3780 OPEN #1:SS
3790 GOSUB 4150
3800 RETURN
3810 B=0
3820 FOR I=1 TO LEN(SS)
3830 A=ASC(SEG$(SS,I,1))
3840 IF (A<58)*(A>47) THEN 3860
3850 B=1
3860 NEXT I
3870 IF B=0 THEN 3890
3880 PRINT "ILLEGAL ENTRY"::
3890 RETURN
3900 A=INT(.6+(1/M))
3910 B=YR-A
3920 A=M+12*A
3930 C=B/100
3940 A=INT(13*(A+1)/5)+INT((5*B)/4)-INT(
      C)+INT(C/4)
3950 FD=A-(7*INT(A/7))+1
3960 IF (M=2)*(YR/400=INT(YR/400)) THEN 3
      980
3970 IF (M<>2)+((YR/4)<>(INT(YR/4))) THEN
      3990
3980 ND=ND+1
3990 RETURN
4000 DW=D-(7*INT(D/7))+FD-1
4010 IF DW<8 THEN 4030
4020 DW=DW-7
4030 IF DW<>0 THEN 4050
4040 DW=7
4050 RETURN
4060 CALL KEY$(0,K,S)
4070 IF S=0 THEN 4060
4080 RETURN
4090 CALL CLEAR
4100 PRINT "YOU MUST FIRST SET THE DATE"
4110 CALL SOUND(100,440,2)
4120 FOR I=1 TO 700
4130 NEXT I
4140 RETURN
4150 PRINT "POSITION THE PRINTER PAPER
    
```

```

4160 PRINT "PRESS [ENTER] TO CONTINUE"
4170 GOSUB 4060
4180 IF K<>13 THEN 4170
4190 RETURN
4200 PRINT "SELECT ONE"
4210 GOSUB 4060
4220 IF (K<49)+(K>T) THEN 4210
4230 RETURN
4240 PRINT "IS THIS CORRECT (Y/N)?"
4250 GOSUB 4060
4260 IF (K<>89)*(K<>78) THEN 4250
4270 RETURN
4280 CALL CLEAR
4290 PRINT "PHONE NUMBERS"::
4300 FOR I=1 TO 18
4310 PRINT "CHR$(64+I);";";
4320 PRINT "#";
4330 IF I>9 THEN 4350
4340 PRINT "STR$(I);";";PN$(I)
4350 PRINT I
4360 NEXT I
4370 PRINT
4380 T=50
4390 GOSUB 4210
4400 ON K=48 GOTO 4410,4510
4410 GOSUB 1370
4420 CALL CLEAR
4430 PRINT "ENTER PHONE NUMBER..."
4440 INPUT ">";SS
4450 K=1
4460 GOSUB 1310
4470 IF B=1 THEN 4440
4480 PN$(K)=SS
4490 GOTO 4280
4500 PRINT "SORTING..."
4510 L=17
4520 A=0
4530 FOR I=1 TO L
4540 IF (PN$(I)=")*(PN$(I+1)<") THEN 4
      570
4550 IF (PN$(I)<=PN$(I+1))+(PN$(I+1)="")
      THEN 4620
4560 SS=PN$(I)
4570 PN$(I)=PN$(I+1)
4580 PN$(I+1)=SS
4590 A=1
4600 L=L-1
4610 NEXT I
4620 IF A=1 THEN 4530
4630 RETURN
4640 CALL CLEAR
4650 PRINT "PHONE NUMBERS FOR:"
4660 GOSUB 3760
4670 PRINT #1:"";MS;SEG$(BLS,1,9-LEN(MS
      ));";";
4680 PRINT #1:"";YR;
4690 GOSUB 3720
4700 FOR I=1 TO 18
4710 IF I/2=INT(I/2) THEN 4740
4720 GOSUB 3740
4730 PRINT #1:"";PN$(I);SEG$(BLS,1,31-L
      EN(PN$(I)));";"
4740 NEXT I
4750 GOSUB 3740
4760 GOSUB 3720
4770 CLOSE #1
4780 PRINT "PRINT ANOTHER (Y/N)"
4790 GOSUB 4250
4800 IF K=89 THEN 4650
4810 RETURN
4820 FOR K=1 TO 18
4830 IF PN$(K)=" THEN 4890
4840 NEXT K
4850 PRINT "SORRY, PHONE NUMBERS ARE ALL
      FULL UP";
4860 GOSUB 4110
4870 RETURN
4880 CALL CLEAR
4890 PRINT "ENTER PHONE NUMBER..."
4900 INPUT ">";SS
4910 GOSUB 1310
4920 IF B=1 THEN 4900
4930 PN$(K)=SS
4940 PRINT
4950 GOSUB 4510
4960 RETURN
4970 FOR I=1 TO 4
4980 MK$(I)=
4990 FOR J=1 TO 31
5000 MK$(I)=MK$(I)&CHR$(0)
5010 NEXT J
5020 NEXT I
5030 RETURN
5040 T=ASC(SEG$(MK$(I),D,1))
5050 RETURN
5060 MK$(I)=SEG$(MK$(I),1,D-1)&CHR$(T)&S
      EG$(MK$(I),D+1,31-D)
5070 RETURN
5080 DATA SUNDAY, MONDAY, TUESDAY, WEDNESDA
      Y, THURSDAY, FRIDAY, SATURDAY
5090 DATA JANUARY, 31, FEBRUARY, 28, MARCH, 3
      1, APRIL, 30, MAY, 31, JUNE, 30
5100 DATA JULY, 31, AUGUST, 31, SEPTEMBER, 30
      , OCTOBER, 31, NOVEMBER, 30, DECEMBER, 31
5110
    
```



```

6100 ON CHGR: CX(1) GOSUB 2840: PRINT S1$; "S"; AS;
6200 (1) SL: GOSUB 2860: PRINT S1$; "S"; AS;
6300 HOME: GOSUB 2870: PRINT S1$; "S"; AS;
6400 GOSUB 1680: IF MX = 1 AND TR < 3 THEN
6500 GOSUB 1630: GOTO 790
6600 HEN MX = 0: GOTO 790
6700 MX: GOSUB 2850: HX(1)
6800 (1) HGR: GOSUB 2860: PRINT S1$; "S"; AS;
6900 1680: S1 = 0: THEN = GOSUB 2750: GOTO 680
7000 VX < 22: FL$ = GOSUB 2870: PRINT S1$; "S"; AS;
7100 2870: FN: RD(90) THEN GOSUB 2740: GOTO
7200 OR A > 690: GOSUB 2880: GOSUB 3040: PRINT S2$; "S";
7300 GOSUB 2880: GOSUB 3040: PRINT S2$; "S";
7400 HX(4) = 0: HGR: TEXT: GOSUB 3210:
7500 HX(4) = 0: HGR: TEXT: GOSUB 3210:
7600 IF CH(CX(1)) = 0 OR CH(CX(1)) = 1: RETURN
7700 GOTO 610: : : : : 1: CH(CX(1)) = 1
7800 ON CX(1) = CX(1) GOTO 760, 820: : : : : 1
7900 ON CH(CX(1)) GOTO 760, 820: : : : : 1
8000 HGR: GOSUB 3400: PR = INT (RND (1)
8100 * 12): TR = 3400: GOSUB 3140: GOSUB 3140:
8200 S1 = GOSUB 3010: IF S1 = "AC" THEN
8300 S1 = SL: S2 = SH: SL: S1 = GOSUB 2840: HX(1)
8400 GOSUB 2940: T = A: GOSUB 2840: HX(1)
8500 GOSUB 2860: S2: PRINT S1$; "S"; S1; "S";
8600 S2$; "S2: GOSUB 2870: PRINT "FI
8700 ND: AS: S2: GOSUB 2870: PRINT "FI
8800 GOSUB 1680: GOSUB 2790: GOSUB 1720:
8900 GOSUB 1630: IF MX = 1 AND TR < 3 THEN
9000 HEN MX = 0: GOTO 790
9100 GOTO 920: : : : : 1: CH(CX(1)) = 1
9200 HGR: GOSUB 3400: GOSUB 2850: HX(1)
9300 HOME: : VTAB 21: VX = 21: HTAB 1: PRI
9400 NT S1$; : : : : GOSUB 1680: S1 = 6: FL$
9500 D( VAL (FL$)): : : : : GOSUB 1680: S1 = 6: FL$
9600 GOSUB 2870: PRINT S2$; "S2: GOSUB 1680: S1 = 6: FL$
9700 1680: S2: FN: RD( VAL (FL$)): IF PR
9800 IF S1 > S2 THEN 910 AC: THEN 880
9900 IF S1 > S2 THEN 910 AC: THEN 880
1000 HOME: VTAB 22: HTAB 1: PRINT S2$; "S2:
1010 MUST: BE LESS THAN: S1$; GOTO 900
1020 IF S2 > S1 THEN 910 HTAB 1: PRINT S1$; "S1:
1030 MUST: BE LESS THAN: S2$; S2: GOTO 830
1040 HOME: VTAB 22: HTAB 1: PRINT S1$; "S1:
1050 MUST: BE LESS THAN: S2$; S2: GOTO 830
1060 FOR DI = 1 TO 1000: NEXT: GOTO 830
1070 GOSUB 2940: VTAB 23: HTAB 1: PRINT
1080 AS: = FN: RD(A): GOSUB 2760:
1090 HX(4) = 0: TEXT: : HOME: GOSUB 3210:
1100 X(2) THEN CX(1) = 0 OR CH(CX(1)) = 1: RETURN
1110 : : : : : 1: CH(CX(1)) = 1
1120 GOTO 750: : : : : 1
1130 CX(1) = CX(1) IF CH(CX(1)) = 0 OR CX(1) = CX(1)
1140 GOSUB 3240: IF CH(CX(1)) = 0 OR CX(1) = CX(1)
1150 H(1) RETURN: THEN CX(1) = CX(1)
1160 ON CH(CX(1)) GOSUB 970, 1140: GOTO 9
1170 50: : : : : 1
1180 CX(1) = CX(1) GOTO 990, 1040: : : : : 1
1190 ON CH(CX(1)) GOTO 990, 1040: : : : : 1
1200 HGR: GOSUB 3430: PR = INT (4 * RND (1)
1210 D(1): TR = 3430: GOSUB 3130: GOSUB 3130:
1220 GOSUB 3150: GOSUB 2920: GOSUB 3130:
1230 GOSUB 2840: HTAB 1: PRINT S1$; "S1:
1240 HOME: VTAB 21: HTAB 1: PRINT S1$; "S1:
1250 : : : : : S2$; "S2:
1260 : : : : : S2$; "S2:
1270 : : : : : S2$; "S2:
1280 : : : : : S2$; "S2:
1290 : : : : : S2$; "S2:
1300 : : : : : S2$; "S2:
1310 : : : : : S2$; "S2:
1320 : : : : : S2$; "S2:
1330 : : : : : S2$; "S2:
1340 : : : : : S2$; "S2:
1350 : : : : : S2$; "S2:
1360 : : : : : S2$; "S2:
1370 : : : : : S2$; "S2:
1380 : : : : : S2$; "S2:
1390 : : : : : S2$; "S2:
1400 : : : : : S2$; "S2:
1410 : : : : : S2$; "S2:
1420 : : : : : S2$; "S2:
1430 : : : : : S2$; "S2:
1440 : : : : : S2$; "S2:
1450 : : : : : S2$; "S2:
1460 : : : : : S2$; "S2:
1470 : : : : : S2$; "S2:
1480 : : : : : S2$; "S2:
1490 : : : : : S2$; "S2:
1500 : : : : : S2$; "S2:
1510 : : : : : S2$; "S2:
1520 : : : : : S2$; "S2:
1530 : : : : : S2$; "S2:
1540 : : : : : S2$; "S2:
1550 : : : : : S2$; "S2:
1560 : : : : : S2$; "S2:
1570 : : : : : S2$; "S2:
1580 : : : : : S2$; "S2:
1590 : : : : : S2$; "S2:
1600 : : : : : S2$; "S2:
1610 : : : : : S2$; "S2:
1620 : : : : : S2$; "S2:
1630 : : : : : S2$; "S2:
1640 : : : : : S2$; "S2:
1650 : : : : : S2$; "S2:
1660 : : : : : S2$; "S2:
1670 : : : : : S2$; "S2:
1680 : : : : : S2$; "S2:
1690 : : : : : S2$; "S2:
1700 : : : : : S2$; "S2:
1710 : : : : : S2$; "S2:
1720 : : : : : S2$; "S2:
1730 : : : : : S2$; "S2:
1740 : : : : : S2$; "S2:
1750 : : : : : S2$; "S2:
1760 : : : : : S2$; "S2:
1770 : : : : : S2$; "S2:
1780 : : : : : S2$; "S2:
1790 : : : : : S2$; "S2:
1800 : : : : : S2$; "S2:
1810 : : : : : S2$; "S2:
1820 : : : : : S2$; "S2:
1830 : : : : : S2$; "S2:
1840 : : : : : S2$; "S2:
1850 : : : : : S2$; "S2:
1860 : : : : : S2$; "S2:
1870 : : : : : S2$; "S2:
1880 : : : : : S2$; "S2:
1890 : : : : : S2$; "S2:
1900 : : : : : S2$; "S2:
1910 : : : : : S2$; "S2:
1920 : : : : : S2$; "S2:
1930 : : : : : S2$; "S2:
1940 : : : : : S2$; "S2:
1950 : : : : : S2$; "S2:
1960 : : : : : S2$; "S2:
1970 : : : : : S2$; "S2:
1980 : : : : : S2$; "S2:
1990 : : : : : S2$; "S2:
2000 : : : : : S2$; "S2:
2010 : : : : : S2$; "S2:
2020 : : : : : S2$; "S2:
2030 : : : : : S2$; "S2:
2040 : : : : : S2$; "S2:
2050 : : : : : S2$; "S2:
2060 : : : : : S2$; "S2:
2070 : : : : : S2$; "S2:
2080 : : : : : S2$; "S2:
2090 : : : : : S2$; "S2:
2100 : : : : : S2$; "S2:
2110 : : : : : S2$; "S2:
2120 : : : : : S2$; "S2:
2130 : : : : : S2$; "S2:
2140 : : : : : S2$; "S2:
2150 : : : : : S2$; "S2:
2160 : : : : : S2$; "S2:
2170 : : : : : S2$; "S2:
2180 : : : : : S2$; "S2:
2190 : : : : : S2$; "S2:
2200 : : : : : S2$; "S2:
2210 : : : : : S2$; "S2:
2220 : : : : : S2$; "S2:
2230 : : : : : S2$; "S2:
2240 : : : : : S2$; "S2:
2250 : : : : : S2$; "S2:
2260 : : : : : S2$; "S2:
2270 : : : : : S2$; "S2:
2280 : : : : : S2$; "S2:
2290 : : : : : S2$; "S2:
2300 : : : : : S2$; "S2:
2310 : : : : : S2$; "S2:
2320 : : : : : S2$; "S2:
2330 : : : : : S2$; "S2:
2340 : : : : : S2$; "S2:
2350 : : : : : S2$; "S2:
2360 : : : : : S2$; "S2:
2370 : : : : : S2$; "S2:
2380 : : : : : S2$; "S2:
2390 : : : : : S2$; "S2:
2400 : : : : : S2$; "S2:
2410 : : : : : S2$; "S2:
2420 : : : : : S2$; "S2:
2430 : : : : : S2$; "S2:
2440 : : : : : S2$; "S2:
2450 : : : : : S2$; "S2:
2460 : : : : : S2$; "S2:
2470 : : : : : S2$; "S2:
2480 : : : : : S2$; "S2:
2490 : : : : : S2$; "S2:
2500 : : : : : S2$; "S2:
2510 : : : : : S2$; "S2:
2520 : : : : : S2$
```

Continued


```

1120 HX(4) CH(0) TEXT: HOME: GOSUB 3220
      IF THEN CX(1) = 0 CX(1) - 1: RETURN
1130 GOTO 980: CX(1) = CX(1) + 1: CH(CX(1)) = 1
1140 ON CX(1) GOTO 1160, 1210:
1150 ON CH(CX(1)) GOTO 1160, 1210:
1160 GOSUB 3160: TR = 0: A2 = A: INT (R
      ND(1) * 10) - 1: IF A2 < 0 THEN
1170 HGR: GOSUB 3460: GOSUB 3140: GOSUB
      2930: A = FN RD(A): A2 = FN RD(A2)
      : SL = FN RD(SL): T = SH: GOSUB 2840
      : HX(1) = 9: HOME: GOSUB 2860: PRINT "AB = ";
1180 SL: /BAD = A2: /CBD = A: A
      : GOSUB 2870: PRINT "FIND CD": GOSUB
1190 1680: GOSUB 2790: GOSUB 1720: GOSU
      B 1630: IF MX = 1 AND TR < 3 THEN M
      X = 0: GOTO 1190
1200 GOTO 1290:
1210 HGR: GOSUB 3460: GOSUB 2850: HX(1)
      = 6:
1220 VX = 21: GOSUB 2860: PRINT "AB = ";
      : GOSUB 1680: SL = FN RD(VAL (FL$))
      : IF SL < 0 THEN GOSUB 2750: G
      OTO 1220
1230 VX = 22: HX(1) = 8: FL$ = LEFT$ (BK$
      , HX(2))
1240 GOSUB 2870: PRINT "/BAD = ": GOSUB
      1680: A2 = FN RD(VAL (FL$)): IF A
      2 < 0 OR A2 > 90 THEN GOSUB
1250 2740: GOTO 1240
1260 VX = 23: FL$ = LEFT$ (BK$, HX(2))
      : GOSUB 2880: PRINT "/CBD = ": GOSUB
      1680: A = FN RD(VAL (FL$)): IF A
      0 < 0 OR A > 90 THEN GOSUB 274
      0: GOTO 1260
1270 IF (A2 > A) THEN GOSUB 2830: HO
      ME: GOTO 1220
1280 GOSUB 2830: GOSUB 2890: PRINT "CD =
      : FN RD(SH): GOSUB 2760
1290 : HGR: TEXT: GOSUB 3220: IF CH(C
      X(1)) = 0 OR CH(CX(1)) = CX(2) THEN
      CX(1) = CX(1) - 1: RETURN
1300 GOTO 1150: CX(1) = CX(1) + 1
1310 : CX(1) = CX(1)
1320 H(CX(1)) = CX(2) THEN CX(1) = CX(1)
      : 1: RETURN
1330 ON CH(CX(1)) GOSUB 1340, 1490: GOTO
      1320: CX(1) = CX(1) + 1: CH(CX(1)) = 1
1340 ON CH(CX(1)) GOTO 1360, 1410:
1350 HGR: GOSUB 3430: TR = 0: PR = INT (
      30: GOSUB 3140: GOSUB 3030: GOSUB 31
      910: T = S1: HX(1) = 9: GOSUB 2840:
1370 HOME: VTAB 21: HTAB 1: PRINT S2$;
      : A: SH: DEGREE$ S3$ = SL: AS$ =
1380 GOSUB 2870: PRINT "FIND "; S1$: GOSU
      B 1680
1390 GOSUB 2790: GOSUB 1720: GOSUB 1630:
      IF MX = 1 AND TR < 3 THEN MX = 0:
      GOTO 1380
1400 GOTO 1470:
1410 HGR: GOSUB 3430: GOSUB 2850: HX(1)
      = 8:
1420 GOSUB 2860: PRINT S1$ = ": GOSUB
      1680: SH = FN RD(VAL (FL$)): IF SH
      < 0 THEN GOSUB 2750: GOTO 1420
1430 VX = 22: FL$ = LEFT$ (BK$, 6): GOSUB
      2870: PRINT S3$ = ": GOSUB 1680:
      SL = FN RD(VAL (FL$)): IF SL <
      0 THEN GOSUB 2760: GOTO 1430
1440 VX = 23: FL$ = LEFT$ (BK$, 6): GOSUB
      2880: PRINT AS$ = ": GOSUB 1680: A
      = FN RD(VAL (FL$)): IF A < 0
      OR A > 90 THEN GOSUB 2740: GOTO
      1440
1450 GOSUB 2910: GOSUB 2890: PRINT S1$:
      : GOSUB 2760:
1460 : HGR: TEXT: GOSUB 3230: IF CH(CX
      (1)) = 0 OR CH(CX(1)) = CX(2) THEN
      CX(1) = CX(1) - 1: RETURN
1480 GOTO 1350: CX(1) = CX(1) + 1: CH(CX(1)) = 1
1490 : CX(1) = CX(1)
1500 ON CH(CX(1)) GOTO 1510, 1550:
1510 HGR: GOSUB 3490: TR = 0: GOSUB 3130
      : A = 180: A: GOSUB 3140: GOSUB 315
      0: GOSUB 2910: T = S1: HX(1) = 9: GOS
      UB 2840: VX = 22: HOME: GOSUB 2860:
      PRINT "AB = SH: AB=DC AD=SL AD
      =BC /ABC = A:
1520 GOSUB 2870: PRINT "FIND AC": GOSUB
      1680
1530 GOSUB 2790: GOSUB 1720: GOSUB 1630:
      IF MX = 1 AND TR < 3 THEN MX = 0:
      GOTO 1520:
1540 GOTO 1610:
1550 HGR: GOSUB 3490: GOSUB 2850: HX(1)
      = 8:
1560 GOSUB 2860: PRINT "AB = ": GOSUB 1
      680: SH = FN RD(VAL (FL$)): IF SH
      < 0 THEN GOSUB 2750: GOTO 1560

```

```

1570 VX = 22: FL$ = LEFT$ (BK$, 6): GOSUB
      2870: PRINT "AD = (FL$) = ": IF SL
      L = THEN: GOSUB 2760: GOTO 1570: GOSUB
1580 VX = 23: FL$ = LEFT$ (BK$, 6): GOSUB
      2880: PRINT "/ABC = (FL$) = ": IF
      : A OR A > 180 THEN GOSUB 2740: G
      OTO 1580
1590 GOSUB 2910: GOSUB 2890: PRINT "AC =
      : FN RD(S1):
1600 : HGR: TEXT: GOSUB 3230: IF CH(CX
      (1)) = 0 OR CH(CX(1)) = CX(2) THEN
      CX(1) = CX(1) - 1: RETURN
1620 GOTO 1500:
1630 IF MX = 1 THEN TR = TR + 1: GOSUB 2
      810: IF TR > 2 THEN GOSUB 2800: RE
      TURN
1640 IF MX = 1 THEN RETURN
      : MX = 0: IF FN RD(VR) = FN RD(T)
      THEN: GOSUB 2780: RETURN:
1650 TR = TR + 1: MX = 1: IF TR > 2 THEN
      GOSUB 2800: RETURN:
1660 GOSUB 2770: RETURN:
1670 GOSUB 2770: RETURN:
1680 X = FRE(0): GOSUB 2570: IF IN$ = CX(1)
      ESC$ THEN HGR: TEXT: CX(1) = CX(1)
      - 1: POP
1690 RETURN:
1700 :
1710 :
1720 GOSUB 2490: ST = 1: GOSUB 1870: CP =
      1: DP = 0: K1 = 0: K2 = 0: JP = 0:
1730 IF CP > 0: FV THEN 1850:
1740 IF ET$ > (CP) < 1850: THEN 1760
1750 CP = CP + 1: GOTO 1730:
1760 PC$ = 0: FOR I = 1 TO 5: BH = VA
      L (MIDS (BH$, I, 1)): IF BH = 0 THEN
      1780
1770 ON I GOSUB 1880, 1900, 1920, 1940, 1960
1780 IF PC$ < 0: GOTO 1820: THEN 1800
1790 NEXT I: GOTO 1820:
1800 JP = 0: ON VAL (MIDS (PC$, 1, 1)) G
      OSUB 2050, 2050, 2070, 2050, 2120: IF J
      P THEN 1820:
1810 CP = CP + 1: GOTO 1730:
1820 IF JP THEN 1840
1830 JP = 1:
1840 GOSUB 2470: RETURN:
1850 : IF K1 > 0 THEN 1820
1860 VR = SK(1): RETURN:
1870 : BH$ = MIDS (B$, ST * 5 + 1, 5): RE
      TURN:
1880 GOSUB 2510: IF P = 0 THEN RETURN
1890 PC$ = 1: CHR$ (48 + P): CP = CP
      + 2: ST = 1: GOSUB 1870: RETURN:
1900 IF ET$ (CP) < 20: ST = 1: THEN RETURN
1910 PC$ = 20: ST = 1: GOSUB 1870: RETU
      RN:
1920 IF ET$ (CP) < 30: ST = 2: THEN RETURN
1930 PC$ = 30: ST = 2: GOSUB 1870: RETU
      RN:
1940 GOSUB 2540: IF P = 0 THEN RETURN
1950 PC$ = 40: STR$ (P): ST = 1: GOSUB
      1870: RETURN:
1960 DP = 0: N$ = ET$ (CP): IF (N$ = "9")
      OR (N$ = "0") AND (N$ < "9")
      ) THEN 1980
1970 RETURN:
1980 PC$ = "5"
1990 IF (N$ < "0") AND DP THEN 2010
2000 DP = 1: GOTO 2020:
2010 IF (N$ < "0" OR N$ > "9") THEN 2030
2020 PC$ = PC$ + N$: CP = CP + 1: N$ = ET$
      (CP): IF (N$ = "9") OR (N$ > "0")
      AND N$ < "9") THEN 1990
2030 ST = 2: GOSUB 1870: CP = CP - 1: RET
      URN:
2040 CP = CP - 1: RETURN
2050 K1 = K1 + 1: SK$(K1) = PC$: RETURN:
2060 K2 = K2 + 1: SK$(K2) = V: RETURN:
2070 IF (K1 > 0) AND (SK$(K1) = "20") TH
      EN 2090
2080 JP = 1: RETURN:
2090 K1 = K1 - 1
2100 IF (K1 = 0) OR (SK$(K1) = "20") THE
      N RETURN
2110 ON VAL (MIDS (SK$(K1), 1, 1)) GOSUB
      2130, 2130, 2070, 2300, 2400: RETURN:
2120 V = VAL (MIDS (PC$, 2, LEN (PC$) -
      1)): GOSUB 2430: RETURN:
2130 IF K2 > 0 THEN 2150
2140 JP = 1: RETURN
2150 ON ASC (MIDS (SK$(K1), LEN (SK$(K
      1)), 1)) - 48 GOSUB 2170, 2180, 2190, 2
      200, 2230, 2260: K1 = K1 - 1: IF
      (K1 = 0) OR (SK$(K1) = "20") THEN
      RETURN
2160 GOSUB 2300: RETURN:
2170 SK(K2) = SIN (FN RA(SK(K2))) : RET
      URN:
2180 SK(K2) = COS (FN RA(SK(K2))) : RET
      URN:

```

Continued

TRIG-TRIX *Continued*

APPLE II Family

```

2190 SK (K2) = TAN ( FN RA (SK (K2))) : RET
2200 IF ABS (SK (K2)) < 1 THEN 2220
2210 JF RETURN :
2220 SK (K2) = FN DG ( FN SN (SK (K2))) : RE
2230 IF ABS (SK (K2)) < 1 THEN 2250
2240 JF RETURN :
2250 SK (K2) = FN DG ( FN CS (SK (K2))) : RE
2260 SK (K2) = FN DG ( ATN (SK (K2))) : RET
2270 IF SK (K2) > 0 THEN 2290
2280 JF SK (K2) = 1 : RETURN :
2290 IF SK (K2) > 1 THEN 2320
2300 JF SK (K2) = 1 : RETURN :
2310 ON VAL ( MIDS (SK (K1), 1, 1)) :
2320 GOSUB 2330, 2340, 2350, 2360, 2
380 : K1 = K2 : RETURN
2330 SK (K2 - 1) = SK (K2 - 1) + SK (K2) : R
2340 SK (K2 - 1) = SK (K2 - 1) - SK (K2) : R
2350 SK (K2 - 1) = SK (K2 - 1) * SK (K2) : R
2360 IF SK (K2) > 0 THEN 2390
2370 JF SK (K2) = 1 : RETURN :
2380 SK (K2 - 1) = SK (K2 - 1) ^ SK (K2) : R
2390 SK (K2 - 1) = SK (K2 - 1) / SK (K2) : R
2400 IF K2 < 0 THEN 2420
2410 JF K2 = 1 : RETURN :
2420 K1 = K1 : RETURN :
2430 GOSUB 2440, 2450 :
2440 IF (K1 > 0) * (SK (K1)) :
2450 IF (K1 < 0) * (MIDS (SK (K1), 1, 1)) :
2460 GOSUB 2470, 2480 :
2470 MX = 1 : RETURN :
2480 * * * * *
2490 : FOR IT = 1 TO 50 : ET$ (IT) = " " : N
EXT : FV = LEN (FL$) : FOR IT = 1 TO N
FV : ET$ (IT) = MID$ (FL$, IT, 1) : NEX
IT : RETURN :
2500 FV = LEN (FL$) : FOR IT = 1 TO FV : E
TS (IT) = MID$ (FL$, IT, 1) : NEXT : R
2510 : ET$ (CP) = X : FOR X = 0 TO 2 : X$ = X$ +
ET$ (CP) : NEXT : P = 0 : IF X$ = MID$ (FMS,
3) : THEN P = X + 1 :
2520 : FOR X = 0 TO 6 : IF X$ = MID$ (FMS,
3) : THEN P = X + 1 :
2530 : NEXT : P = 0 : FOR I = 1 TO 5 : IF ET$ (CP)
= MID$ (OPS, I, 1) : THEN P = I :
2540 : NEXT : MID$ (OPS, I, 1) : THEN P = I :
2550 : NEXT : MID$ (OPS, I, 1) : THEN P = I :
2560 : * * * * *
2570 : GOSUB 2720 : VTAB VX : HTAB HX (1) :
FOR IT = 1 TO HX (2) : ET$ (IT) = MID$
(FMS, IT, 1) : NEXT : HX (3) = 1 :
2580 IF HX (3) < 1 THEN HX (3) = 1 :
2590 IF HX (3) < 1 THEN HX (3) = 1 :
2600 ZH = HX (1) : HX (3) = HX (3) + (HX
(3) < 1) :
2610 GET IN$ : IF IN$ = "Z" : ZH = HX (3) :
2620 IF IN$ = "ESC" : THEN RETURN :
2630 IF IN$ = "1" : GOTO 2580 :
2640 IF IN$ = "2" : GOTO 2580 :
2650 IF IN$ = "3" : GOTO 2580 :
2660 IF HX (4) < 0 THEN 2690
2670 IF ASC (IN$) > 32 THEN ET$ (HX (3)
) = IN$ : PRINT IN$ : HX (3) = HX (3)
+ 1 : GOTO 2610 :
2680 IF PRINT BL$ : GOTO 2610 :
2690 IF (IN$ = "0") AND IN$ < "9" :
OR IN$ = " " : OR IN$ = "<" : THEN ET$
(HX (3)) = IN$ : PRINT IN$ : HX (3) = HX
(3) + 1 : GOTO 2580 :
2700 PRINT BL$ : GOTO 2610 :
2710 FL$ = " " : FOR IT = 1 TO HX (2) : FL$ =
FL$ + ET$ (IT) : NEXT : RETURN :
2720 : VTAB VX : HTAB HX (1) : IF HX (2) > H
X (1) : THEN CALL - 868 : PRINT
FL$ : RETURN :
2730 CALL - 868 : PRINT LEFT$ (FL$, 39) -
HX (1) : VTAB VX + 1 : HTAB HX (1) : C
ALL + HX (1) : PRINT RIGHT$ (FL$, HX (2)
- 39) : RETURN :
2740 : INVERSE : PRINT "ANGLE OUT OF RANGE" :
NORMAL : FOR DX = 1 TO 2000 : NE
XT : RETURN :
2750 : INVERSE : PRINT "SIDE MUST BE > 0" :
NORMAL : FOR DX = 1 TO 2000 : NEXT
RETURN :
2760 VC = 24 : HC = 31 : VTAB 24 : HTAB 31 :
INVERSE : PRINT "HIT A KEY" : NORM
AL : GOSUB 3520 : GOSUB 2890 : RETURN

```

```

2770 GOSUB 2890 : INVERSE : PRINT "NO" : T
RY AGAIN : INVERSE : PRINT "CORRE" :
2780 GOSUB 2890 : INVERSE : PRINT "RD (T)" :
CT, GOSUB 2760 : RETURN :
2790 : INVERSE : GOSUB 2890 : HTAB 5 : PRINT RETU
RN : CALCULATING :
2800 GOSUB 2890 : INVERSE : PRINT "THE A" :
NSWER IS : GOSUB 2760 : RETURN :
2810 GOSUB 2890 : PRINT BL$ : BL$ = INVERSE :
: PRINT "SYNTAX ERROR" : NORMAL :
2820 GOTO 2800 :
2830 VTAB 24 : HTAB 5 : CALL - 868 : INVER
SE : PRINT "ILLEGAL ANGLE VALUES" :
: NORMAL : GOSUB 2760 : RETURN :
2840 : TR = 0 : HX (2) = 50 : FL$ = LEFT$ (BK
$, HX (2)) : HX (4) = 0 : VX = 22 : RETURN
2850 : VX = 21 : HX (4) = 1 : HX (2) = 6 : FL$ =
LEFT$ (BK$, HX (4)) : RETURN :
2860 VTAB 21 : GOSUB 2890 :
2870 VTAB 22 : GOSUB 2890 :
2880 VTAB 23 : GOSUB 2890 :
2890 VTAB 24 : GOSUB 2890 :
2900 : HTAB 1 : CALL - 868 : POP : RETURN
2910 : S1 = SQR ((SH ^ 2) + (SL ^ 2)) - (
2 : S1 = SH : SL = COS ( FN RA (A)) : R
ETURN :
2920 : X = FN SL : SIN ( FN RA (A)) / SH : A2 =
999999 : DG ( FN SN (X) : RETURN :
2930 : SH = FN SL : SIN ( FN RA (A)) :
N ( FN RA (A)) : SIN ( FN RA (A)) - S1
A2 :
2940 : ON PR + 1 GOSUB 2950, 2950, 2960, 29
60, 2970, 2970, 2980, 2980, 2990, 2990, 30
00, 3000 :
2950 : A = FN DG ( FN SN (S2 / S1)) : RET
URN :
2960 : A = FN DG ( FN SN (S1 / S2)) : RET
URN :
2970 : A = FN DG ( FN CS (S2 / S1)) : RET
URN :
2980 : A = FN DG ( FN CS (S1 / S2)) : RET
URN :
2990 : A = FN DG ( ATN (S2 / S1)) : RETU
RN :
3000 : A = FN DG ( ATN (S1 / S2)) : RETU
RN :
3010 : S1$ = MID$ (PR$ (1), PR * 8 + 1, 2) :
A$ = MID$ (PR$ (1), PR * 8 + 3, 4) : S
2$ = MID$ (PR$ (1), PR * 8 + 7, 2) : R
ETURN :
3020 : S1$ = MID$ (PR$ (2), PR * 12 + 1, 2) :
A$ = MID$ (PR$ (2), PR * 12 + 3, 4) : S
2$ = MID$ (PR$ (2), PR * 12 + 7, 2) : R
ETURN :
3030 : S2$ = MID$ (PR$ (3), PR * 10 + 1, 2) :
S3$ = MID$ (PR$ (3), PR * 10 + 3, 2) :
S1$ = MID$ (PR$ (3), PR * 10 + 5, 2) :
A$ = MID$ (PR$ (3), PR * 10 + 7, 4) :
RETURN :
3040 : PR = PR + 1 : GOSUB 3050, 3050, 3060, 3
060, 3070, 3070, 3080, 3080, 3090, 3090, 3
100, 3100 :
3050 : S2 = S1 * SIN ( FN RA (A)) : RETUR
N :
3060 : S2 = S1 / SIN ( FN RA (A)) : RETUR
N :
3070 : S2 = S1 * COS ( FN RA (A)) : RETUR
N :
3080 : S2 = S1 / COS ( FN RA (A)) : RETUR
N :
3090 : S2 = S1 * TAN ( FN RA (A)) : RETUR
N :
3100 : S2 = S1 / TAN ( FN RA (A)) : RETUR
N :
3110 : A = FN DG ( FN SN (S1 / S2)) : RETU
RN :
3120 : A = INT ( RND (1) * 89) + 1 : RET
URN :
3130 : SL = INT ( RND (1) * 20) + 1 : RE
TURN :
3140 : SH = INT ( RND (1) * 20) + SL +
1 : RETURN :
3150 : A = INT ( RND (1) * 75) + 11 : RE
TURN :
3160 : CALL PWNT : FOR PX = 1 TO LEN (
PS$) : PZ$ = CHR$ (ASC ( MID$ (PS$,
PX, 1))) : IF PZ$ < " " : OR PZ$ >
" " : THEN PZ$ = NEXT : CALL OFF : RETURN
3170 :
3180 : CX (2) = 4 : CX (3) = 1 : CX (4) = 0 : GOS
UB 3280 : RETURN :
3190 : CX (2) = 3 : CX (3) = 5 : CX (4) = 1 : GOSU
B 3280 : RETURN :
3200 : CX (2) = 3 : CX (3) = 8 : CX (4) = 1 : GOSU
B 3280 : RETURN :
3210 : CX (2) = 3 : CX (3) = 8 : CX (4) = 1 : GOSU
B 3280 : RETURN :

```

Continued

APPLE // Family

[illegible]

COMMODORE 64

```

420 ITT="RIGHT TRIANGLES":GOSUB2730:PRIN
430 NT"CRSRDOWN"1.DETERMINE SIDES:"PRINT"3.
440 RETURN TO MAIN MENU":GOSUB2600
450 GOSUB2320:IFK<49ORK>51THEN440
460 PRINTK$:IFK=51THENRETURN
470 GOSUB2680
480 ONK-48GOSUB660,780
490 GOTO420
500 ITT="LAW OF SINES":GOSUB2730:PRINT"
510 T"CRSRDOWN"1.DRILL":PRINT"2. CHAL
520 LLENGE":PRINT"3. RETURN TO MAIN MENU":GOSUB
530 2600
540 GOSUB2320:IFK<49ORK>51THEN520
550 PRINTK$:IFK=51THENRETURN
560 GOSUB2680
570 ONK-48GOSUB1190,1330
580 GOTO490
590 ITT="LAW OF COSINES":GOSUB2730:PRIN
600 T"CRSRDOWN"1.DRILL":PRINT"2. CHAL
610 LLENGE":PRINT"3. RETURN TO MAIN MENU":GOSUB
620 2600
630 GOSUB2320:IFK<49ORK>51THEN590
640 PRINTK$:IFK=51THENRETURN
650 GOSUB2680
660 ONK-48GOSUB2370,2490
670 GOTO570
680 GOSUB2730:GOSUB2860:RETURN
690 GOSUB2730:GOSUB3090:RETURN
700 PR=INT(RND(1)*12):GOSUB2690:GOSUB27
710 00:S1=SL:TR=Z
720 GOSUB760:GOSUB1040:GOSUB640
730 PRINT"CRSRDOWN"1.S1$:"=";S1:PRINTA
740 S$:"=";A:"DEGREES":PRINT"FIN D ";S2$;
750 TR=S2:GOSUB1130
760 IFK=49THEN660
770 IFK=51THENRETURN
780 GOSUB640:PRINT"CRSRDOWN"1.S1$:"=";
790 :GOSUB3330:S1=X1
800 PRINTA$:"=";GOSUB3330:A=X1
810 GOSUB760:PRINTS2$:"=";FN RD(S2):GOSU
820 B2340:GOSUB1080
830 GOTO700
840 ONPR+EGOSUB2610,2610,2620,2620,2630
850 ,2630,2640,2640,2650,2650,2660,2660
860 RETURN

```

Continued


```

780 PR=INT((RND(1)*12)+1):TR=Z:GOSUB2700:GO
SUB2710:GOSUB1040
790 IF S1<>V:AC=THEN810
800 S1=SH:S2=SL:GOTO820
810 S1=SL:S2=SH
820 GOSUB960:GOSUB640:PRINT"CRSRDOWN"
S1$:S2$:S1:PRINTS2$:S2:
830 PRINT"FIN"
840 IF K=49 THEN780
850 IF K=51 THEN970
860 GOSUB640:PRINTS1$:"="::GOSUB3330:S1
=X1
870 PRINTS2$:"="::GOSUB3330:S2=X1
880 IF P=7 THEN950
890 IF S1<>V:AC=THEN920
900 IF S1>S2 THEN950
910 PRINTS2$:S2:MUST BE LESS THAN":S1$:
GOTO940
920 IF S2>S1 THEN950
930 PRINTS1$:S1:MUST BE LESS THAN":S2$:
940 GOSUB2340:GOTO860
950 GOSUB960:PRINTAS:"=":FNRD(A):GOSUB2
340:GOSUB1080:GOTO840
960 ON PR+K GOSUB980,980,990,990,1000,100
0,1010,1010,1020,1020,1030,1030
970 RETURN
980 A=FNDE(FNAN(S2/S1)):RETURN
990 A=FNDE(FNAN(S1/S2)):RETURN
1000 A=FNDE(FNAS(S2/S1)):RETURN
1010 A=FNDE(FNAS(S1/S2)):RETURN
1020 A=FNDE(ATN(S2/S1)):RETURN
1030 A=FNDE(ATN(S1/S2)):RETURN
1040 RP$=P1$(PR):S1$=MID$(RP$,E,2)
1050 AS=MID$(RP$,3,4):S2$=MID$(RP$,7,2)::
RETURN
1060 PRINT"NOT CORRECT, TRY AGAIN.":RETU
RN
1070 PRINT"NO,":RETURN
1080 PRINT"SHIFT CLR=5CRSRDOWN1. DO A
PROBLEM OF THIS TYPE":PRINT"CRSRD
OWN2. ENTER VALUES
1090 PRINT"CRSRDOWN3. EXIT"
1100 GOSUB2600
1110 GOSUB2320:IF K<49 OR K>51 GOTO1110
1120 RETURN
1130 GOSUB3430:GOSUB1460
1140 IF FNRD(VR)=FNRD(T) THEN1180
1150 TR=TR+E:IF TR>2 THEN1170
1160 GOSUB1060:GOTO1130
1170 GOSUB1070:PRINTFNRD(T):"IS CORRECTI
":GOSUB2340:GOSUB1080:RETURN
1180 PRINTFNRD(T):"IS CORRECTI":GOSUB234
0:GOSUB1080:RETURN
1190 PR=INT(RND(1)*4):TR=Z:GOSUB2690:GOS
UB2700:GOSUB2710
1200 GOSUB1310:GOSUB1290:GOSUB650
1210 PRINT"CRSRDOWN":S1$:"=":SH:PRINTS
2$:S2$:S1:PRINTAS:"=":A:"DEGREES"
1220 PRINT"FIN"
1230 IF K=49 THEN1190
1240 IF K=51 THENRETURN
1250 IV=0:GOSUB650:PRINT"CRSRDOWN":S1$
:"="::GOSUB3330:SH=X1:PRINTS2$:S2$:
GOSUB3330:SH=X1:PRINTS2$:S2$:
1260 SL=X1:PRINTAS:"="::GOSUB3330:A=X1:G
OSUB1310:IF IV=0 THEN1280
1270 PRINT"CTRL RVSON=ILLEGAL VALUES-
TRY AGAIN":GOSUB2340:GOTO1250
1280 PRINTAS2$:S2$:FNRD(A2):GOSUB2340:GOS
UB1080:GOTO1230
1290 RP$=P2$(PR):S1$=MID$(RP$,E,2):AS=MI
D$(RP$,3,4)
1300 S2$=MID$(RP$,7,2):A2$=MID$(RP$,9,4)
:RETURN
1310 IF ABS(SL*SIN(FNRD(A))/SH)>1 THEN I
V=1:RETURN
1320 A2=FNDE(FNAN(SL*SIN(FNRA(A))/SH)):R
ETURN
1330 TR=Z:GOSUB2720:A2=A-INT(RND(1)*10)-
E:GOSUB2700
1340 GOSUB1450:GOSUB2730:GOSUB2960
1350 PRINTAB="":SL:PRINT"CMDR E=BAD":A
2:PRINT"CMDR E=CBD":A:PRINT"FIN"
CD:
1360 T=SH:GOSUB1130
1370 IF K=49 THEN1330
1380 IF K=51 THENRETURN
1390 GOSUB2730:GOSUB2960
1400 PRINTAB="":GOSUB3330:SL=X1:PRINT"
CMDR E=BAD":GOSUB3330:A2=X1
1410 PRINT"CMDR E=CBD":GOSUB3330:A=X1
1420 IF (A<90) OR (A2<A) THEN1440
1430 PRINT"ILLEGAL ANGLE VALUES.":GOSUB2
340:GOTO1390
1440 GOSUB1450:PRINT"CD=":FNRD(SH):GOSUB
2340:GOSUB1080:GOTO1370
1450 SH=SL*SIN(FNRA(A2))*SIN(FNRA(A))/SI
N(FNRA(A-A2)):RETURN
1460 S5=E:GOSUB1610:CP=E:DF=Z:KS(1)=Z:KS
(2)=Z:JP=Z
1470 IF CP>LEN(IN$) THEN1500
1480 X=CP:GOSUB2740:IF EQ$<>CHR$(32) THEN1
500
1490 CP=CP+E:GOTO1470
1500 PC$="00":FOR I=ETOS:BH=VAL(MID$(BH$,
I,E))
1510 IF BH=Z THEN1530
1520 ON I GOSUB1620,1670,1690,1710,1760

```

```

1530 IF PC$<>"00" THEN1550
1540 NEXT I:GOTO1580
1550 JP=Z:ON VAL(MID$(PC$,E,E)) GOSUB1860,
1860,1880,1860,1930
1560 IF JP THEN1580
1570 CP=CP+E:GOTO1470
1580 JP=1:PRINT"SYNTAX ERROR":RETURN
1590 IF KS(1)>Z THEN1580
1600 VR=SK(E):RETURN
1610 BH$=MID$(BH$,S5*5+1,5):RETURN
1620 Q=E:P=Z:IF (LEN(IN$)-CP)<3 THEN1650
1630 IF MID$(FM$,Q,3)=MID$(IN$,CP,3) THENP
=Q:GOTO1650
1640 Q=Q+3:IF Q<20 THEN1630
1650 IF P=Z THENRETURN
1660 PC$="1"+CHR$(49+(INT(P/3))):CP=CP+2
:S5=Z:GOSUB1610:RETURN
1670 X=CP:GOSUB2740:IF EQ$<>("THENRETURN
1680 PC$="20":S5=E:GOSUB1610:RETURN
1690 X=CP:GOSUB2740:IF EQ$<>("THENRETURN
1700 PC$="30":S5=2:GOSUB1610:RETURN
1710 Q=E:P=Z:X=CP:GOSUB2740
1720 IF MID$(OP$,Q,E)=MID$(IN$,CP,E) THENP
=Q:GOTO1740
1730 Q=Q+1:IF Q<6 THEN1720
1740 IF P=Z THENRETURN
1750 PC$="4"+STR$(P):S5=E:GOSUB1610:RETU
RN
1760 DF=Z:X=CP:GOSUB2740:N$=EQ$
1770 IF (N$="")+(N$>="0")*(N$<="9")) THE
N1790
1780 RETURN
1790 PC$="5"
1800 IF (N$<>".")+DP THEN1820
1810 DF=E:GOTO1830
1820 IF (N$<"0")+(N$>="9") THEN1850
1830 PC$="6"+N$:CP=CP+E:X=CP:GOSUB2740:N
$=EQ$
1840 IF (N$="")+(N$>="0")*(N$<="9")) THE
N1800
1850 S5=2:GOSUB1610:CP=CP-E:RETURN
1860 KS(1)=KS(1)+E:SK$(KS(1))=PC$:RETURN
1870 KS(2)=KS(2)+E:SK$(KS(2))=V:RETURN
1880 IF (KS(1)>Z)*(SK$(KS(1))="20") THEN19
00
1890 JP=E:RETURN
1900 KS(1)=KS(1)-E:IF (KS(1)=Z)+(SK$(KS(1
))="20") THENRETURN
1910 ON VAL(MID$(SK$(KS(1)),E,E)) GOSUB194
0,1940,1880,2140,2250
1920 RETURN
1930 V=VAL(MID$(PC$,2,LEN(PC$)-E)):GOSUB
2280:RETURN
1940 IF KS(2)>Z THEN1960
1950 JP=E:RETURN
1960 XC=ASC(MID$(SK$(KS(1)),LEN(SK$(KS(1
))),E))-48
1970 ON XC GOSUB2010,2020,2030,2040,2070,
2100,2110
1980 KS(1)=KS(1)-E
1990 IF (KS(1)=Z)+(SK$(KS(1))="20") THENRE
TURN
2000 GOSUB2140:RETURN
2010 SK(KS(2))=SIN(FNRA(SK(KS(2)))):RETU
RN
2020 SK(KS(2))=COS(FNRA(SK(KS(2)))):RETU
RN
2030 SK(KS(2))=TAN(FNRA(SK(KS(2)))):RETU
RN
2040 IF ABS(SK(KS(2)))<=E THEN2060
2050 JP=E:RETURN
2060 SK(KS(2))=FNDE(FNAN(SK(KS(2)))):RET
URN
2070 IF ABS(SK(KS(2)))<=E THEN2090
2080 JP=E:RETURN
2090 SK(KS(2))=FNDE(FNAS(SK(KS(2)))):RET
URN
2100 SK(KS(2))=FNDE(ATN(SK(KS(2)))):RETU
RN
2110 IF SK(KS(2))>Z THEN2130
2120 JP=E:RETURN
2130 SK(KS(2))=SQB(SK(KS(2))):RETURN
2140 IF KS(2)>E THEN2160
2150 JP=E:RETURN
2160 ON VAL(MID$(SK$(KS(1)),LEN(SK$(KS(1
))),E)) GOSUB2180,2190,2200,2210,2230
2170 KS(1)=KS(1)-E:KS(2)=KS(2)-E:RETURN
2180 SK(KS(2))-E=SK(KS(2))-E+SK(KS(2)):R
ETURN
2190 SK(KS(2))-E=SK(KS(2))-E-SK(KS(2)):R
ETURN
2200 SK(KS(2))-E=SK(KS(2))-E*SK(KS(2)):R
ETURN
2210 IF SK(KS(2))>Z THEN2240
2220 JP=E:RETURN
2230 SK(KS(2))-E=SK(KS(2))-E!SK(KS(2)):R
ETURN
2240 SK(KS(2))-E=SK(KS(2))-E/SK(KS(2)):R
ETURN
2250 IF SK(2)>Z THEN2270
2260 JP=E:RETURN
2270 KS(1)=KS(1)-E:RETURN
2280 GOSUB1870:IF (KS(1)>Z)*(SK$(KS(1))<
"20") THEN2300
2290 RETURN
2300 IF (KS(1)=Z)+(MID$(SK$(KS(1)),E,E)<
"4") THENRETURN
2310 GOSUB2140:RETURN

```

Continued


```

2320 GET K$:IF K$="GOTO 2320
2330 K=ASC(K$):RETURN
2340 PRINT "CTRL R DOWN OFF TO CONTINUE"
2350 GOSUB 2320:IF K$<>"13" THEN 2350
2360 RETURN
2370 PR=INT((RND(1)+3):GOSUB 2470:TR=Z:GOS
2380 UB 2690:GOSUB 2700
2390 GOSUB 2710:GOSUB 2590:GOSUB 650
2400 PRINT "CRSR DOWN":S2$="":SH:PRINTS
2410 S$="":SL:PRINTA$:"=":";A:
2420 PRINT "FIND":S1$:"=":";T=S1:GOSUB 1130
2430 IF K$=49 THEN 2370
2440 IF K$=51 THEN RETURN
2450 GOSUB 650:PRINT "CRSR DOWN":S2$="=":";
2460 GOSUB 3330:SH=X1
2470 PRINTS$:"=":";GOSUB 3330:SL=X1
2480 PRINTA$:"=":";GOSUB 3330:A=X1:GOSUB 25
2490 90
2500 PRINTS1$:"=":";FNRD(S1):GOSUB 2340:GOS
2510 UB 1080:GOTO 2410
2520 RPS=P3$(PR):S2$=MID$(RPS,E,2):S3$=M
2530 IDS(RPS,3,2)
2540 S1$=MID$(RPS,5,2):A$=MID$(RPS,7,4):
2550 RETURN
2560 TR=Z:GOSUB 2690:A=180-A:GOSUB 2700:GO
2570 SUB 2710:GOSUB 2590
2580 GOSUB 2570:PRINT "AB=":";SH;" AB=DC":PR
2590 INT "AD=":";SL:"AD=BC":
2600 PRINT "CMRDR E ABC=":";A:PRINT "FIND AC
2610 "":T=S1:GOSUB 1130
2620 IF K$=49 THEN 2490
2630 IF K$=51 THEN RETURN
2640 GOSUB 2570:PRINT "AB=":";GOSUB 3330:SH=
2650 X1:PRINT "AD=":";GOSUB 3330:SL=X1
2660 PRINT "CMRDR E ABC=":";GOSUB 3330:A=X1
2670 GOSUB 2590:PRINT "AC=":";FNRD(S1):GOSUB 2
2680 340:GOSUB 1080:GOTO 2520
2690 GOSUB 2730:GOSUB 3190
2700 RETURN
2710 S1=SQR((SH/2)+(SL/2)-(2*SH*SL+COS(F
2720 NRA(A)))):RETURN
2730 PRINT "3 CRSR DOWN":CRSRRIGHT:YOUR CH
2740 OICE="":RETURN
2750 S2=S1/SIN(FNRA(A)):RETURN
2760 S2=S1/SIN(FNRA(A)):RETURN
2770 S2=S1/COS(FNRA(A)):RETURN
2780 S2=S1/COS(FNRA(A)):RETURN
2790 S2=S1/TAN(FNRA(A)):RETURN
2800 S2=S1/TAN(FNRA(A)):RETURN
2810 A=FNDE(FNAN(S1/S2)):RETURN
2820 PRINT "CRSR DOWN":CALCULATING . . .":R
2830 RETURN
2840 A=INT((RND(1)*89)+E:RETURN
2850 SL=INT((RND(1)*20)+E:RETURN
2860 SH=INT((RND(1)*20)+SL+E:RETURN
2870 A=INT((RND(1)*75)+11:RETURN
2880 PRINT "SHIFT CLR":TAB(20-LEN(TTS)/
2890 2):TTS:RETURN
2900 EQ$=MID$(INS,X,1):RETURN
2910 DATA 095,01,002,004,008,016,032,064,
2920 128,097,003,014,052,200,016,032,064,
2930 128,098,01,003,006,010,020,036,072,
2940 136,102,008,008,004,004,002,002,001
2950 001
2960 DATA 104,128,128,064,064,032,032,16,
2970 016,105,003,013,054,194,004,004,008
2980 008
2990 DATA 106,255,128,128,128,128,128,128
3000 128,107,03,012,024,192,000,000,000
3010 000
3020 DATA 108,000,000,000,000,000,03,012,
3030 048,109,001,001,002,002,004,004,008
3040 008
3050 DATA 110,16,016,032,032,064,064,128,
3060 128,111,016,016,032,032,067,076,208
3070 192
3080 DATA 112,03,012,048,192,192,048,012,
3090 003,113,000,004,008,016,032,064,252
3100 000
3110 DATA-1,"AC"CMRDR E"ABACBC","AC"CMRDR E
3120 "ACBABC","BC"CMRDR E"ABACAC","AB"CMRDR
3130 E"ACBAC","AC"CMRDR E"ABACAB","AC"CMRDR
3140 E"ACBEC"
3150 DATA "AB"CMRDR E"ABACAC","BC"CMRDR E"AB
3160 CBAC","AB"CMRDR E"ABACBC","BC"CMRDR E"
3170 ACBABC","BC"CMRDR E"ABACAB","AB"CMRDR E
3180 "ACBEC"
3190 DATA "AB"CMRDR E"ABACAC"CMRDR E"ABAC","
3200 "AB"CMRDR E"ABACBC"CMRDR E"ABAC","BC"CMR
3210 DR E"ABACAC"CMRDR E"ABAC","BC"CMRDR E"BA
3220 CAC"CMRDR E"ABAC"
3230 DATA "ABBCAC"CMRDR E"ABAC","ABACBC"CM
3240 DR E"ABAC","BCACAB"CMRDR E"ABAC"
3250 PRINTTAB(19);
3260 PRINTTAB(16);
3270 G
3280 PRINTTAB(14);
3290 RRIGHT:CMRDR
3300 PRINTTAB(12);
3310 RRIGHT:CMRDR
3320 PRINTTAB(10);
3330 RRIGHT:CMRDR
3340 PRINTTAB(8);
3350 RRIGHT:CMRDR
3360 PRINTTAB(6);
3370 RRIGHT:CMRDR

```

```

2930 PRINTTAB(4);
2940 RRIGHT:CMRDR
2950 PRINT "2 CRSR RIGHT:CMRDR D:CMRDR Q:12 CRS
2960 13 CRSR RIGHT:CMRDR N:CMRDR G:
2970 PRINT "A":16CMRDR T:"B":RETURN
2980 PRINTTAB(17);
2990 "D":
3000 PRINTTAB(15);
3010 "CMRDR I:CMRDR G:
3020 PRINTTAB(14);
3030 "CMRDR S:CMRDR
3040 PRINTTAB(13);
3050 "CMRDR Z:CRSR
3060 RIGHT:CMRDR G:
3070 PRINTTAB(12);
3080 "CMRDR CRSRRIGHT:CMRDR S:CRSRRIGHT:CMRDR G:
3090 PRINTTAB(11);
3100 "CMRDR CRSRRIGHT:CMRDR
3110 MDR Z:2CRSRRIGHT:CMRDR G:
3120 PRINTTAB(10);
3130 "CMRDR 2CRSRRIGHT:CMRDR
3140 CMRDR S:2CRSRRIGHT:CMRDR G:
3150 PRINTTAB(9);
3160 "CMRDR 2CRSRRIGHT:CMRDR
3170 MDR Z:3CRSRRIGHT:CMRDR G:
3180 PRINTTAB(8);
3190 "CMRDR 3CRSRRIGHT:CMRDR
3200 MDR S:3CRSRRIGHT:CMRDR G:
3210 PRINTTAB(7);
3220 "CMRDR 3CRSRRIGHT:CMRDR
3230 MDR Z:4CRSRRIGHT:CMRDR G:
3240 PRINTTAB(6);
3250 "CMRDR 4CRSRRIGHT:CMRDR
3260 MDR S:3CRSRRIGHT:CMRDR N:CMRDR G:
3270 PRINTTAB(5);
3280 "A":10CMRDR T:"B":
3290 PRINTTAB(4);
3300 "C":RETURN
3310 PRINTTAB(3);
3320 "A":
3330 PRINTTAB(2);
3340 "CMRDR D:CMRDR Q:2 CRS
3350 RRIGHT:CMRDR
3360 PRINTTAB(1);
3370 "CMRDR D:CMRDR Q:5 CRS
3380 RRIGHT:CMRDR
3390 PRINTTAB(0);
3400 "CMRDR D:CMRDR Q:7 CRS
3410 RRIGHT:CMRDR
3420 PRINTTAB(0);
3430 "CMRDR D:CMRDR Q:10 CRS
3440 RRIGHT:CMRDR
3450 PRINTTAB(0);
3460 "CMRDR D:CMRDR Q:12 CRS
3470 RRIGHT:CMRDR
3480 PRINTTAB(0);
3490 "CMRDR D:CMRDR Q:15 CRS
3500 RRIGHT:CMRDR
3510 PRINTTAB(0);
3520 "CMRDR D:CMRDR Q:17 CRS
3530 RRIGHT:CMRDR
3540 PRINT "CRSR RIGHT:CMRDR 20CMD
3550 R T:"B":RETURN
3560 PRINTTAB(6);
3570 "D":18CMRDR @:CRSRRIGHT
3580 "C"
3590 PRINTTAB(7);
3600 "CMRDR Z:16CRSRRIGHT:
3610 CMRDR D:SHIFT
3620 PRINTTAB(7);
3630 "CMRDR S:14CRSRRIGHT:
3640 CMRDR D:CRSRRIGHT:CMRDR S:
3650 PRINTTAB(6);
3660 "CMRDR Z:13CRSRRIGHT:
3670 CMRDR D:CMRDR Q:2CRSRRIGHT:CMRDR Z:
3680 CMRDR D:CMRDR Q:4CRSRRIGHT:CMRDR S:
3690 PRINTTAB(6);
3700 "CMRDR S:11CRSRRIGHT:
3710 CMRDR D:CMRDR Q:4CRSRRIGHT:CMRDR S:
3720 PRINTTAB(5);
3730 "CMRDR Z:10CRSRRIGHT:
3740 CMRDR D:CMRDR Q:5CRSRRIGHT:CMRDR Z:
3750 CMRDR D:CMRDR Q:7CRSRRIGHT:CMRDR S:
3760 PRINTTAB(4);
3770 "CMRDR Z:7CRSRRIGHT:CMRDR S:
3780 MDR D:CMRDR Q:8CRSRRIGHT:CMRDR Z:
3790 PRINTTAB(4);
3800 "CMRDR S:5CRSRRIGHT:CMRDR
3810 MDR D:CMRDR Q:10CRSRRIGHT:CMRDR S:
3820 PRINTTAB(3);
3830 "CMRDR Z:4CRSRRIGHT:CMRDR
3840 MDR D:CMRDR Q:11CRSRRIGHT:CMRDR Z:
3850 PRINTTAB(3);
3860 "CMRDR S:2CRSRRIGHT:CMRDR
3870 MDR D:CMRDR Q:13CRSRRIGHT:CMRDR S:
3880 PRINTTAB(2);
3890 "CMRDR Z:CRSRRIGHT:CMRDR
3900 DR D:CMRDR Q:14CRSRRIGHT:CMRDR Z:
3910 PRINTTAB(2);
3920 "CMRDR P:CMRDR Q:16CRS
3930 RRIGHT:CMRDR S:
3940 PRINT "AB:CRSRRIGHT:18CMRDR T:CRSRRI
3950 GHT:"B":RETURN
3960 X$="
3970 PRINT "CMRDR @:2SHIFT CRSRLEFT:";
3980 GETX1$:IF X1$="":THEN 3350
3990 IFASC(X1$)=13ANDX$<>"":THENX1=VAL(X$
4000 ):GOTO 3410
4010 IFASC(X1$)=20ANDX$<>"":THENPRINT "SH
4020 IFT CRSRLEFT:SHIFT CRSRLEFT:";X$
4030 =LEFT$(X$,LEN(X$)-1):GOTO 3340
4040 IFLEN(X$)>9THEN 3350
4050 IFASC(X1$)<48ORASC(X1$)>57THEN 3350
4060 X$=X$+X1$:PRINTX1$;GOTO 3340
4070 IFX1=0THENPRINT "PRINT MUST HAVE
4080 A VALUE":GOTO 3330
4090 PRINT "":RETURN
4100 INS="":PRINT "
4110 PRINT "CMRDR @:2SHIFT CRSRLEFT:";
4120 GETX1$:IF X1$="":THEN 3450
4130 IFASC(X1$)=13THENPRINT "":GOTO 2680
4140 IFASC(X1$)=20ANDINS<>"":THENPRINT "S
4150 HIFT CRSRLEFT:SHIFT CRSRLEFT:";1
4160 NS=LEFT$(INS,LEN(INS)-1):GOTO 3440
4170 IFLEN(INS)>50THEN 3450
4180 IFASC(X1$)<32ORASC(X1$)>94THEN 3450
4190 INS=INS+X1$:PRINTX1$;GOTO 3440
4200

```

SCREENS TYPE-IN LISTINGS


```

870 CLS:LOCATE 2,8:PRINT T1$:PSET (60,8
5):DRAW PARALLEL$S:LOCATE 11,6:PRINT
A:LOCATE 1,29:PRINT C:LOCATE 11,26:PRINT B
880 CLS:PSET (60,100):DRAW HEIGHT$:LOCA
TE 13,6:PRINT A:LOCATE 1,28:PRINT
C:LOCATE 13,31:PRINT B
890 ON PR+1 GOTO 900,900,910,910,920,92
0,930,930,940,940,950,950
900 S2=S1/SIN(FNRAD(A)):RETURN
910 S2=S1/SIN(FNRAD(A)):RETURN
920 S2=S1/COS(FNRAD(A)):RETURN
930 S2=S1/COS(FNRAD(A)):RETURN
940 S2=S1/TAN(FNRAD(A)):RETURN
950 S2=S1/TAN(FNRAD(A)):RETURN
960 ON PR+1 GOTO 970,970,980,980,990,99
0,1000,1000,1010,1010,1020,1020
970 A=FNDGEG(FNASN(S2/S1)):RETURN
980 A=FNDGEG(FNASN(S1/S2)):RETURN
990 A=FNDGEG(FNACS(S2/S1)):RETURN
1000 A=FNDGEG(FNACS(S1/S2)):RETURN
1010 A=FNDGEG(ATN(S2/S1)):RETURN
1020 A=FNDGEG(ATN(S1/S2)):RETURN
1030 A2=FNDGEG(FNASN(SL*SIN(FNRAD(A))/SH)
:RETURN
1040 S1=SQR((SH^2)+(SL^2)-(2*SH*SL*COS(F
NRAD(A))):RETURN
1050 A=INT(RND*89)+1:RETURN
1060 SL=INT(RND*20)+1:RETURN
1070 SH=INT(RND*20)+1:RETURN
1080 A=INT(RND*75)+11:A2=A-INT(RND*10)-1
:RETURN
1090 IF K$=ESC$:THEN IN$="":GOTO 1530
1100 ST=1:GOSUB 1180:CP=1:DP=0:SK1=0:SK2
=0:JP=0
1110 IF CP>LEN(IN$) THEN 1170 ELSE IF FN
EQ$(CP)=SP$ THEN CP=CP+1:GOTO 1110
1120 PC$="00":FOR I=1 TO 5:BH=VAL(MID$(B
H$,I,1)):IF BH THEN ON I GOSUB 1190
1130 IF PC$="00" THEN NEXT I:GOTO 1150 E
LSE JP=0:ON VAL(MID$(PC$,1,1)) GOSU
B 1100,1100,1110,1100,1130
1140 IF JP=0 THEN CP=CP+1:GOTO 1110
1150 JP=1:BEep:PRINT "SYNTAX ERROR":TI=T
IMER+3
1160 IF TIMER>TI THEN RETURN ELSE 1160
1170 IF SK1>0 THEN RETURN ELSE VR=SK(1):RE
TURN
1180 BH$=MID$(B$,ST*5+1,5):RETURN
1190 P=1:INSTR(1, FUNC$, MID$(IN$, CP, 3)):IF
P=0 THEN RETURN ELSE PC$="1"+CHR$(4
9+INT(P/3)):CP=CP+2:ST=0:GOSUB 11
80:RETURN
1200 IF FNEQ$(CP)<>"(" THEN RETURN ELSE
PC$="20":ST=1:GOSUB 1180:RETURN
1210 IF FNEQ$(CP)<>")" THEN RETURN ELSE
PC$="30":ST=2:GOSUB 1180:RETURN
1220 P=INSTR(1, OP$, FNEQ$(CP)):IF P=0 THE
N RETURN ELSE PC$="4"+RIGHT$(STR$(P
),LEN(STR$(P))-1):ST=1:GOSUB 1180:R
ETURN
1230 DP=0:N$=FNEQ$(CP):IF N$="." OR (IN$
V$="0.") AND (N$<"9") THEN 1240 ELS
E RETURN
1240 PC$="5":N$="." AND DP=0 THEN DP=1:GOTO 1
250
1250 IF N$="." AND DP=0 THEN DP=1:GOTO 1
260
1260 IF N$="0" OR N$>"9" THEN 1280
1270 PC$="60":N$="." OR N$>"9":N$=FNEQ$(CP):IF
N$="." OR (N$>"9") AND N$<"9" THEN
N$="1250"
1280 ST=2:GOSUB 1180
1290 CP=CP+1:RETURN
1300 SK1=SK1+1:SK$(SK1)=PC$:RETURN
1310 IF SK1>0 AND SK$(SK1)="20" THEN SK1
=SK1-1:IF SK1=0 OR SK$(SK1)="20" TH
EN RETURN ELSE ON VAL(MID$(SK$(SK1)
,1,1)) GOSUB 1340,1340,1310,1460,14
30:RETURN ELSE JP=1:RETURN
1320 JP=1:RETURN
1330 V=VAL(MID$(PC$,2,LEN(PC$)-1)):GOSUB
1440:RETURN
1340 IF SK2=0 THEN JP=1:RETURN ELSE ON
ASC(MID$(SK$(SK1),LEN(SK$(SK1)),1))
48 GOSUB 1350,1360,1370,1380,1390,
1400,1410:SK1=SK1-1:IF SK1=0 OR SK
$(SK1)="20" THEN RETURN ELSE 1460
1350 SK$(SK2)=SIN(FNRAD(SK$(SK2))):RETURN
1360 SK$(SK2)=COS(FNRAD(SK$(SK2))):RETURN
1370 SK$(SK2)=TAN(FNRAD(SK$(SK2))):RETURN
1380 IF ABS(SK$(SK2))<=1 THEN SK$(SK2)=FND
EG(FNASN(SK$(SK2))):RETURN ELSE JP=1
:RETURN
1390 IF ABS(SK$(SK2))<=1 THEN SK$(SK2)=FND
EG(FNACS(SK$(SK2))):RETURN ELSE JP=1
:RETURN
1400 SK$(SK2)=FNDGEG(ATN(SK$(SK2))):RETURN
1410 IF SK$(SK2)>=0 THEN SK$(SK2)=SQR(SK(S
K2)):RETURN ELSE JP=1:RETURN
1420 IF SK2>1

```

```

1430 IF SK2<0 THEN SK1=SK1-1:RETURN ELS
E JP=1:RETURN
1440 SK2=SK2+1:SK$(SK2)=V:IF SK1>0 AND SK
$(SK1)<>"20" THEN 1450 ELSE RETURN
1450 IF SK1=0 OR MID$(SK$(SK1),1,1)<>"4"
THEN RETURN ELSE GOSUB 1460:RETURN
1460 IF SK2>1 THEN ON VAL(MID$(SK$(SK1)
,LEN(SK$(SK1)),1)) GOSUB 1470,1480,1
490,1500,1510:SK1=SK1-1:SK2=SK2-1:R
ETURN ELSE JP=1:RETURN
1470 SK$(SK2-1)=SK$(SK2-1)+SK$(SK2):RETURN
1480 SK$(SK2-1)=SK$(SK2-1)-SK$(SK2):RETURN
1490 SK$(SK2-1)=SK$(SK2-1)*SK$(SK2):RETURN
1500 IF SK$(SK2)>0 THEN SK$(SK2-1)=SK$(SK2-
1)/SK$(SK2):RETURN ELSE JP=1:RETURN
1510 SK$(SK2-1)=SK$(SK2-1)^SK$(SK2):RETURN
1520 INPUT SUBROUTINE *****
1530 PT=1
1540 LOCATE ROW, COL, 0:PRINT IN$+SPACES(M
AXLEN-LEN(IN$)):IF PT<41 THEN LOCA
TE ROW, COL+(PT-1), 1 ELSE LOCATE ROW
+1, COL+(PT-41), 1
1550 K$="":YP=8*(CSRLIN-1):XP=8*(POS(XP)
-1):PUT (XP,YP),CRS%:WHILE K$="":K$
=INKEY$:WEND:PUT (XP,YP),CRS%
1560 IF K$=CR$ THEN IF LEN(IN$)>PT THEN
PRINT MID$(IN$,PT,1):RETURN ELSE R
ETURN
1570 IF K$=ESC$ THEN IN$="":GOTO 1530
1580 IF INSTR(SELECT$,K$) THEN IN$=LEFT$(
IN$,PT-1)+K$+MID$(IN$,PT+1):PT=PT+
1:IF PT>MAXLEN THEN PT=MAXLEN:GOTO
1540
1590 IF INSTR(SELECT$,K$) THEN IN$=LEF
T$(IN$,PT-1)+CHR$(ASC(K$)-32)+MID$(
IN$,PT+1):PT=PT+1:IF PT>MAXLEN THEN
PT=MAXLEN:GOTO 1540
1600 IF K$=CHR$(8) AND PT>1 THEN IN$=LEF
T$(IN$,PT-2)+MID$(IN$,PT):PT=PT-1:G
OTO 1540
1610 IF K$=CHR$(0)+CHR$(83) THEN IN$=LEF
T$(IN$,PT-1)+MID$(IN$,PT+1):GOTO 15
40
1620 IF K$=CHR$(0)+CHR$(82) AND LEN(IN$)
<MAXLEN THEN IN$=LEFT$(IN$,PT-1)+
"+MID$(IN$,PT):GOTO 1540
1630 IF K$=CHR$(0)+CHR$(77) AND LEN(IN$)
>PT THEN PT=PT+1:IF PT>MAXLEN THEN
PT=MAXLEN:BEep:GOTO 1540
1640 IF K$=CHR$(0)+CHR$(75) AND LEN(IN$)
>1 THEN PT=PT-1:IF PT<1 THEN PT=1:
GOTO 1540
1650 GOTO 1540
1660 CLS:SCREEN 1:COLOR 4,0:KEY OFF:RAND
OMIZE TIMER:FOR I=1 TO 10:KEY I,"":
NEXT
1670 PI=ATN(1)*4
1680 DEF FNDEG(X)=(180*X/PI)
1690 DEF FNRAD(X)=(X*PI/180)
1700 DEF FNEQ$(X)=MID$(IN$,X,1)
1710 DEF FNRD(X)=INT((X+.005)*100)/100
1720 DEF FNASN(X)=ATN(X/SQR(1-X*X))
1730 DEF FNACS(X)=-ATN(X/SQR(1-X*X))+PI/
2
1740 DIM SK$(50):DIM SK(50)
1750 FUNC$="SINCOSTANASNACSATNSQR":OP$="
+*/^":B$="0100001100100110":CR$=CHR
$(13):SP$=CHR$(32):ESC$=CHR$(27)
1760 RIGHTTR$="C3 M+160,-90 M+0,+90 M-16
0,0 BM+150,0 U10 R10"
1770 OBLIQUETR$="C3 M+130,-90 M+30,+90 L
160"
1780 HEIGHT$="XRIGHTTR$:BU80 M-120,90"
1790 PARALLELS$="C3 M+30,-60 R130 M-160,6
0 R130 M+30,-60"
1800 DIM ANG$(10):ANG$="BD6 R6 BL6 E5":D
RAW "BM100,100:XANG$:":GET (100,10
0)-(107,107),ANG%
1810 DIM ENT$(18):ENT$="BD4 E3 D1 G2 D1
E2 D1 G1 D1 F1 U2 E1 R6 U3 L1 D2 L1
U2":DRAW "BM110,110:XENT$:":GET (1
10,110)-(125,117),ENT%
1820 DIM CRS$(10):LINE (140,140)-(147,14
7),3,BF:GET (140,140)-(147,147),CRS
%
1830 RETURN
1840 DATA TRIG-TRIX,4,RIGHT TRIANGLES,LAW
OF SINES,LAW OF COSINES,END PROGR
AM
1850 DATA RIGHT TRIANGLES,3,DETERMINE SI
DES,DETERMINE ANGLES,RETURN TO MAIN
MENU
1860 DATA " ",3,DO A PROBLEM OF THIS TY
PE,ENTER VALUES,EXIT
1870 DATA LAW OF SINES,3,DRILL,CHALLENGE
,RETURN TO MAIN MENU
1880 DATA LAW OF COSINES,3,DRILL,CHALLEN
GE,RETURN TO MAIN MENU
1890 DATA ACBACBCACACBACBACBACBACBACB
ACBACBACBACBACBACBACBACBACBACB
ACBACBACBACBACBACBACBACBACBACB
1900 DATA ABACBACBACBACBACBACBACBACB
BACBACBACBACBACBACBACBACBACB
1910 DATA ABBCACBACBACBACBACBACBACBACB

```

SONG LISTINGS
TYPE-IN LISTINGS


```

100 REM ** TRIG-TRIX **
110 REM ** COPYRIGHT 1985 **
120 REM ** EMERALD VALLEY PUBLISHING CO. **
130 REM ** BY HOMEROGGER WOOD **
140 REM ** VERSION 5.4.1 **
150 REM ** TI BASIC **
160 REM ** OR EXTENDED BASIC **
170 REM ** RANDOMIZE **
180 Z=0
190 E=1
200 W=2
210 DEF FN DEG(X)=(180/X/PIE)
220 DEF FN RAD(X)=(X/PIE/180)
230 DEF FN EQ$(X)=SEG$(IN$(X,E))
240 DEF FN RD(X)=INT((X+.005)*100)/100
250 DEF FN ASN(X)=ATN(X/SQR(E-X*X))
260 DEF FN ACS(X)=-ATN(X/SQR(E-X*X))+PIE/W
270 DIM SK$(50)
280 DIM SK$(50)
290 FN$="SINCOSTANASNACSATNSQR"
300 OP$="+-*/>^"
310 CALL CLEAR
320 CALL SCREEN(4)
330 GOSUB 5250
340 RESTORE 5500
350 GOSUB 5110
360 IF K=52 THEN 440
370 ON K-48 GOSUB 470,560,640
380 GOT OTO 390
390 CALL CLEAR
400 PRINT "SEE YOU NEXT TIME"
410 END
420 RESTORE 5510
430 GOSUB 5110
440 IF K=51 THEN 550
450 GOSUB 5010
460 RESTORE 5550
470 READ PR$
480 ON K-48 GOSUB 840,1080
490 GOT OTO 470
500 RESTORE 5530
510 GOSUB 5110
520 IF K=51 THEN 550
530 GOSUB 5010
540 RESTORE 5570
550 READ PR$
560 ON K-48 GOSUB 1840,2220
570 GOT OTO 560
580 RESTORE 5540
590 GOSUB 5110
600 IF K=51 THEN 550
610 RESTORE 5570
620 READ PR$
630 ON K-48 GOSUB 4220,4510
640 GOT OTO 640
650 RESTORE 5340
660 NL=140
670 SC=4
680 GOSUB 5190
690 RETURN
700 GOSUB 5160
710 RESTORE 5410
720 NL=10
730 SC=W
740 GOSUB 5190
750 RETURN
760 PR=INT(RND*12)
770 GOSUB 5030
780 GOSUB 5050
790 S1=SL
800 TR=Z
810 GOSUB 1060
820 GOSUB 1590
830 GOSUB 720
840 PRINT S1$;"=";S1:A$;"=";A;"DEGREES"
850 IF FIND S1$;"=";S2$ THEN 1700
860 IF K=49 THEN 840
870 IF K=51 THEN 1070
880 GOSUB 720
890 PRINT S1$;"=";
900 INPUT S1
910 PRINT A$;"=";
920 INPUT A
930 GOSUB 1060
940 PRINT S2$;"=";RD(S2)
950 GOSUB 1810
960 GOT OTO 950
970 ON PR+E GOSUB 4880,4880,4900,4900,4
980 920,4920,4940,4960,4960,4980,4
990 RETURN
1000 PR=INT(RND*12)
1010 TR=Z
1020 GOSUB 5050
1030 GOSUB 5070
1040 GOSUB 1590
1050 IF S1$<>"AC" THEN 1170
1060 S1=SH

```

```

1150 S2=SL
1160 GOT OTO 1190
1170 S1=SL
1180 S2=SH
1190 GOSUB 1450
1200 GOSUB 720
1210 PRINT S1$;"=";S1:S2$;"=";S2:"FIND"
1220 A$
1230 T=A
1240 GOSUB 1700
1250 IF K=49 THEN 1080
1260 IF K=51 THEN 1460
1270 GOSUB 720
1280 PRINT S1$;"=";
1290 INPUT S1
1300 PRINT S2$;"=";
1310 INPUT S2
1320 IF PR=7 THEN 1410
1330 IF S1$<>"AC" THEN 1360
1340 IF S1>S2 THEN 1410
1350 PRINT S2$;"MUST BE LESS THAN ";S1$
1360 GOT OTO 1380
1370 IF S2>S1 THEN 1410
1380 PRINT S1$;"MUST BE LESS THAN ";S2$
1390 GOSUB 4180
1400 GOT OTO 1260
1410 GOSUB 5010
1420 GOSUB 1450
1430 PRINT A$;"=";RD(A)
1440 GOSUB 1810
1450 GOT OTO 1240
1460 ON PR+E GOSUB 1470,1470,1490,1490,1
1470 510,1510,1530,1530,1550,1550,1570,1
1480 RETURN
1490 A=DEG(ASN(S2/S1))
1500 RETURN
1510 A=DEG(ASN(S1/S2))
1520 RETURN
1530 A=DEG(ACS(S2/S1))
1540 RETURN
1550 A=DEG(ACS(S1/S2))
1560 RETURN
1570 A=DEG(ATN(S2/S1))
1580 RETURN
1590 A=DEG(ATN(S1/S2))
1600 RETURN
1610 S1$=SEG$(PR$,PR*8+E,W)
1620 A$=SEG$(PR$,PR*8+3,4)
1630 S2$=SEG$(PR$,PR*8+7,W)
1640 RETURN
1650 PRINT "NOT CORRECT, TRY AGAIN."
1660 RETURN
1670 PRINT "NO, ";
1680 RETURN
1690 RESTORE 5520
1700 GOSUB 5110
1710 RETURN
1720 GOSUB 4130
1730 GOSUB 2530
1740 IF RD(VR)=RD(T) THEN 1780
1750 TR=TR+E
1760 IF TR>W THEN 1770
1770 GOSUB 1630
1780 GOT OTO 1700
1790 GOSUB 1650
1800 PRINT RD(T);"IS CORRECT!"
1810 GOSUB 1810
1820 RETURN
1830 GOSUB 4180
1840 GOSUB 1670
1850 RETURN
1860 PR=INT(RND*4)
1870 TR=Z
1880 GOSUB 5030
1890 GOSUB 5050
1900 GOSUB 5070
1910 GOSUB 2150
1920 GOSUB 2100
1930 PRINT S1$;"=";SH:S2$;"=";SL:A$;"=";
1940 A:"DEGREES";"FIND";A2$
1950 T=A2
1960 GOSUB 1700
1970 IF K=49 THEN 1840
1980 IF K=51 THEN 2140
1990 IV=Z
2000 GOSUB 780
2010 PRINT S1$;"=";
2020 INPUT SH
2030 PRINT S2$;"=";
2040 INPUT SL
2050 PRINT A$;"=";
2060 INPUT A
2070 GOSUB 2150
2080 IF IV THEN 1970
2090 PRINT A2$;"=";RD(A2)
2100 GOSUB 1810
2110 GOT OTO 1950
2120 S1$=SEG$(PR$,PR*12+E,W)
2130 A$=SEG$(PR$,PR*12+3,4)
2140 S2$=SEG$(PR$,PR*12+7,W)
2150 A2$=SEG$(PR$,PR*12+9,4)
2160 RETURN
2170 IF ABS(SL*SIN(RAD(A)))/SH<=E THEN 2
2180 PRINT "ILLEGAL VALUES--TRY AGAIN"
2190 IV=E

```

Continued


```

21180 GOSUB 4180
21190 RETURN
22100 A2=DEG (ASN (SL * SIN (RAD (A)) / SH))
22200 RETURN
22210 TR=Z
22220 GOSUB 5090
22230 A2=A-INT (RND * 10) - E
22240 GOSUB 5090
22250 GOSUB 2510
22260 GOSUB 2450
22270 PRINT "AB="; SL: "qBAD="; A2: "qCBD="; A
22280 : "CD="
22290 : "FIND
22300 GOSUB 1700
22310 IF SK1=51 THEN 2220
22320 IF SK1=51 THEN 2520
22330 GOSUB 2450
22340 INPUT "AB="; SL: A2
22350 INPUT "qBAD="; A
22360 INPUT "qCBD="; A
22370 IF (A < 90) * (A2 < A) THEN 2410
22380 PRINT "ILLEGAL ANGLE VALUES."
22390 GOSUB 4180
22400 GOSUB 2330
22410 GOSUB 2330
22420 PRINT "CD="; RD (SH)
22430 GOSUB 1810
22440 GOSUB 2310
22450 GOSUB 5160
22460 NL=13
22470 SC=5
22480 RESTORE 5370
22490 GOSUB 5190
22500 RETURN
22510 SH=SL * SIN (RAD (A2)) * SIN (RAD (A)) / SIN (RAD (A-A2))
22520 RETURN
22530 ST=E
22540 GOSUB 2840
22550 CP=E
22560 DP=E
22570 SK1=N
22580 SK2=N
22590 JP=N
22600 IF CP=V THEN 2810
22610 IF CP=V THEN 2640
22620 CP=CP+E
22630 GOTO 2600
22640 PC=0
22650 FOR I=1 TO 5
22660 BH=VAL (SEG$ (BH$, I, E))
22670 IF BH=N THEN 2600
22680 ON I GOSUB 2860, 2930, 2980, 3030, 3090
22690 IF PC=0 THEN 2720
22700 NEXT I
22710 GOTO 2770
22720 JP=N
22730 ON VAL (SEG$ (PC$, E, E)) GOSUB 3260, 326
0, 3320, 3260, 3390
22740 IF JP THEN 2770
22750 CP=CP+E
22760 GOTO 2600
22770 IF JP THEN 2790
22780 JP=E
22790 GOSUB 4040
22800 GOTO 2830
22810 IF SK1 > Z THEN 2770
22820 VR=SK (E)
22830 RETURN
22840 BH$=SEG$ (BH$, ST * 5 + E, 5)
22850 RETURN
22860 P=POS (FN$, SEG$ (IN$, CP, 3), E)
22870 IF P=N THEN 2920
22880 PC$="1" & CHR$ (49 + (INT (P / 3)))
22890 CP=CP+P
22900 ST=N
22910 GOSUB 2840
22920 RETURN
22930 IF EQ$ (CP) <> "(" THEN 2970
22940 PC$="20"
22950 ST=E
22960 GOSUB 2840
22970 RETURN
22980 IF EQ$ (CP) <> ")" THEN 3020
22990 PC$="30"
23000 ST=W
23010 GOSUB 2840
23020 RETURN
23030 P=POS (OP$, EQ$ (CP), E)
23040 IF P=N THEN 3080
23050 PC$="4" & STR$ (P)
23060 ST=E
23070 GOSUB 2840
23080 RETURN
23090 DP=N
23100 NS=EQ$ (CP)
23110 IF (NS$="." ) + ((NS$="0") * (NS$<="9")) THEN 3130
23120 EN 3130
23130 RETURN
23140 PC$="5"
23150 DP=E
23160 GOTO 3180
23170 IF (NS$<="0") + ((NS$>"9")) THEN 3220
23180 PC$=PC$&NS
23190 CP=CP+E
23200 NS=EQ$ (CP)

```

```

3210 IF (NS$="." ) + ((NS$="0") * (NS$<="9")) THEN 3140
3220 ST=W
3230 GOSUB 2840
3240 CP=CP+E
3250 RETURN
3260 SK1=SK1+E
3270 SK$ (SK1)=PC$
3280 RETURN
3290 SK2=SK2+E
3300 SK$ (SK2)=V
3310 RETURN
3320 IF (SK1 > Z) * (SK$ (SK1) = "20") THEN 3350
3330 JP=E
3340 RETURN
3350 SK1=SK1-E
3360 IF (SK1=Z) + (SK$ (SK1) = "20") THEN 3380
3370 ON VAL (SEG$ (SK$ (SK1), E, E)) GOSUB 3420, 3420, 3320, 3730, 3930
3380 RETURN
3390 V=VAL (SEG$ (PC$, W, LEN (PC$) - E))
3400 GOSUB 3980
3410 RETURN
3420 IF SK2 > Z THEN 3450
3430 JP=E
3440 RETURN
3450 ON ASC (SEG$ (SK$ (SK1), LEN (SK$ (SK1)), E)) - 48 GOSUB 3500, 3520, 3540, 3560, 3610, 3660, 3680
3460 SK1=SK1-E
3470 IF (SK1=Z) + (SK$ (SK1) = "20") THEN 3490
3480 GOSUB 3730
3490 RETURN
3500 SK (SK2)=SIN (RAD (SK (SK2)))
3510 RETURN
3520 SK (SK2)=COS (RAD (SK (SK2)))
3530 RETURN
3540 SK (SK2)=TAN (RAD (SK (SK2)))
3550 RETURN
3560 IF ABS (SK (SK2)) <= E THEN 3590
3570 JP=E
3580 RETURN
3590 SK (SK2)=DEG (ASN (SK (SK2)))
3600 RETURN
3610 IF ABS (SK (SK2)) <= E THEN 3640
3620 JP=E
3630 RETURN
3640 SK (SK2)=DEG (ACS (SK (SK2)))
3650 RETURN
3660 SK (SK2)=DEG (ATN (SK (SK2)))
3670 RETURN
3680 IF SK (SK2) >= Z THEN 3710
3690 JP=E
3700 RETURN
3710 SK (SK2)=SQR (SK (SK2))
3720 RETURN
3730 IF SK2 > E THEN 3760
3740 JP=E
3750 RETURN
3760 ON VAL (SEG$ (SK$ (SK1), LEN (SK$ (SK1)), E)) GOSUB 3800, 3820, 3840, 3860, 3890
3770 SK1=SK1-E
3780 SK2=SK2-E
3790 RETURN
3800 SK (SK2-E)=SK (SK2-E) + SK (SK2)
3810 RETURN
3820 SK (SK2-E)=SK (SK2-E) - SK (SK2)
3830 RETURN
3840 SK (SK2-E)=SK (SK2-E) * SK (SK2)
3850 RETURN
3860 IF SK (SK2) > Z THEN 3910
3870 JP=E
3880 RETURN
3890 SK (SK2-E)=SK (SK2-E) ^ SK (SK2)
3900 RETURN
3910 SK (SK2-E)=SK (SK2-E) / SK (SK2)
3920 RETURN
3930 IF SK2 <> Z THEN 3960
3940 JP=E
3950 RETURN
3960 SK1=SK1-E
3970 RETURN
3980 GOSUB 3290
3990 IF (SK1 > Z) * (SK$ (SK1) <> "20") THEN 4010
4000 RETURN
4010 IF (SK1=Z) + (SEG$ (SK$ (SK1), E, E) <> "4") THEN 4030
4020 GOSUB 3730
4030 RETURN
4040 PRINT "SYNTAX ERROR"
4050 PRINT "SYNTAX ERROR"
4060 RETURN
4070 CALL KEY (Z, K, S)
4080 IF S <> E THEN 4070
4090 RETURN
4100 RESTORE 5520
4110 GOSUB 5110
4120 RETURN
4130 INPUT IN$
4140 IF LEN (IN$) < 51 THEN 4170
4150 PRINT "LIMIT INPUT TO 50 CHARACTERS"
4160 GOTO 4130
4170 GOTO 5010
4180 PRINT "PRESS [ENTER] TO CONTINUE"
4190 GOSUB 4070
4200 IF K <> 13 THEN 4190

```

TYPE-IN LISTINGS

TYPE-IN LISTINGS

```

4210 RETURN
4220 PR=INT (RND*3)
4230 GOSUB 4460
4240 TR=Z
4250 GOSUB 5030
4260 GOSUB 5050
4270 GOSUB 5070
4280 GOSUB 4770
4290 GOSUB 780
4300 PRINT S2$;"=";SH:S3$;"=";SL:A$;"=";
4310 A:"FIND";S1$
4320 T=S1
4330 GOSUB 1700
4340 IF K=49 THEN 4220
4350 IF K=51 THEN 4500
4360 GOSUB 780
4370 PRINT S2$;"=";
4380 INPUT SH
4390 PRINT S3$;"=";
4400 INPUT SL
4410 PRINT A$;"=";
4420 INPUT A
4430 GOSUB 4770
4440 PRINT S1$;"=";RD(S1)
4450 GOSUB 1810
4460 GOTO 4330
4470 S2$=SEG$(PR$,PR*10+E,W)
4480 S3$=SEG$(PR$,PR*10+3,W)
4490 S1$=SEG$(PR$,PR*10+5,W)
4500 A$=SEG$(PR$,PR*10+7,4)
4510 RETURN
4520 TR=Z
4530 GOSUB 5030
4540 A=180-A
4550 GOSUB 5050
4560 GOSUB 5070
4570 GOSUB 4770
4580 PRINT "AB=";SH;" AB=DC":AD=";SL;"
AD=BC":qABC=";A:"FIND AC"
4590 T=S1
4600 GOSUB 1700
4610 IF K=49 THEN 4510
4620 IF K=51 THEN 4780
4630 GOSUB 4770
4640 INPUT "AB=";SH
4650 INPUT "AD=";SL
4660 INPUT "qABC=";A
4670 GOSUB 4770
4680 PRINT "AC=";RD(S1)
4690 GOSUB 1810
4700 GOTO 4610
4710 GOSUB 5160
4720 NL=14
4730 SC=E
4740 RESTORE 5450
4750 GOSUB 5190
4760 RETURN
4770 S1=SQR((SH^W)+(SL^W)-(W*SH*SL*COS(RAD(A))))
4780 RETURN
4790 READ T$
4800 READ OP
4810 GOSUB 5160
4820 FOR I=E TO OP
4830 READ CH$
4840 PRINT :STR$(I);". ";CH$
4850 NEXT I
4860 PRINT :::"YOUR CHOICE:":::
4870 RETURN
4880 S2=S1*SIN(RAD(A))
4890 RETURN
4900 S2=S1/SIN(RAD(A))
4910 RETURN
4920 S2=S1*COS(RAD(A))
4930 RETURN
4940 S2=S1/COS(RAD(A))
4950 RETURN
4960 S2=S1*TAN(RAD(A))
4970 RETURN
4980 S2=S1/TAN(RAD(A))
4990 RETURN
5000 A=DEC(ASN(S1/S2))
5010 PRINT "CALCULATING:"
5020 RETURN
5030 A=INT(RND*89)+E
5040 RETURN
5050 SL=INT(RND*20)+E
5060 RETURN
5070 SH=INT(RND*20)+SL+E
5080 RETURN

```

[illegible]

ARCHEODROID

APPLE // Family

```

100 REM
110 REM
120 REM
130 REM
140 REM
150 REM
160 REM
170 REM
180 REM
190 REM
200 IF PEEK (104) = 64 THEN 230

```

```

210 POKE 16384,0
220 PRINT CHR$(4) : RUN = ARCHEOCHAR = 2
230 TEXT : HOME = 2051 : HCHAR FOR FLASH : HTAB SPKR 9 : VTAB K NORMA EODROID : VTAB 8 " NEXT ARCH F.P. POKK NEXT 4096 TO 4188 : READ P : POKE

```

Continued


```

260 READ X$ CHR$(7) IF X$ < "DATA" THEN P
270 FOR K=1 TO 2048: NEXT K: PRINT CHR$(7); < "DATA" E
280 SCREEN=2: C=21: POKE=1: TTL=0: CG=10
290 VBOARD=14: HTAB=3: OYSTICK=1: SELECT (K) E
300 SOUND=16: EFFECTS=1: PRINT "DO YOU WANT 3
310 POKE=48: SET UP=1: SCALE=1: ROT=0:
320 HCOLOR=232: POKE=5: 233: FOR K=4 TO 1
330 CALL HCHAR,2,1,1,5,5,0: CALL HCHAR,K
340 CALL HCHAR,2,1,1,5,5,0: CALL HCHAR,K
350 CALL HCHAR,2,1,1,5,5,0: CALL HCHAR,3,1
360 SNUG=0: FOR STEP 2: K=5 TO 18: FOR XX
370 IF YY,XX,K=2+7,0: SNUG=CALL HCHAR,YY,XX
380 NEXT K,XX,YY: SNUG=SNUG+2
390 HCOLOR=5: DRAW 1 AT 230,0: FOR K=KF
400 HCOLOR=3: HPLOT 222,14 TO 222,16: 1
410 HPLOT 227,14 TO 227,16: HPLOT 233,1
420 VBOARD=21: HTAB=12: PRINT "<< ARCHEOD
430 REM *** DISPLAY MEN ***
440 CALL HCHAR,2,10,0,0: CALL HCHAR,2,1
450 FOR K=1 TO 0 STEP 1: IF K=1
460 CALL HCHAR,1,10,K,0: CALL HCHAR,2,1
470 CALL HCHAR,1,12,K,0: CALL HCHAR,2,1
480 FOR X=1 TO 4: CALL HCHAR,X,21,K,0
490 CALL NOISE,3: CALL NOISE,3: NEXT K
500 CALL HCHAR,2,21,2,0: IF MINER > 1 T
510 CALL HCHAR,2,12,2,0: IF MINER
520 VBOARD=22: HTAB=1: PRINT "SCORE=TT
530 SS="MNUG: TAB(24): "CHARGES="CG
540 REM *** MAIN GAME LOOP ***
550 TIME=120: THEN 1130
560 IF KJ THEN 590
570 A=PEEK(16384)-128: IF A >
580 THEN POKE 16384,0
590 IF A > 32 THEN 900
600 IF A > 72 THEN 650
610 GOTO 730
620 IF P=0 THEN 16287 > 127 THEN 900
630 IF P=0 THEN 216 THEN 690
640 IF P=0 THEN 216 THEN 650
650 IF P=0 THEN 216 THEN 730
660 GOTO 530
670 IF 1: CALL GCHAR,R-1,C: CH=PE
680 CALL HCHAR,R,C,0,0: R=1: GOTO
690 IF 2: CALL GCHAR,R,C-1: CH=PE
700 IF 3: CALL NOISE,248 > 3 AND CH <
710 IF 4: CALL NOISE,248 > 3 AND CH <
720 CALL HCHAR,R,C,0,0: C=C-1: GOTO
730 IF 18 THEN 530
740 IF 3: CALL GCHAR,R+1,C: CH=PE
750 IF 3: CALL NOISE,248 > 3 AND CH <
760 CALL HCHAR,R,C,0,0: R=R+1: GOTO
770 IF C > 38 THEN 530
    
```

```

780 POKE=4: CALL GCHAR,R,C+1: CH=PE
790 IF 6: CALL NOISE,248 > 3 AND CH <
800 IF 2 AND C=31 THEN 1200
810 CALL HCHAR,R,C,0: C=C+1: GOTO
820 REM *** PICK UP AN OBJECT ***
830 IF GO TO 530 THEN CALL HCHAR,R,C,2,0:
840 IF GO TO 530 THEN MNUG=MNUG+DNUG:D
850 IF 127: CALL HCHAR,R,C,2,0: GOTO 510
860 SNUG=SNUG-1: IF SNUG=0 OR CH=
870 IF 3 OR CH=9 THEN MNUG=MNU
880 IF CH=10 OR CH=11 THEN MNUG=M
890 CALL NOISE,8: CALL HCHAR,R,C,2,0: G
900 REM *** SET AN EXPLOSIVE CHARGE *
910 IF R < 3 OR C < 3 OR C > 38 OR R >
920 R1=(PO=2)+(PO=3): C1=C
930 CALL GCHAR,R1,C1-1: GOSUB 1030
940 CALL GCHAR,R1+1,C1-1: GOSUB 1030
950 CALL GCHAR,R1-1,C1+1: GOSUB 1030
960 CALL GCHAR,R1+1,C1+1: GOSUB 1030
970 FOR K=1 TO 0 STEP 1: CALL HCHA
980 CALL HCHAR,R1,K,0
990 CALL HCHAR,R1+1,K,0
1000 CALL HCHAR,R1+1,K,0
1010 CALL HCHAR,R1+1,K,0
1020 CALL NOISE,K*2+127: NEXT: GOTO
1030 CH=PEEK(0): IF CH=5 THEN POP
1040 IF CH > 5 THEN SNUG=SNUG-1
1050 RETURN
1060 REM *** CAVE-IN ROUTINE ***
1070 CG=1: 1 TO 10: R1=INT(RND(1)
1080 FOR I=14: C1=INT(RND(1)*35
1090 IF CH > 5 THEN 1170
1100 IF CH=0 THEN 1130
1110 IF CH=0 THEN CALL HCHAR,R1,C1,4,
1120 GOTO 1170
1130 MNUG=MNUG: CALL HCHAR,R,C,3,0: CALL
1140 NOISE,255: CALL NOISE,255: MINER=M
1150 FLAG=0: R=2: C=21: CG=10: IF M
1160 GOTO 1260
1170 NEXT I: IF CG=0 THEN FLAG=1: TIM
1180 E=0: CALL NOISE,63
1190 REM *** BACK TO THE SHIP ***
1200 IF CG < 0 THEN CG=.5
1210 IF FLAG=0: TTL=10: (MNUG*CG)+T
1220 TL: IF MNUG > 0 THEN CALL NOISE
1230 IF CG < 1 THEN CG=10
1240 MNUG=0: IF SNUG > 0 THEN 510
1250 SCREEN=SCREEN+1: GOTO 330
1260 REM *** END OF GAME ***
1270 TEXT: HOME: VBOARD=10: PRINT "THE PRI
1280 ARTIFACTS YOU HAVE ARE WORTH": THE PRI
1290 NT: PRINT TTL: "CREDITS."
1300 VBOARD=18: PRINT "DO YOU WANT TO PLA
1310 Y AGAIN? (Y/N) "Y" AND AS < > "N"
1320 IF AS="Y" THEN HOME: GOTO 280
1330 HOME: END
1340 REM *** SPACE SHIP TABLE ***
1350 DATA 1,0,4,0,7,1,45,17,63,63,63,19
1360 45,45,45,45,45,45,45,45,45,45,45,2
1370 45,45,45,45,45,45,45,45,45,45,45,45
1380 45,45,45,45,45,45,45,45,45,45,45,45
1390 45,45,45,45,45,45,45,45,45,45,45,45
1400 45,45,45,45,45,45,45,45,45,45,45,45
1410 45,45,45,45,45,45,45,45,45,45,45,45
1420 45,45,45,45,45,45,45,45,45,45,45,45
1430 45,45,45,45,45,45,45,45,45,45,45,45
1440 45,45,45,45,45,45,45,45,45,45,45,45
1450 45,45,45,45,45,45,45,45,45,45,45,45
1460 45,45,45,45,45,45,45,45,45,45,45,45
1470 45,45,45,45,45,45,45,45,45,45,45,45
1480 45,45,45,45,45,45,45,45,45,45,45,45
1490 45,45,45,45,45,45,45,45,45,45,45,45
1500 45,45,45,45,45,45,45,45,45,45,45,45
1510 45,45,45,45,45,45,45,45,45,45,45,45
1520 45,45,45,45,45,45,45,45,45,45,45,45
1530 45,45,45,45,45,45,45,45,45,45,45,45
1540 45,45,45,45,45,45,45,45,45,45,45,45
1550 45,45,45,45,45,45,45,45,45,45,45,45
1560 45,45,45,45,45,45,45,45,45,45,45,45
1570 45,45,45,45,45,45,45,45,45,45,45,45
1580 45,45,45,45,45,45,45,45,45,45,45,45
1590 45,45,45,45,45,45,45,45,45,45,45,45
1600 45,45,45,45,45,45,45,45,45,45,45,45
1610 45,45,45,45,45,45,45,45,45,45,45,45
1620 45,45,45,45,45,45,45,45,45,45,45,45
1630 45,45,45,45,45,45,45,45,45,45,45,45
1640 45,45,45,45,45,45,45,45,45,45,45,45
1650 45,45,45,45,45,45,45,45,45,45,45,45
1660 45,45,45,45,45,45,45,45,45,45,45,45
1670 45,45,45,45,45,45,45,45,45,45,45,45
1680 45,45,45,45,45,45,45,45,45,45,45,45
1690 45,45,45,45,45,45,45,45,45,45,45,45
1700 45,45,45,45,45,45,45,45,45,45,45,45
1710 45,45,45,45,45,45,45,45,45,45,45,45
1720 45,45,45,45,45,45,45,45,45,45,45,45
1730 45,45,45,45,45,45,45,45,45,45,45,45
1740 45,45,45,45,45,45,45,45,45,45,45,45
1750 45,45,45,45,45,45,45,45,45,45,45,45
1760 45,45,45,45,45,45,45,45,45,45,45,45
1770 45,45,45,45,45,45,45,45,45,45,45,45
    
```

Continued

[illegible]

COMMODORE 64

```

650 POKER2+CM,26:POKER2+CM+1,26:POKER2+
660 CM+CL,8:POKER2+CM+1+CL,8
670 NEXTCM:NEXTR2
680 POKE1089,27:POKE1089+CL,7:POKE1090,
28:POKE1090+CL,7:POKE1091,29:
690 POKE1129,59:POKE1129+CL,7:POKE1130,
60:POKE1130+CL,7:POKE1131,61
700 POKE1091+CL,7:POKE1131+CL,7
710 ONMINERGOTO730,720,710
720 W=1029:FORX=WTO(W+80)STEP40:POKEX,1
02:POKEX+CL,7:NEXTX
730 W=1033:FORX=WTO(W+80)STEP40:POKEX,1
02:POKEX+CL,7:NEXTX
740 W=1044:FORX=WTO(W+160)STEP40:POKEX,
102:POKEX+CL,7:NEXTX
750 S9=54272:FOR9=S9TOS9+24:POKET9,0:N
EXT:POKES54296,15
760 POKE54284,17:POKE54285,27:POKE54286
,10:POKE54279,255:POKE54283,129
770 POKE53281,1:POKE53280,1:POKE53280,0
:POKE53281,0
780 ON MINER GOTO800,790,780
790 W=1029:FORX=WTO(W+80)STEP40:POKEX,3
2:POKEX+CL,0:NEXTX
800 W=1033:FORX=WTO(W+80)STEP40:POKEX,3
2:POKEX+CL,0:NEXTX
810 W=1044:FORX=WTO(W+160)STEP40:POKEX,
32:POKEX+CL,0:NEXTX
820 POKE54283,128:POKE198,0
830 ON MINER GOTO850,840,830
840 POKE1109,17:POKE1109+CL,7
850 POKE1113,17:POKE1113+CL,7
860 POKE1124,17:POKE1124+CL,7
870 C=20
880 GOTO1710
890 MNUG=MNUG+DNUG:DNUG=0
900 PRINT"PHOME"23CRSRDOWN";TAB(16);M
NUG;"PHOME":GOTO2340
910 IF BLAST<1THEN1710
920 IFR<3ORR>20THEN1710
930 IFC<3ORR>37THEN1710
940 ONPOGOSUB1160,1170,1180,1190
950 IFC1<3ORC1>37THEN1710
960 IFR1<3ORR1>20THEN1710
970 C5=PEEK(1024+C1+40*(R1-1))
980 C6=PEEK(1024+(C1-1)+40*(R1-1))
990 C7=PEEK(1024+(C1-1)+40*(R1+1))
1000 C8=PEEK(1024+(C1+1)+40*(R1-1))
1010 C9=PEEK(1024+(C1+1)+40*(R1+1))
1020 IFC5=24ORC6=24ORC7=24ORC8=24ORC9=24
THEN1710
1030 GOSUB1200
1040 M1=1:M2=1:GOSUB2660
1050 POKE(1024+C1+40*(R1))+CL,M1
1060 POKE(1024+(C1-1)+40*(R1-1)+CL),M1
1070 POKE(1024+(C1-1)+40*(R1+1)+CL),M1
1080 POKE(1024+(C1+1)+40*(R1-1)+CL),M1
1090 POKE(1024+(C1+1)+40*(R1+1)+CL),M1
1100 POKE(1024+(C1-1)+40*(R1-1)),32
1110 POKE(1024+(C1-1)+40*(R1+1)),32
1120 POKE(1024+(C1+1)+40*(R1-1)),32
1130 POKE(1024+(C1+1)+40*(R1+1)),32
1140 IFM2=1THENM1=0:M2=0:GOTO1040
1150 GOTO1310
1160 R1=R-1:C1=C:RETURN
1170 C1=C-1:R1=R:RETURN
1180 R1=R+1:C1=C:RETURN
1190 C1=C+1:R1=R:RETURN
1200 IFC5<38ORC5>43THEN1220
1210 SNUG=SNUG-1
1220 IFC6<38ORC6>43THEN1240
1230 SNUG=SNUG-1
1240 IFC7<38ORC7>43THEN1260
1250 SNUG=SNUG-1
1260 IFC8<38ORC8>43THEN1280
1270 SNUG=SNUG-1
1280 IFC9<38ORC9>43THEN1300
1290 SNUG=SNUG-1
1300 RETURN
1310 BLAST=BLAST-1
1320 PRINT"PHOME"23CRSRDOWN";TAB(35);"
PHOME"
1330 PRINT"PHOME"23CRSRDOWN";TAB(35);B
LAST;"PHOME"
1340 FORI=1TO10:RR=INT(RND(1)*16)+4:RC=I
NT(RND(1)*35)+2
1350 RS=PEEK(1024+RC+40*RR)
1360 IFRS=17THEN1620
1370 IFRS=32THEN1390
1380 GOTO1400

```

Continued


```

1390 POKE(1024+RC+40*RR),26:POKE(1024+RC
+40*RR+CL),8:GOTO1400
1400 NEXT I
1410 IFBLAST<1 THEN1600
1420 GOTO1710
1430 IFBLAST>0 THEN1450
1440 BLAST=.5
1450 TTL=10*(MNUG*BLAST)+TTL
1460 IFBLAST>.5 THEN1490
1470 BLAST=10:FL=0:TM=0
1480 PRINT:"HOME"23CRSRDOWN";TAB(35);B
LAST:"HOME"
1490 MNUG=0:PRINT:"HOME"CRSRDOWN";TAB(
35):TTL
1500 PRINT:"HOME"23CRSRDOWN";TAB(16);"
NUG:"HOME"
1510 PRINT:"HOME"23CRSRDOWN";TAB(16);M
NUG:"HOME"
1520 IFSNUG>0 THEN1710
1530 SCREEN=SCREEN+1:PRINT:"SHIFT CLR":
GOTO450
1540 PRINT:"SHIFT CLR"2CRSRDOWN"THE ART
IFACTS YOU HAVE":PRINT:"CRSRDOWN"AR
E WORTH:TTL:CREDITS
1550 PRINT:"CRSRDOWN"DO YOU WANT TO PLA
Y AGAIN?
OR CTRL RVSON CTRL RVSON
1560 GETK2$:IFK2$="Y" THEN1560
1570 IFK2$="N" THEN430
1580 IFK2$="Y" THEN1560
1590 POKE53272,21:END
1600 FL=1:TM=TM+1:IFTM>40 THENFL=0:TM=0:G
OTO1620
1610 GOTO1710
1620 REM MINER DIES
1630 MINER=MINER-1
1640 DNUG=MNUG:MNUG=0:BLAST=10
1650 POKE(1024+C+40*R),37:POKE(1024+C+40
*R+CL),14:GOSUB2620
1660 IFMINER<1 GOTO1540
1670 PRINT:"HOME"23CRSRDOWN";TAB(16);"
NUG:"HOME"
1680 PRINT:"HOME"23CRSRDOWN";TAB(16);M
NUG:"HOME"
1690 PRINT:"HOME"23CRSRDOWN";TAB(35);B
LAST:"HOME"
1700 R=2:C=20:SP=1124:PRINT:"HOME"2CRSR
DOWN":POKE198,0:GO
TO700
1710 REM GET SUBROUTINE
1720 SP=1024+C+40*R
1730 IFK$="J" THEN1850
1740 GETJVS:IFJVS=" " THEN1760
1750 GOTO1790
1760 IFFL=1 THENQK=QK+1
1770 IFQK>40 THENQK=0:GOSUB2690:GOTO1600
1780 GOTO1740
1790 JV=ASC(JVS):IFJVS=89 THEN1920
1800 IFJVS=72 THEN2130
1810 IFJVS=66 THEN2060
1820 IFJVS=71 THEN1990
1830 IFJVS=75 THEN900
1840 GOTO1740
1850 JV=PEEK(56320)AND15:JV=JV-6:FR=((P
EEK(56320)AND16)=0)
1860 IFFR=1 THEN900
1870 IFJVS=1 ORJV>8 GOTO1890
1880 ONJVGOTO2130,1890,1890,1890,1990,18
90,2060,1920
1890 IFFL=1 THENQK=QK+1
1900 IFQK>40 THENQK=0:GOSUB2690:GOTO1600
1910 GOTO1710
1920 IFR=1:2 THEN1710
1930 PO=1:CH=PEEK(1024+C+40*(R-1))
1940 IFCH=24 ORCH=26 THEN1710
1950 IFCH=59 THEN1430
1960 IFFL=0 THENGOSUB2690
1970 POKE53272,32:POKE53272,0
1980 R=R-1:GOTO2200
1990 IF(C-1)<2 OR(R=2 ANDC<19) THEN1710
2000 PO=2:CH=PEEK(1024+(C-1)+40*R)
2010 IFCH=24 ORCH=26 THEN1710
2020 IFCH=59 THEN1430
2030 IFFL=0 THENGOSUB2690
2040 POKE53272,32:POKE53272,0
2050 C=C-1:GOTO2200
2060 IFR=1:2 THEN1710
2070 PO=3:CH=PEEK(1024+C+40*(R+1))
2080 IFCH=24 ORCH=26 THEN1710
2090 IFCH=59 THEN1430
2100 IFFL=0 THENGOSUB2690

```

```

2110 POKE53272,32:POKE53272,0
2120 R=R+1:GOTO2200
2130 IFC=1:38 THEN1710
2140 PO=4:CH=PEEK(1024+(C+1)+40*R)
2150 IFCH=24 ORCH=26 THEN1710
2160 IFCH=59 THEN1430
2170 IFFL=0 THENGOSUB2690
2180 POKE53272,32:POKE53272,0
2190 C=C+1
2200 IFCH=32 THEN2340
2210 IFCH=37 THEN880
2220 SNUG=SNUG-1
2230 IFCH=40 ORCH=41 THEN2250
2240 GOTO2260
2250 MNUG=MNUG+3:GOTO2320
2260 IFCH=38 ORCH=39 THEN2280
2270 GOTO2290
2280 MNUG=MNUG+5:GOTO2320
2290 IFCH=42 ORCH=43 THEN2310
2300 GOTO2330
2310 MNUG=MNUG+1
2320 FM=1
2330 PRINT:"HOME"23CRSRDOWN";TAB(16);M
NUG:"HOME"
2340 SP=1024+C+40*R
2350 POKE53272,17:POKE53272,7
2360 IFFM=1 THENGOSUB2720:FM=0
2370 IFFL=1 THEN1600
2380 GOTO1710
2390 DATA017,060,060,255,195,255,255,102
,231
2400 DATA024,255,221,255,187,255,221,255
,255
2410 DATA026,255,221,255,187,255,221,255
,255
2420 DATA027,000,063,063,127,127,255,128
,064
2430 DATA028,000,255,255,255,129,129,126
,126
2440 DATA029,000,252,252,254,254,255,001
,002
2450 DATA037,024,024,126,024,024,024,024
,024
2460 DATA038,000,000,112,221,247,221,112
,000
2470 DATA040,000,015,017,049,127,255,255
,048
2480 DATA041,000,192,032,016,255,255,255
,012
2490 DATA042,000,000,057,063,127,255,063
,030
2500 DATA043,000,096,086,246,255,255,223
,252
2510 DATA039,000,097,195,134,252,134,195
,097
2520 DATA059,064,063,018,018,018,018,018
,055
2530 DATA060,000,255,034,034,034,034,034
,119
2540 DATA061,002,252,036,036,036,036,036
,118
2550 DATA-1
2560 DIMDM(33):X=1:POKE54296,15
2570 READA
2580 IFA=-1 THENRETURN
2590 DM(X)=A:X=X+1:GOTO2570
2600 DATA22,96,192,22,96,192,22,96,96,22
,96,256,26,156,192,25,30,128
2610 DATA25,30,128,22,96,128,22,96,128,
21,31,128,22,96,128,-1
2620 FORLL=54272 TO54295:POKELL,0:NEXT
2630 POKE54284,64:POKE54285,248
2640 FORA=1 TO33STEP3:POKE54280,DM(A):POK
E54279,DM(A+1):POKE54283,17
2650 FORT=1 TO25:NEXTT:NEXTA:RETURN
2660 FORLL=54272 TO54295:POKELL,0:NEXT
2670 POKE54284,31:POKE54285,17:POKE54283
,129:POKE54280,100
2680 POKE54279,143:FORW=1 TO200:NEXT:POKE
54283,0:POKE54284,0:RETURN
2690 FORLL=54272 TO54295:POKELL,0:NEXT
2700 POKE54284,34:POKE54285,34:POKE54283
,33:POKE54280,11
2710 POKE54279,43:FORW=1 TO100:NEXT:POKE5
4283,0:POKE54284,0:RETURN
2720 FORLL=54272 TO54295:POKELL,0:NEXT
2730 POKE54284,34:POKE54285,34:POKE54283
,33:POKE54280,79
2740 POKE54279,5:FORW=1 TO175:NEXT:POKE54
283,0:POKE54384,0:RETURN

```

ARCHEODROID

IBM PC & IBM PCjr

```

1100 *****
1110 ***** ARCHEODROID *****
1120 *****
1130 *****
1140 ***** COPYRIGHT 1983, 1984, 1985
1150 ***** EMERALD VALLEY PUBLISHING CO. *****
1160 ***** BY B. J. BRUNS *****
1170 ***** AND THE HCM STAFF *****
1180 ***** HOME COMPUTER MAGAZINE *****
1190 ***** VERSION 5.4.1 *****
1200 ***** DOS 2.1 AND EITHER *****

```

```

2100 IBM PCjr WITH CARTRIDGE BASIC OR
2110 IBM PC WITH BASICA and
2120 COLOR/GRAPHICS ADAPTER and
2130 COLOR MONITOR
2140
2150 CLS:KEY OFF:SCREEN 1:COLOR 1,0:LOCA
TE 12,9:PRINT "A R C H E O D R O I
D":LOCATE 22,7:PRINT "PRESS [ENTER]
TO CONTINUE":GOSUB 1400

```

Continued


```

260 RANDOMIZE TIMER:DEFINT A-Z:DIM ART(
39,23),ROB(12),SKEL(20),CAR(20),GAR
B(20),BLNK(20),FILL(20),RA(2),BIGFI
LL(40)
270 STRIG ON:CLS:LOCATE 12,1:PRINT "PRE
SS:::PRINT:PRINT "K" FOR KEYBOARD::
PRINT:PRINT "J" FOR JOYSTICK PORT 1
280 GOSUB 1400:IF AS="K" OR AS="L" THEN
KJ=1 ELSE IF AS"<"J" AND AS">"J" T
HEN 280 ELSE KJ=2:AM=STICK(0):BM=ST
ICK(1)
290 PCHARGE=10:NROB=3:GOSUB 370:GOSUB 5
70
300 ' MAIN CONTROL LOOP
310 DEF SEG=0:POKE 1050,PEEK(1052):GOSU
B 690:IF NROB>0 THEN PCHARGE=10:GOS
UB 570:GOTO 330 ELSE 1440
320 ' GET GRAPHICS ARRAYS
330 CLS:GET (100,100)-(107,107),BLNK:DR
AW BM100,100C2BR2R2FL2NL2D2L3DUR6D
UL3D2LDLDBR6LULULU:GET (100,100)-
(107,107),ROB
340 CLS:DRAW BM100,100C1BD3DFU3RD3EURD
2F2R2BU7L2G2DBRDRBRURBRDF2R3BU5L3G2
D:GET (100,100)-(115,107),SKEL
350 CLS:DRAW BM100,100C2BR5NG3RND3RND
3RF3RN15DNL15DNL15L2DL2NFBLSL2FE":
GET (100,100)-(115,107),CAR
360 CLS:DRAW BM100,100C1BR2R9FDNL6FDNL
7GDGL9HUHUEUFDGUC2URHUEUDFDLUU":
GET (100,100)-(115,107),GARB
370 CLS:FOR Z=1 TO 300:PSET (INT(RND*16
)+100,INT(RND*16)+100),INT(RND*4):N
EXT:GET (100,100)-(115,115),BIGFILL
380 ' DISPLAY PLAYING SCREEN
390 CLS:FOR Z=0 TO 7:LINE (0+Z,16+Z)-(3
19-Z,191-Z),3:B:NEXT:FOR Z=8 TO 296
STEP 16:FOR Y=24 TO 168 STEP 16:PU
T (Z,Y),BIGFILL:NEXT:NEXT:GET (100,
100)-(107,107),FILL
400 SNUG=0:FOR Y=3 TO 22:FOR X=1 TO 37
STEP 2:IF X=16 AND X<24 AND Y<7 THE
N 500 ELSE R=INT(RND*20):IF R<1 THE
N 470 ELSE IF R<3 THEN 480 ELSE IF
R<6 THEN 490 ELSE 500
410 PUT (X*8,Y*8),SKEL,PSET:ART(X,Y)=3:
ART(X+1,Y)=3:SNUG=SNUG+2:GOTO 510
420 PUT (X*8,Y*8),CAR,PSET:ART(X,Y)=2:A
RT(X+1,Y)=2:SNUG=SNUG+2:GOTO 510
430 PUT (X*8,Y*8),GARB,PSET:ART(X,Y)=1:
ART(X+1,Y)=1:SNUG=SNUG+2:GOTO 510
440 ART(X,Y)=9:ART(X+1,Y)=9
450 NEXT:NEXT:FOR Z=0 TO 39:ART(Z,2)=99
:NEXT:ART(19,2)=0:ART(19,3)=0
460 DRAW BM250,9C2NR33E2RER3ER3ER4U2HE
FGUC1DC2D2R4FR3FR3FRF2G2L29H2B265,
7P1,2BM260,10P1,2:FOR Z=0 TO 3:LIN
E (255,12+Z)-(258,12+Z),1:LINE (264
,12+Z)-(268,12+Z),1:LINE (274,12+Z)
(277,12+Z),1:SOUND 300*Z+110,1:NEX
T Z:SOUND 110,.5
470 LOCATE 25,1:PRINT "ARTIFACTS: 0";TA
B(25):CHARGES:10::RETURN
480 ' BEAM DOWN ROBOTS
490 C=2:GOSUB 580:SOUND 200,18:SOUND 11
0,.5:FOR TD=1 TO 100:NEXT:C=0:GOSUB
580:ART(19,1)=0:ART(19,2)=0:ART(19
,3)=0:GOTO 620
500 ON NROB GOTO 610,600,590
510 PUT (24,0),FILL:PUT (24,8),FILL
520 PUT (44,0),FILL:PUT (44,8),FILL
530 LINE (152,0)-(159,31),C,BF:RETURN
540 PUT (24,8),BLNK,PSET:PUT (44,8),BLN
K,PSET:PUT (152,8),BLNK,PSET:ON NRO
B GOTO 650,640,630
550 PUT (24,8),ROB,PSET
560 PUT (44,8),ROB,PSET
570 PUT (152,8),ROB,PSET:ROBX=19:ROBY=1
:N:MNUG::LOCATE 25,34:PRINT C
HG::LOCATE 1,1:PRINT "SCORE:";S
CORE::RETURN
580 ' ROBOT CONTROL ROUTINE
590 ON KJ GOSUB 780,790:IF CHG<1 THEN C
NT=CNT+1:IF CNT>400 THEN GOSUB 1340
:RETURN ELSE IF AS=" " THEN 690 EL
S CNT=CNT+10:GOTO 710
600 IF AS=" " THEN 690
610 IF LEN(AS)=2 THEN GOSUB 850 ELSE IF
AS=CHRS(32) THEN ONR=NROB:GOSUB 10
60:IF ONR<>NROB THEN RETURN
620 IF AS=CHRS(27) THEN GOSUB 1440
630 GOTO 690
640 IF AS">" THEN RETURN ELSE FOR TD=1
TO 250:NEXT:SOUND 880,.5:RETURN
650 ' SCAN KEYBOARD OR JOYSTICK
660
670
680
690

```

```

770 AS=INKEYS:IF AS=" " THEN RETURN ELSE
780 SOUND 1200,3:RETURN
790 C=ABS(AM-A)>ABS(BM-B)):A=STICK(0):
B=STICK(1):IF A<(AM+20) AND C THEN
AS="K" ELSE IF A>(AM+20) AND C THEN C
AS="M" ELSE IF B<(BM+20) AND NOT C
THEN AS="H" ELSE IF B>(BM+20) AND
NOT C THEN AS="P" ELSE AS=" "
800 IF STRIG(1)=1 THEN AS=CHRS(32) EL
S AS=CHRS(0)+AS
810 BS=INKEYS:IF BS=CHRS(27) THEN AS=BS
:RETURN ELSE RETURN
820 ' MOVE ROBOT
830 DIR=INSTR("HKMP",RIGHT$(AS,1)):IF D
IR=0 THEN RETURN ELSE PDIR=DIR:ON D
IR GOTO 860,890,920,950
840 IF ROBY=1 THEN RETURN ELSE IF ROBY=
2 THEN SCN=0:GOTO 880 ELSE IF ROBY=
3 THEN IF ROBX<17 OR ROBY>21 THEN R
ETURN
850 SCN=ART(ROBX,ROBY-1):IF SCN=9 OR SC
N=99 THEN RETURN
860 GOSUB 1010:ROBY=ROBY-1:GOSUB 1020:G
OTO 880
870 IF ROBY=1 THEN RETURN ELSE IF ROBY=
1 THEN IF ROBX<15 THEN RETURN
880 SCN=ART(ROBX-1,ROBY):IF SCN=9 OR SC
N=99 THEN RETURN
890 GOSUB 1010:ROBX=ROBX-1:GOSUB 1020:G
OTO 980
900 IF ROBY=38 THEN RETURN ELSE IF ROBY
=1 THEN IF ROBX=30 THEN GOSUB 1260:
RETURN
910 SCN=ART(ROBX+1,ROBY):IF SCN=9 OR SC
N=99 THEN RETURN
920 GOSUB 1010:ROBX=ROBX+1:GOSUB 1020:G
OTO 980
930 IF ROBY=22 THEN RETURN ELSE IF ROBY
=1 THEN IF ROBX<17 OR ROBX>21 THEN
RETURN
940 SCN=ART(ROBX,ROBY+1):IF SCN=9 OR SC
N=99 THEN RETURN
950 GOSUB 1010:ROBY=ROBY+1:GOSUB 1020:G
OTO 980
960 IF SCN=4 AND SCN<9 THEN MNUG=MNUG+R
A(SCN-6):LOCATE 25,12:PRINT MNUG::
FOR Z=1 TO 100 STEP 1.56:SOUND
SIN(Z)*200+400,.8:NEXT:GOTO 1000
970 IF SCN>0 THEN SOUND 660,1:MNUG=MNUG
+1:(SCN-1)*2+1:SNUG=SNUG-1:LOCATE 25,
12:PRINT MNUG::SOUND 770,.5:SOUND 4
40,1
980 ART(ROBX,ROBY)=0:RETURN
990 PUT (ROBX*8,ROBY*8),BLNK,PSET:RETUR
N
1000 PUT (ROBX*8,ROBY*8),ROB,PSET:SOUND
440,.5:RETURN
1010 ' BLASTER ROUTINE
1020 IF ROBY<3 OR CHG<1 THEN RETURN ELSE
ON PDIR GOTO 1070,1080,1090,1100
1030 BX=ROBX:BY=ROBY-1:GOTO 1110
1040 BX=ROBX-1:BY=ROBY:GOTO 1110
1050 BX=ROBX+1:BY=ROBY:GOTO 1110
1060 BX=ROBX:BY=ROBY+1:GOTO 1110
1070 IF BX-1<1 OR BX+1>38 OR (BY-1<3 AND
BY+1>22 THEN RETURN
1080 SOUND 110,.25:SOUND 220,.5:LINE (BX
*8-7,BY*8-7)-(BX*8+14,BY*8+14),2:LI
NE (BX*8-7,BY*8+14)-(BX*8+14,BY*8-7
),2:SOUND 110,3:SCN1=ART(BX,BY):SCN
2=ART(BX-1,BY-1):SCN3=ART(BX+1,BY-1
):SCN4=ART(BX-1,BY+1):SCN5=ART(BX+1
,BY+1)
1090 PUT (BX*8-8,BY*8-8),BLNK,PSET:PUT (
BX*8+8,BY*8-8),BLNK,PSET:PUT (BX*8,
BY*8),BLNK,PSET:PUT (BX*8-8,BY*8+8)
,BLNK,PSET:PUT (BX*8+8,BY*8+8),BLNK
,PSET
1100 IF SCN1>0 AND SCN1<4 THEN SNUG=SNUG
-1
1110 IF SCN2>0 AND SCN2<4 THEN SNUG=SNUG
-1
1120 IF SCN3>0 AND SCN3<4 THEN SNUG=SNUG
-1
1130 IF SCN4>0 AND SCN4<4 THEN SNUG=SNUG
-1
1140 IF SCN5>0 AND SCN5<4 THEN SNUG=SNUG
-1
1150 ART(BX,BY)=0:ART(BX-1,BY-1)=0:ART(B
X+1,BY+1)=0:ART(BX-1,BY+1)=0:ART(BX
+1,BY-1)=0:CHG=CHG-1:LOCATE 25,34:P
RINT CHG:
1160 FOR Z=1 TO (MNUG+50)*.2:CIX=INT(RND
*37)+2:CIY=INT(RND*19)+4:SCN=ART(CI
X,CIY):IF SCN<>0 THEN 1220
1170 PUT (CIX*8,CIY*8),FILL,PSET:IF ROBX
=CIX AND ROBY=CIY THEN GOSUB 1340 E
LSE ART(CIX,CIY)=9
1180 NEXT:RETURN
1190 ' ROBOT RETURNS TO THE SHIP
1200
1210
1220
1230
1240
1250

```

Continued


```

1260 IF MNUG>0 THEN FOR Z=110 TO 860 STEP 10
P 5: SOUND 25,12: PRINT MNUG;
LOCATE 1,7: PRINT SCORE;
1270 IF CHG=0 THEN CHG=10: CNT=0: FOR Z=1
TO 3: SOUND 440,3: SOUND 497,3: NEXT: L
OCATE 25,34: PRINT CHG;
1280 IF SNUG=0 THEN GOSUB 450: PCHARGE=CH
G: GOSUB 570: RETURN ELSE RETURN
1290 IF CHG=0 THEN SCORE=SCORE+(MNUG*5)
ELSE SCORE=SCORE+(MNUG*10*CHG)
1300 RETURN
1310 ' ROBOT OUT OF POWER
1320
1330
1340 FOR Z=660 TO 110 STEP -5: SOUND Z,.1
BY: 8+7): DRAW "C2U4L3UR3U2RD2R3DL3D4O
L": SOUND 440,5: ART (ROBX,ROBY)=5+NRO
B: IF NROB>1 THEN RIA (NROB-1)=MNUG

```

```

1350 MNUG=0: NROB=NROB-1: IF NROB>0 THEN P
UT (24,8),BLNK,PSET: PUT (44,8),BLNK
1360 RETURN
1370
1380 ' SCAN KEYBOARD
1390
1400 AS="": WHILE AS="" : AS=INKEY$: WEND: RE
TURN
1410
1420 ' END OF GAME
1430
1440 CLS: LOCATE 12,1: PRINT "YOUR SCORE I
S": SCORE: LOCATE 18,1: PRINT "PLAY AG
AIN (Y/N)?"
1450 GOSUB 1400: IF AS="N" THEN 1450 ELSE I
F AS="Y" OR AS="R" THEN END ELSE I
F AS<>"Y" AND AS<>"R" THEN 1450 ELS
E CLEAR: GOTO 260

```

HCM

ARCHEODROID

TI-99/4A

```

100 REM *****
110 REM ***** ARCHEODROID *****
120 REM *****
130 REM *****
140 REM ***** COPYRIGHT 1983, 1984, 1985
150 REM ***** EMERALD VALLEY PUBLISHING CO. *****
160 REM ***** BY B. J. BRUNS *****
170 REM ***** HOME COMPUTER MAGAZINE *****
180 REM ***** AND THE HCM STAFF *****
190 REM ***** VERSION 5.4.1 *****
200 REM ***** TI BASIC OR *****
210 REM ***** TI EXTENDED BASIC *****
220 CALL CLEAR
230 GOTO 3440
240 CALL HCHAR(3,1,125,32)
250 CALL HCHAR(3,14,120,5)
260 CALL HCHAR(4,1,120,32)
270 CALL HCHAR(22,1,125,32)
280 CALL VCHAR(5,1,125,17)
290 CALL VCHAR(5,32,125,17)
300 SNUG=0
310 FOR R2=5 TO 21
320 FOR C2=2 TO 30 STEP 2
330 RANDOMIZE
340 T=INT(RND*3)+1
350 ON T GOTO 470,470,360
360 ON INT(RND*10)+1 GOTO 370,370,370,3
370 CALL HCHAR(R2,C2,116)
380 CALL HCHAR(R2,C2+1,117)
390 GOTO 450
400 CALL HCHAR(R2,C2,112)
410 CALL HCHAR(R2,C2+1,113)
420 GOTO 450
430 CALL HCHAR(R2,C2,114)
440 CALL HCHAR(R2,C2+1,115)
450 SNUG=SNUG+2
460 GOTO 480
470 CALL HCHAR(R2,C2,120,2)
480 NEXT C2
490 NEXT R2
500 R1=1
510 C1=19
520 AS=SH1$
530 GOSUB 2730
540 R1=2
550 AS=SH2$
560 GOSUB 2730
570 FOR LD=1 TO 4
580 CALL SOUND(-4000,220,LD,30/(LD^2*2)
)
590 CALL HCHAR(2,21,135+LD)
600 CALL HCHAR(2,22,139+LD)
610 FOR TD=1 TO 50
620 NEXT TD
630 NEXT LD
640 CALL SOUND(3000,110,0,220,0,440,0,-
3,0)
650 ON MINER GOTO 680,670,660
660 CALL VCHAR(1,8,107,2)
670 CALL VCHAR(1,10,107,2)
680 CALL VCHAR(1,16,107,4)
690 CALL SOUND(300,500,0,0,-6,0)
700 ON MINER GOTO 730,720,710
710 CALL VCHAR(1,8,32,2)
720 CALL VCHAR(1,10,32,2)
730 CALL VCHAR(1,16,32,4)
740 CALL HCHAR(2,1,32,19)
750 ON MINER GOTO 780,770,760
760 CALL HCHAR(2,8,64)
770 CALL HCHAR(2,10,64)
780 CALL HCHAR(2,16,64)
790 C=16
800 ON MINER GOTO 860,830,810
810 CALL HCHAR(2,16,64)
820 GOTO 880
830 CALL HCHAR(2,8,32)
840 CALL HCHAR(2,16,64)
850 GOTO 880
860 CALL HCHAR(2,10,32)
870 CALL HCHAR(2,16,64)
880 CALL KEY(1,KEY,ST)

```

```

890 IF ST=0 THEN 950
900 IF KEY=11 THEN 1610
910 IF KEY>5 THEN 880
920 IF KEY<1 THEN 930 ELSE 940
930 KEY=0
940 ON KEY+1 GOTO 1180,880,1090,1270,88
0,1060
950 CALL KEY(2,KEY,ST)
960 IF KEY=18 THEN 1610
970 CALL JOYST(2,X,Y)
980 KEY=((X+3*Y)/4)+5
990 ON KEY GOTO 880,1180,880,1090,880,1
270,880,1000,880
IF R-1<2 THEN 880
1000 PO=1
1010 CALL GCHAR(R-1,C,CH)
1020 CALL SOUND(75,440,5)
1030 CALL ((CH=32)+(CH=132))+((CH>111)*(CH<1
18))+((CH=91) THEN 1050 ELSE 880
1040 IF CH=132 THEN 2490
1050 CALL HCHAR(R,C,32)
1060 R=R-1
1070 GOTO 1350
1080 IF C-1<1 THEN 880
1090 PO=2
1100 CALL GCHAR(R,C-1,CH)
1110 CALL SOUND(75,440,5)
1120 CALL ((CH=32)+(CH=132))+((CH>111)*(CH<1
18))+((CH=91) THEN 1140 ELSE 880
1130 IF CH=132 THEN 2490
1140 CALL HCHAR(R,C,32)
1150 C=C-1
1160 GOTO 1350
1170 IF R+1>22 THEN 880
1180 PO=3
1190 CALL GCHAR(R+1,C,CH)
1200 CALL SOUND(75,440,5)
1210 CALL ((CH=32)+(CH=132))+((CH>111)*(CH<1
18))+((CH=91) THEN 1230 ELSE 880
1220 IF CH=132 THEN 2490
1230 CALL HCHAR(R,C,32)
1240 R=R+1
1250 GOTO 1350
1260 IF C+1>32 THEN 880
1270 PO=4
1280 CALL GCHAR(R,C+1,CH)
1290 CALL SOUND(75,440,5)
1300 CALL ((CH=32)+(CH=132))+((CH>111)*(CH<1
18))+((CH=91) THEN 1320 ELSE 880
1310 IF CH=132 THEN 2490
1320 CALL HCHAR(R,C,32)
1330 C=C+1
1340 IF CH=32 THEN 1510
1350 IF CH=91 THEN 1540
1360 SNUG=SNUG-1
1370 IF (CH<>112)*(CH<>113) THEN 1410
1380 MNUG=MNUG+3
1390 GOTO 1460
1400 IF (CH<>114)*(CH<>115) THEN 1440
1410 MNUG=MNUG+5
1420 GOTO 1460
1430 IF (CH<>116)*(CH<>117) THEN 1470
1440 MNUG=MNUG+1
1450 CALL SOUND(200,-6,0)
1460 R1=23
1470 C1=12
1480 AS=STR$(MNUG)
1490 GOSUB 2730
1500 CALL HCHAR(R,C,64)
1510 CALL SOUND(100,-5,2)
1520 GOTO 880
1530 MNUG=MNUG+DNUG
1540 DNUG=0
1550 R1=23
1560 C1=12
1570 AS=STR$(MNUG)
1580 GOSUB 2730
1590 GOTO 1510
1600 IF (R<2)+(R>21)+(C<2)+(C>31) THEN 88
0
1610
1620 ON PO GOSUB 1820,1850,1880,1910

```

Continued

TYPE-IN LISTINGS

```

1630 IF (C1<2)+(C1>31)+(R1<2)+(R1>21) THEN  

1640 CALL GCHAR(R1,C1,CH1)  

1650 CALL GCHAR(R1+1,C1+1,CH2)  

1660 CALL GCHAR(R1+1,C1+1,CH3)  

1670 CALL GCHAR(R1+1,C1+1,CH4)  

1680 CALL GCHAR(R1+1,C1+1,CH5)  

1690 GOSUB 1940  

1700 CALL HCHAR(R1,C1,104)  

1710 CALL HCHAR(R1+1,C1+1,105)  

1720 CALL HCHAR(R1+1,C1+1,106)  

1730 CALL HCHAR(R1+1,C1+1,106)  

1740 CALL HCHAR(R1+1,C1+1,105)  

1750 CALL SOUND(1000,110,7,2)  

1760 CALL HCHAR(R1+1,C1+1,32)  

1770 CALL HCHAR(R1+1,C1,32)  

1780 CALL HCHAR(R1+1,C1+1,32)  

1790 CALL HCHAR(R1+1,C1+1,32)  

1800 CALL HCHAR(R1+1,C1+1,32)  

1810 GOTO 2060  

1820 R1=R-1  

1830 C1=C-1  

1840 RETURN  

1850 C1=C-1  

1860 R1=R-1  

1870 RETURN  

1880 R1=R+1  

1890 C1=C  

1900 RETURN  

1910 C1=C+1  

1920 R1=R  

1930 RETURN  

1940 IF (CH1=125)+(CH2=125)+(CH3=125)+(C  

H4=125)+(CH5=125) THEN 880  

1950 IF (CH1<112)+(CH1>117) THEN 1970  

1960 SNUG=SNUG-1  

1970 IF (CH2<112)+(CH2>117) THEN 1990  

1980 SNUG=SNUG-1  

1990 IF (CH3<112)+(CH3>117) THEN 2010  

2000 SNUG=SNUG-1  

2010 IF (CH4<112)+(CH4>117) THEN 2030  

2020 SNUG=SNUG-1  

2030 IF (CH5<112)+(CH5>117) THEN 2050  

2040 SNUG=SNUG-1  

2050 RETURN  

2060 CHARGES=CHARGES-1  

2070 R1=23  

2080 C1=27  

2090 A$=STR$(CHARGES)  

2100 CALL HCHAR(23,27,32,3)  

2110 GOSUB 2740  

2120 FOR I=1 TO 10  

2130 R1=INT(RND*17)+4  

2140 C1=INT(RND*29)+2  

2150 CALL GCHAR(R1,C1,CH)  

2160 IF (CH>111)*(CH<118) THEN 2470  

2170 IF CH=64 THEN 2210  

2180 IF CH<32 THEN 2470  

2190 CALL HCHAR(R1,C1,120)  

2200 GOTO 2470  

2210 DNUG=MNUG  

2220 IF (R<>2)+(DNUG=0) THEN 2290  

2230 TOTAL=DNUG*10+TOTAL  

2240 A$=STR$(TOTAL)  

2250 R1=2  

2260 C1=24  

2270 GOSUB 2750  

2280 DNUG=0  

2290 RESTORE 3750  

2300 CALL HCHAR(R,C,94)  

2310 MNUG=0  

2320 CALL HCHAR(23,12,32,5)  

2330 MINER=MINER-1  

2340 R=2  

2350 C=16  

2360 CHARGES=10  

2370 R1=23  

2380 C1=27  

2390 A$=STR$(CHARGES)  

2400 GOSUB 2740  

2410 RESTORE 3750  

2420 READ A,B  

2430 IF A=0 THEN 2460  

2440 CALL SOUND(A,B,0)  

2450 GOTO 2420  

2460 IF MINER>0 THEN 640 ELSE 2800  

2470 NEXT I  

2480 IF CHARGES=0 THEN 2880 ELSE 880  

2490 IF CHARGES>0 THEN 2510  

2500 CHARGES=.5  

2510 TOTAL=10*(MNUG+CHARGES)+TOTAL  

2520 IF MNUG=0 THEN 2560  

2530 FOR SND=440 TO 3000 STEP 50  

2540 CALL SOUND(-30,SND,0)  

2550 NEXT SND  

2560 IF CHARGES>=1 THEN 2640  

2570 CHARGES=10  

2580 R1=23  

2590 C1=27  

2600 CALL HCHAR(23,27,32,3)  

2610 A$=STR$(CHARGES)  

2620 CALL HCHAR(2,24,32,8)  

2630 GOSUB 2740  

2640 MNUG=0  

2650 R1=2  

2660 C1=24  

2670 A$=STR$(TOTAL)  

2680 GOSUB 2740

```

```

2690 CALL HCHAR(23,12,32,5)
2700 IF SNUG>0 THEN 880
2710 SCREEN=SCREEN+1
2720 GOTO 240
2730 CALL HCHAR(23,12,32,5)
2740 REM PRINT AT ROUTINE
2750 FOR Z=1 TO LEN(AS)
2760 BS=SEG$(AS,Z,1)
2770 CALL HCHAR(R1,C1+Z,ASC(BS))
2780 NEXT Z
2790 RETURN
2800 CALL CLEAR
2810 CALL SOUND(4000,110,5)
2820 PRINT "THE ARTIFACTS YOU HAVE": "ARE
      WORTH TOTAL: CREDITS": "WANT TO PLAY AGAIN? Y/N":
2830 CALL KEY(3,KEY,ST)
2840 IF ST=0 THEN 2830
2850 IF KEY=78 THEN 2870
2860 IF KEY=89 THEN 3700 ELSE 2830
2870 END
2880 TIME=0
2890 TIME=TIME+1
2900 IF TIME=40 THEN 2210
2910 CALL KEY(1,KEY,ST)
2920 IF ST=0 THEN 2970
2930 IF KEY=5 THEN 2890
2940 IF KEY>1 THEN 2960 ELSE 2950
2950 KEY=0
2960 ON KEY+1 GOTO 3140,2890,3070,3210,2890,3000
2970 CALL JOYST(2,X,Y)
2980 KEY=((X+3*Y)/4)+5
2990 ON KEY GOTO 2890,3140,2890,3070,2890,3210,2890,3000,2890
3000 IF R=1<2 THEN 2890
3010 CALL GCHAR(R-1,C,CH)
3020 IF (CH=32)+(CH=132)+((CH>111)*(CH<118)) THEN 3030 ELSE 2890
3030 IF CH=132 THEN 2490
3040 CALL HCHAR(R,C,32)
3050 R=R-1
3060 GOTO 3270
3070 IF C=1<1 THEN 2890
3080 CALL GCHAR(R,C-1,CH)
3090 IF (CH=32)+(CH=132)+((CH>111)*(CH<118)) THEN 3100 ELSE 2890
3100 IF CH=132 THEN 2490
3110 CALL HCHAR(R,C,32)
3120 C=C-1
3130 GOTO 3270
3140 IF R+1>22 THEN 2890
3150 CALL GCHAR(R+1,C,CH)
3160 IF (CH=32)+(CH=132)+((CH>111)*(CH<118)) THEN 3170 ELSE 2890
3170 IF CH=132 THEN 2490
3180 CALL HCHAR(R,C,32)
3190 R=R+1
3200 GOTO 3270
3210 IF C+1>32 THEN 2890
3220 CALL GCHAR(R,C+1,CH)
3230 IF (CH=32)+(CH=132)+((CH>111)*(CH<118)) THEN 3240 ELSE 2890
3240 IF CH=132 THEN 2490
3250 CALL HCHAR(R,C,32)
3260 C=C+1
3270 IF CH=32 THEN 3410
3280 SNUG=SNUG-1
3290 IF (CH<>112)*(CH<>113) THEN 3320
3300 MNUG=MNUG+3
3310 GOTO 3370
3320 IF (CH<>114)*(CH<>115) THEN 3350
3330 MNUG=MNUG+5
3340 GOTO 3370
3350 IF (CH<>116)*(CH<>117) THEN 3370
3360 MNUG=MNUG+1
3370 R1=23
3380 C1=12
3390 AS=STR$(MNUG)
3400 GOSUB 2730
3410 CALL HCHAR(R,C,64)
3420 CALL SOUND(100,-5,0)
3430 GOTO 2890
3440 PRINT TAB(8); "ARCHEODROID":
      PRINT "PRESS ENTER TO START"
3450 CALL KEY(0,K,S)
3460 IF (K=13)*(S<>0) THEN 3490
3470 GOTO 3460
3480 CALL CLEAR
3490 PRINT "CHECK ALPHA LOCK": "DOWN FOR
      KEYBOARD": "UP FOR JOYSTICKS":
3500 RESTORE 3760
3510 FOR Z=1 TO 30
3520 READ A,AS
3530 CALL CHAR(A,AS)
3540 NEXT Z
3550 SH1$=CHR$(128)&CHR$(129)&CHR$(130)&CHR$(131)
3560 SH2$=CHR$(132)&CHR$(133)&CHR$(134)&CHR$(135)
3570 CALL COLOR(10,7,11)
3580 CALL COLOR(12,2,11)
3590 CALL COLOR(13,7,1)
3600 CALL COLOR(14,7,1)
3610 CALL COLOR(11,2,11)
3620 FOR TD=1 TO 10
3630 TB=INT(RND*23)+1
3640

```

Continued

Continued

TI-99/4A

```

3650 PRINT TAB(18); "SH1:"
3660 PRINT TAB(18); "SH2:"
3670 CALL SOUND(5, RND*10000+10000, 30, 5, RND*200+10, 5, RND*10000+10000, 30, -4, 5)
3680 NEXT I
3690 PRINT:
3700 NEXT I
3710 READ R, C, PO, TOTAL, CHARGES, SCREEN, MINER, MNU, TAB(22); "SCORE": TAB(23); "00":
3720 PRINT:
3730 PRINT:
3740 GOTO 240
3750 DATA 750, 117, 750, 117, 183, 117, 558, 117, 139, 183, 131, 558, 131, 183, 117, 558, 117, 750, 110, 1500, 117, 0.0

```

[illegible]

HCM

MINE OVER MATTER

APPLE // Family

```

100 REM *** MINE OVER MATTER ***
110 REM *****
120 REM *****
130 REM *****
140 REM *****
150 REM *****
160 REM *****
170 REM *****
180 REM *****
190 REM *****
200 REM *****
210 REM *****
220 REM *****
230 REM *****
240 REM *****
250 REM *****
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 NEXT P: S2 = 1: GOSUB 620: NEXT X: P
510 PRINT CHR$(7): GOTO 1770
520 REM
530 REM *** MAIN CONTROL MENU ***
540 POKE 49232,0: HOME: VTAB 21: PRINT
    PLS(P): S CASH = "": Q = T(P,1): GO
    SUB 1710
550 PRINT "1) LAND SURVEY 4) NEX
    T TURN"
560 PRINT "2) PRODUCTION": PRINT "
    3) MINE REPORT ">":
570 VTAB 24: HTAB 24: GET A$: IF A$ < "
    THEN A$ = "
580 PRINT A$: IF A$ < "1" OR A$ > "4"
    THEN 570
590 IF A$ = "4" THEN RETURN
600 ON VAL (A$) GOSUB 1400,1020,890: G
    OTO 540
610 REM
620 REM *** END OF YEAR ROUTINE ***
630 REM
640 D = FRE (0): POKE 49233,0: GOSUB 1
    700: VTAB 12: HTAB 10: PRINT "HA
    PPY NEW YEAR "R + X + 1" *
    FOR P = 1 TO J: FOR X = 1 TO 5: IF
    MP(P,Z,5) = 0 THEN 800
    660 I = MP(P,Z,7): H = MP(P,Z,6)
    670 ON MP(P,Z,5) GOTO 680,800,680,800,7
    00,740,680,770,770,770
    680 MP(P,Z,5) = MP(P,Z,5) + 1: IF MP(P,
    Z,5) = 8 THEN T(P,1) = T(P,1) + MP(
    P,Z,2): MP(P,Z,2) = 0
    690 GOTO 800
    700 GOSUB 1650: T(P,1) + S * M
    MP(P,Z,3): MP(P,Z,8) = MP(P,Z,8) -
    S * 001: MP(P,Z,4) = MP(P,Z,4) + 1
    IF MP(P,Z,8) < 2 THEN MP(P,Z,5) =
    6
    710 IF INT (MP(P,Z,1) * .00035) = S AN
    D INT (RND (1) * 5) = 1 THEN MP(P,
    Z,5) = 10: MP(P,Z,3) = 0: GOTO 800
    720 IF INT (RND (1) * (100 - MP(P,Z,1
    1))) = 1 THEN MP(P,Z,5) = 10: MP(P,Z
    3) = 0
    730 GOTO 800
    740 IF MP(P,Z,4) > 0 THEN MP(P,Z,4)
    = 3: MP(P,Z,3) = 0: GOTO 800
    750 MP(P,Z,4) = MP(P,Z,4) + 1: IF MP(P,
    Z,4) < 0 THEN 800
    760 MP(P,Z,5) = 10: HCOLOR = 5: CALL HCH
    AR I,H,1,0: S%(I,H) = 4: HCOLOR = 3
    : GOTO 800
    770 IF MP(P,Z,5) = 8 THEN CALL HCHAR, I
    ,H,2,1,0: S%(I,H) = 2: GOTO 790
    780 HCOLOR = 5: CALL HCHAR, I,H,0,1,0: S%(
    I,H) = 4: HCOLOR = 3
    790 FOR N = 1 TO 8: MP(P,Z,N) = 0: NEXT
    : T(P,2) = T(P,2) - 1
    800 NEXT Z: IF T(P,1) > 0 THEN 830
    810 VTAB 15 + P: HTAB 1: PRINT PLS(P):
    TAB (25) "BANKRUPT": FOR Z = 1 TO 5
    : IF MP(P,Z,5) > 0 THEN CALL HCHAR
    : H,2,1,0: S%(I,H) = 3
    820 NEXT Z: GOTO 840
    830 VTAB 15 + P: HTAB 1: PRINT PLS(P):
    Q = T(P,1): GOSUB 1710
    840 NEXT P: M = M + INT (RND (1) * 155
    2) - 776: IF M < 500 THEN M = INT
    (RND (1) * 2000) + 3000
    850 VTAB 23: HTAB 6: PRINT "PRESS ANY K
    EY TO CONTINUE -->": GET A$: PRIN
    T A$: RETURN
    860 REM
    870 REM *** PRINT REPORTS ***
    880 REM
    890 IF T(P,2) = 0 THEN RETURN
    900 GOSUB 1700: VTAB 12: FOR Z = 1 TO 5
    : IF MP(P,Z,5) < 0 THEN PRINT
    : CHR$(K + Z) > "": MSG$(MP(P,Z
    ,5)): PRINT
    910 NEXT Z

```

Continued

TYPE-IN LISTINGS


```

920 VTAB 8: HTAB 14: PRINT "SELECT ONE (A
$) : K: GET IF Z < 1 OR Z > 5 THEN RET
930 IF MP(P,Z,5) = 0 THEN RETURN
940 GOSUB 1650: VREPORT FOR MILL: A$:
PRINT "CASH ON HAND": Q = T(P,1): G
OSUB 1710: PRINT "MILL COST": Q = M
P(P,Z,1): GOSUB 1710: PRINT "BOND PR
OSTINT: Q = MP(P,Z,2): GOSUB 1710: PR
INT "BARRELS PROD.": Q = S: GOSUB 1
720
960 PRINT "MAX PRODUCTION": Q = MP(P,Z,
1): GOSUB 1720: PRINT "VALU
UE/BARREL": Q = M: GOSUB 1710: PRIN
T "GROSS": Q = S * M: GOSUB 1710: P
GOSUB 1710: Q = S * M: MP(P,Z,3): P
970 PRINT "QUALITY": Q = MP(P,Z
,9): GOSUB 1720: PRINT "DEPT
MP(P,Z,10): GOSUB 1720: PRINT "ENV
IRONMENT": Q = MP(P,Z,11): GOSUB 17
20
980 VTAB 23: HTAB 6: PRINT "PRESS ANY K
EY TO CONTINUE": GET A$: PRINT
A$: RETURN
990 REM *** PRODUCTION ROUTINES ***
1000 REM
1010 REM
1020 IF T(P,2) = 0 THEN RETURN
1030 FOR Z = 1 TO 5: IF MP(P,Z,5) < 2 OR
MP(P,Z,5) > 6 THEN NEXT Z: RET
POKE 49232,0: HOME: VTAB 22: PRINT
"1") SET PRODUCTION RATE: PRINT
2) START RECLAMATION: PRINT
2) EXPAND MINE RECLAMATION: PRINT
HTAB 23: PRINT "SELECT (1-3) >":
GET A$: Z = A: VAL (A$)
1060 IF Z > 0 AND Z < 4 THEN ON Z GOSUB
1080, 1190, 1250: GOTO 1020
1070 RETURN: VTAB 22: HTAB 10: PRINT "WHI
HOME: VTAB 22: HTAB 10: PRINT "WHI
CH MINE (K + 1)": CHR$(K + 1)
1090 GET A$: PRINT A$: Z = A: ASC (A$) - K:
IF Z < 1 OR Z > 5 THEN RETURN
1100 IF MP(P,Z,5) < 4 OR MP(P,Z,5) > 5 T
HEN GOTO 1080
1110 GOSUB 1700: VTAB 8: PRINT "PRODUCTI
ON FOR MILL": A$: VTAB 10: PRINT
"CASH": Q = T(P,1): GOSUB 1710: PRIN
T "CURRENT PRODUCTION": Q = MP(
P,Z,3): GOSUB 1710: PRINT "PER
1720: GOSUB 1710: PRINT "PER BARREL": Q
= M: GOSUB 1710: PRINT "GROSS": Q = S * M: GOSUB 171
3) : GOSUB 1710: PRINT "MAX PRODUCTI
ON": Q = MP(P,Z,1) * .00035: GOSUB
1720
1140 VTAB 18: HTAB 1: CALL 958: PRINT
"NEW PRODUCTION": MP(P,Z,9): HT
AB 16: INPUT ? : A$: VTAB 12: PRINT
IF LEN (A$) > 12 THEN RETURN
1160 IF VAL (A$) < 0 OR VAL (A$) > T(P
,1) THEN RETURN
1170 IF (VAL (A$)) > MP(P,Z,5) = INT
(VAL (A$)): MP(P,Z,5) = 5
1180 RETURN: VTAB 22: PRINT "START RECLAM
ATION ON WHICH MINE (K + 1)":
1190 GET A$: PRINT A$: Z = A: ASC (A$) - (K
+ 1): IF Z < 1 OR Z > 5 THEN RETURN
1210 IF MP(P,Z,5) < 4 OR MP(P,Z,5) > 6 T
HEN RETURN
1220 D = MP(P,Z,2) * .4: IF T(P,1) - D <
0 THEN HOME: VTAB 22: HTAB 12: P
RINT "NOT ENOUGH MONEY": CHR$(7)
: GOSUB 1690: RETURN
1230 T(P,1) = T(P,1) - D: MP(P,Z,5) = 7: M
P(P,Z,3) = MP(P,Z,4): HOME:
VTAB 22: PRINT "RECLAMATION STARTED
ON MINE": A$:
1240 PRINT "RECLAMATION COST": Q = D: GO
SUB 1710: INVERSE: CALL HCHAR, MP(P
,Z,7), MP(P,Z,6), 20, 1, 0: NORMAL: S%
(MP(P,Z,7), MP(P,Z,6)) = 3: GOSUB 169
0: GOSUB 1690: RETURN
1250 HOME: VTAB 22: HTAB 5: PRINT "EXPA
ND ON WHICH MINE (K + 1)":
1260 GET A$: PRINT A$: Z = A: ASC (A$) - K:
IF Z < 1 OR Z > 5 THEN RETURN
1270 IF MP(P,Z,5) < 2 OR MP(P,Z,5) > 5 T
HEN RETURN
1280 GOSUB 1700: VTAB 8: PRINT "MILL
EXPANSION REPORT FOR MINE": A$:
1290 VTAB 10: PRINT "CASH": Q = T(P,1):
GOSUB 1710: PRINT "MILL SIZE": Q =
MP(P,Z,1): GOSUB 1710: PRINT "MAX P
ROD.": Q = MP(P,Z,1) * .00035: GOSU
B 1720

```

```

1300 PRINT "QUALITY": Q = MP(P,Z
,9): GOSUB 1720: PRINT "DEPT
MP(P,Z,10): GOSUB 1720: PRINT "ENV
IRONMENT": Q = MP(P,Z,11): GOSUB 17
20
1310 VTAB 18: HTAB 1: CALL 958: INPUT
INCREASE CONST: S$: A$:
1320 IF LEN (A$) > 12 THEN RETURN
1330 D = VAL (A$): IF D < 1 OR D > T(P
,1) THEN RETURN
1340 PRINT "GOSUB 1670: IF T(P,1) - D -
F < 0 THEN VTAB 23: PRINT
YOU CANT AFFORD THAT MUCH!!" CHR$(
7): GOSUB 1690: RETURN
1350 MP(P,Z,1) = MP(P,Z,1) + D: T(P,1) =
T(P,1) + D: F: MP(P,Z,2) = MP(P,Z,2
) + F: MP(P,Z,3) = 0: MP(P,Z,5) = 3
1360 VTAB 23: PRINT "PRESS ANY KE
Y FOR MENU": GET A$: PRINT A$:
RETURN
1370 REM *** SURVEY ROUTINE ***
1380 REM
1390 REM
1400 IF T(P,2) = 5 THEN RETURN
1410 HOME: VTAB 21: PRINT "PLS(P)" 'S CAS
H: Q = T(P,1): GOSUB 1710
1420 H = 20: I = 10: PRINT "USE
THE I/J/K/M & ARROW KEYS TO MOVE,
SPACE TO SELECT & RETURN TO KE
EP":
1430 CALL HCHAR, I, H, 0, 1, 1: C = S%(I,H): G
ET A$: PRINT A$: F = ASC (A$) - 12
8: H = 0: I = 1: IF (ASC (A$) > 127): CALL HCHAR, I
, H, 0, 1, 1
1440 IF C = 1: THEN HCOLOR = 6: CALL HCHA
R, I, H, 1, 1, 0: HCOLOR = 3: S%(I,H) = 1
1450 IF (F = 73 OR F = 11) AND I > 2 THE
N I = I - 1
1460 IF (F = 77 OR F = 10) AND I < 19 TH
EN I = I + 1
1470 IF (F = 74 OR F = 8) AND H > 3 THEN
H = H - 1
1480 IF (F = 75 OR F = 21) AND H < 38 TH
EN H = H + 1
1490 IF F = 32 AND (C = 0 OR C = 1) THEN
V(3) = V(3) + 1
1500 IF F = 13 THEN 1580
1510 GOTO 1430
1520 V1 = 1: V2 = H: GOSUB 1790
1530 D = 1000: IF C < > 1 THEN D = V(2)
* .115 + 1000
1540 IF T(P,1) - D < 0 THEN GOSUB 1700:
VTAB 15: HTAB 13: PRINT "NOT ENOU
H MONEY": CHR$(7): GOSUB 1690: RET
URN
1550 T(P,1) = T(P,1) - D: HOME: VTAB 21
: PRINT "PLS(P)" 'S CASH =": Q = T(P
,1): GOSUB 1710
1560 VTAB 22: PRINT "SURVEY COST": Q =
D: GOSUB 1710: PRINT "PRINT "QUAL
ITY": V(1): DEPTH = "V(2)" ENVMT
= "V(3)":
1570 CALL HCHAR, I, H, 21, 1, 0: S%(I,H) = 5:
GOTO 1430
1580 IF C < 5 THEN RETURN
1590 T(P,2) = T(P,2) + 1: FOR Z = 1 TO 5
: IF MP(P,Z,5) < > 0 THEN NEXT Z
RETURN
1600 CALL HCHAR, I, H, K - 62 + Z, 1, 0: S%(I
,H) = F: F = 59 + Z
1610 MP(P,Z,5) = 1: MP(P,Z,6) = H: MP(P,Z
,7) = 1: MP(P,Z,8) = V(4): MP(P,Z,9) =
V(1): MP(P,Z,10) = V(2): MP(P,Z,11) =
V(3): RETURN
1620 REM *** MISC. ROUTINES ***
1630 REM
1640 REM
1650 S = (MP(P,Z,3) / (.5 + MP(P,Z,10) +
(MP(P,Z,11) * .5)): S = S * ((MP(
P,Z,8) * .5) + MP(P,Z,9)) * .0003:
Q = MP(P,Z,1) * .00035: IF S > Q TH
EN S = Q
1660 INT (S): RETURN
1670 F = INT ((MP(P,Z,11) * .0133) * D)
: IF MP(P,Z,1) = 0 THEN F = F + 500
00
1680 PRINT "RECLAMATION =": Q = F: GOS
UB 1710: RETURN
1690 FOR N = 1 TO 2000: NEXT: RETURN
1700 POKE 49233,0: HOME: FLASH: FOR N
= 2 TO 4: VTAB N: HTAB 9: PRINT SP
CB: 10: PRINT "MINE OVER MATTER"
: RETURN
1710 HTAB 25: PRINT "$":
1720 Q = INT (Q): C = Q: NS = STR$(ABS
(Q)): A$ = HTAB 26: PRINT SPC(
12):
1730 N = LEN (NS): FOR Q = 1 TO N: A$ =
MID$(NS, N - Q + 1, 1) + A$: IF (Q
/ 3) = INT (Q / 3) AND Q < > N TH
EN A$ = INT (Q / 3) + A$
1740 NEXT: HTAB 37: LEN (A$): PRINT
CHRS (32 + 13 * (C < 0)): A$: RETURN
1750 HOME: VTAB 22: PRINT "ARE YOU SUR
E YOU WANT TO EXIT ?N (Y/N)": HTAB
33: GET A$: PRINT A$

```

Continued

Continued

APPLE II Family

```

1760 IF GOSUB = PRINT J: RETURN
1770 IF GOSUB = PRINT J: RETURN
1780 IF GOSUB = PRINT J: RETURN
1790 IF GOSUB = PRINT J: RETURN
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

```

```

0080 INPUT E,J,M,S2,R,S1
2090 PRINT CHR$(4); "CLOSE"
2100 RETURN
2110 RETURN
2120 REM *** ERROR HANDLING ***
2130 Z = PEEK(222): D = PEEK(218) +
      PEEK(219) * 256: PRINT CHR$(7)
2140 CALL 3288
2150 VTAB 21: CALL 958: PRINT: IF Z
      = 4 THEN PRINT "WRITE PROTECTED":
      GOTO 2220
2160 IF Z = 6 OR Z = 7 THEN PRINT "FILE
      NOT FOUND": GOTO 2220
2170 IF Z = 8 THEN PRINT "I/O ERROR": G
      OTO 2220
2180 IF Z = 9 THEN PRINT "DISK FULL": G
      OTO 2220
2190 IF Z = 11 THEN PRINT "NOT A VALID
      FILENAME": GOTO 2220
2200 IF Z = 255 THEN PRINT "PROGRAM ABO
      RTED ON LINE #": PRINT
2210 TEXT: HOME: "D: END
      Z": IN LINE #": D: END
      "FATAL ERROR #"
2220 GOSUB 1690: IF FLAG = 1 THEN RUN
2230 GOTO 1870
2240 REM
2250 REM
2260 REM
2270 DATA *** PROGRAM DATA ***
      DATA SURVEY IN PROGRESS, SURVEY C
      OMPLETE, UNDER CONSTRUCTION, CONSTR
      UCTION COMPLETED, IN PRODUCTION, OU
      T OF ORE
2280 DATA RECLAMATION STARTED, RECLAMATI
      ON DONE, MINE CLOSED, CLOSED FOR CONT
      AMINATION
2290 DATA 32,76,231,202,134,37,32,76,231,2
      202,134,36,32,76,231,134,8,0,32,76,37
      31,202,134,6,32,76,231,134,41,1,240,4
      32,34,252,164,228,165,38,201,7,240,4
2300 DATA 13,152,41,7,240,8,201,7,240,4,
      152,73,127,168,132,9,32,69,8,169,16
      0,32,240,253,198,6,16,221,96,165,8
      10,10,10,170,169,7,133,7,164
      DATA 36,24,165,40,133,38,165,41,105
      24,133,39,24,165,39,105,4,133,39,1
      89,126,8,9,128,37,9,36,50,48,2,73,1
      27,36,0,240,2,81,38,145,38
2320 DATA 252,198,2,7,167,223,96,127,127,12
      7,127,127,127,127,6,30,0,8,20,3
      6,6,0,99,54,28,8,28,54,99,0,8,20,3
      4,34,62,34,34,0,30,34,0,28,34,2,2,3
2330 DATA 34,30,34,34,34,30,0,28,34,2,2,3
      4,28,0,30,34,34,34,34,34,30,0,62,2,2
      2,30,2,2,62,0,62,2,2,30,2,2,0,60,
      2
2340 DATA 2,2,50,34,60,0,34,34,34,62,34,
      34,34,0,28,8,8,8,8,28,0,32,32,32,
      32,32,34,28,0,34,18,10,6,10,18,34,0
      2,2
2350 DATA 2,2,2,2,62,0,34,54,42,42,34,34
      34,0,34,34,38,42,50,34,30,28,34,
      34,34,34,28,0,30,34,34,30,2,2,2,
      0,28,34
2360 DATA 34,34,42,18,44,0,30,34,34,30,1
      0,18,34,0,28,34,2,28,32,34,28,0,62,
      8,8,8,8,0,9999

```

MINE OVER MATTER

COMMODORE 64

```

100 REM *** MINE OVER MATTER ***
110 REM *** MINE OVER MATTER ***
120 REM *** MINE OVER MATTER ***
130 REM COPYRIGHT 1985
140 REM EMERALD VALLEY PUBLISHING CO.
150 REM BY WILLIAM K. BALTHROP
160 REM AND THE HCM STAFF
170 REM HOME COMPUTER MAGAZINE
180 REM VERSION 5.4.1
190 REM COMMODORE 64 BASIC
200 REM
210 FOR X=1 TO 4: READ U(X): NEXT X
220 FOR X=1 TO 4: READ W(X): NEXT X
230 PRINT "CONTROL BLU": POKE 53280,15: POKE
53281,15
240 PRINT "SHIFT CLR" + 12 CRSRDOWN + 11 CRS
RRIGHT "MINE OVER MATTER"
250 FOR T=832 TO 895: READ D: POKE T,D: NEXT T: S
P=53248: POKE 2040,13: POKESP+39,0
260 DIM MS$(10): FOR X=1 TO 10: READ MS$(X): NE
XT X
270 DIM PL$(4),MP(4,5,8),T(4,2),B(8),V(4
): CL=54272: X=1: SP=53248
280 HY=44: HK=16: DW$="HOME" + 12 CRSRDOWN +
TT
290 FOR T=49152 TO 49181: READ D: POKE T,D: NEX
T T
300 PRINT "6 CRSRDOWN" + "TAB(12)" + "PRESS ANY
KEY": GOSUB 2290
310 PRINT "SHIFT CLR" + 4 CRSRDOWN + "DO YOU
WANT TO LOAD A GAME? PRINT" YOU SAVE
D BEFORE? (Y/N)"
320 GOSUB 2290: IFF$="Y" THEN 2890
330 IFF$<"Y": GOTO 320
340 FOR Z=1 TO 8: STEP 2: B(Z)=INT(RND(1)*10)+
1: B(Z+1)=INT(RND(1)*28)+1: NEXT Z

```

```

350 PRINT "SHIFT CLR";1CRSRRIGHT"MINE
OVER MATTER":PRINT"5CRSRDOWN"NUMBE
R OF PLAYERS (1-4): "2";
360 GOSUB 2290:IFASC(F$)=13THENJ=2:GOTO3
80
370 J=F:IFJ<1ORJ>4THEN360
380 PRINT "SHIFT CRSLEFT"J:PRINT"4C
RSRDOWN":FORZ=1TOJ:PRINTCHR$(U(Z))
"CRSRDOWN"3CRSRRIGHT"PLAYER #":Z;
390 GOSUB 3390:PL$(Z)=S$:IFS$="THENPL$(
Z)="PLAYER #"+MID$(STR$(Z),2)
400 PL$(Z)=LEFT$(PL$(Z),10)
410 NEXTZ:PRINT"
420 PRINT "SHIFT CLR";1CRSRRIGHT"MINE
OVER MATTER":PRINT"CRSRDOWN" GAME
TIME (YEARS): "3":FORZ=1TO7
430 PRINT "CRSRDOWN"5CRSRRIGHT"Z"SHIFT
FT CRSRLEFT"Z"2:NEXTZ
440 GOSUB 2290:IFASC(F$)=13THENF=3:GOTO4
60
450 IFF<1ORF>7THEN420
460 E=F*20:PRINT"HOME"2CRSRDOWN"TAB(
20);E
470 PRINTDWS"8CRSRDOWN"8CRSRRIGHT"CUR
RENT YEAR: 1985";:PRINT"4SHIFT CRS
RLEFT";
480 XN=4:GOSUB 3290:IFASC(A$)=13THENR=19
84:GOTO500
490 R=VAL(A$)-1
500 FORP=1TO4:T(P,1)=2500000:T(P,2)=0:N
EXTP:PRINT"HOME"
510 FORRW=1TO10:POKE251,RW:POKE252,32:S
YS49152:NEXTRW
520 FORZ=1464TO1503:POKEZ,120:POKEZ+CL,

```

Continued


```

530 M=INT(RND(1)*2000)+3000:PS=1:XS=1
540 FORX=XSTOE:FORP=1STOJ:IFT(P,1)<=0TH
550 GOSUB600:FORRW=12TO24:POKE251,RW:PO
560 KE252,32:SYS49152:NEXTRW
570 PRINTDWS="3CRSRDOWNNEXT TURN- PRES
580 S RETURN":PRINT"CRSRDOWNEXIT GAME
590 - PRESS CTRL RVSONFUNCTION 8CTR
600 L RVSONOFF
610 GOSUB2290:IFASC(F$)=140THENGOSUB233
620 IFASC(F$)<>13THEN560
630 PS=1:NEXTP:GOSUB600:NEXTX:GOSUB2330
640 :RUN
650 FORRW=12TO24:POKE251,RW:POKE252,32:
660 SYS49152:NEXTRW
670 PRINTDWS=CHR$(U(P)):"CTRL RVSON"CT
680 RL RVSONOFF:PL$(P):TAB(20):"CTRL BL
690 U"YEAR:R+X
700 PRINTDWS="2CRSRDOWNCASH:"TAB(20):
710 INT(T(P,1))
720 PRINT"CRSRDOWN(1) SURVEY:PRINT(2)
730 PRODUCTION:PRINT(3) REPORT:PRINT
740 (4) NEXT TURN
750 GOSUB2290:IFF<1ORF>4THEN640
760 IFF=4THENRETURN
770 ONFGOSUB1900,1240,1040:GOTO600
780 REM *** END OF YEAR ROUTINE ***
790 FORRW=12TO24:POKE251,RW:POKE252,32:
800 SYS49152:NEXTRW
810 PRINTDWS="6CRSRRIGHT" * HAPPY NEW
820 YEAR: * :PRINTTAB(18):R+X+1
830 PRINT"CRSRDOWN5CRSRRIGHTNAME:
840 CASH:CRSRDOWN
850 FORP=1TOJ:FORZ=1TO5
860 H=MP(P,Z,7):I=MP(P,Z,6):GOSUB2470
870 IFMP(P,Z,5)=0THEN910
880 ON(MP(P,Z,5)):GOTO750,910,750,910,77
890 0,840,750,880,880,880
900 MP(P,Z,5)=MP(P,Z,5)+1:IFMP(P,Z,5)=8
910 THEN(T(P,1)=T(P,1)+MP(P,Z,2)):MP(P,Z,
920 2)=0
930 GOTO910
940 GOSUB2240:T(P,1)=T(P,1)+S*M-MP(P,Z,
950 3):MP(P,Z,8)=MP(P,Z,8)-S*.001
960 MP(P,Z,4)=MP(P,Z,4)+1:IFMP(P,Z,8)<2
970 THENMP(P,Z,5)=6
980 IFINT(MP(P,Z,1)*.00035)=SANDINT(RND
990 (4)*5)=1THENMP(P,Z,5)=10:GOTO810
1000 GOTO820
1010 MP(P,Z,3)=0:GOSUB3380:PRINT"CTRL R
1020 ED"SHIFT Z":GOTO810
1030 IFINT(RND(1)*(100-V(3)))=1THENMP(P,
1040 Z,5)=10:MP(P,Z,3)=0
1050 GOTO910
1060 IFMP(P,Z,4)=0THENMP(P,Z,4)=-3:MP(P
1070 ,Z,3)=0:GOTO810
1080 MP(P,Z,4)=MP(P,Z,4)+1:IFMP(P,Z,4)<0
1090 THEN580
1100 MP(P,Z,5)=10:POKE(1024+MP(P,Z,7)+40
1110 *MP(P,Z,6)),90
1120 POKE(1024+MP(P,Z,7)+40*MP(P,Z,6)+CL
1130 ),2:GOTO910
1140 POKE(1024+MP(P,Z,7)+40*MP(P,Z,6)),1
1150 31
1160 POKE(1024+MP(P,Z,7)+40*MP(P,Z,6)+CL
1170 ),W(P)
1180 FORN=1TO8:MP(P,Z,N)=0:NEXTN:T(P,2)=
1190 T(P,2)-1:NEXTZ
1200 IFT(P,1)<=0THENGOSUB2490:PRINTDPS="
1210 6CRSRRIGHT":PL$(P):"3CRSRRIGHTBA
1220 NKRUPTHOME":GOTO830
1230 GOTO970
1240 FORZ=1TO5:IFMP(P,Z,5)>0THENPOKE(102
1250 4+MP(P,Z,7)+40*MP(P,Z,6)),131:GOTO9
1260 50
1270 GOTO960
1280 POKE(1024+MP(P,Z,7)+40*MP(P,Z,6)+CL
1290 ),W(P)
1300 NEXTZ:GOTO980
1310 PRINTDPS="6CRSRRIGHT":CHR$(U(P))PL
1320 $(P):TAB(20):"CTRL BLU":INT(T(P,
1330 1))
1340 PS=1:NEXTP:M=M+INT(RND(1)*1552)-776
1350 :IFM<500THENM=INT(RND(1)*2000)+3000
1360 PRINTDWS="1CRSRDOWN9CRSRRIGHTTO
1370 CONTINUE - PRESS RETURNHOME"
1380 GOSUB2290:IF(ASC(F$)<>13)AND(ASC(F$
1390 )<>140)THEN1000
1400 IFASC(F$)=140 THEN GOSUB2330
1410 RETURN
1420 REM *** PRINT REPORTS ***
1430 IFT(P,2)=0THENRETURN
1440 FORRW=13TO24:POKE251,RW:POKE252,32:
1450 SYS49152:NEXTRW
1460 PRINTDWS="2CRSRDOWNCRSRRIGHTSELE
1470 CT ONE:FORZ=1TO5:IFMP(P,Z,5)=0THE
1480 N1080
1490 GOSUB2510:PRINTDZ$:Z:MS$(MP(P,Z,5))
1500 NEXTZ
1510 GOSUB2290:IFASC(F$)=140THENRETURN
1520 IFF<1ORF>5THEN1090
1530 IFMP(P,Z,5)=0THEN1090
1540 Z=F:H=MP(P,Z,7):I=MP(P,Z,6):GOSUB24
1550 70
1560 GOSUB2240:FORRW=14TO24:POKE251,RW:P
1570 OKE252,32:SYS49152:NEXTRW
1580 PRINTDWS="1CRSRDOWNMILL:"TAB(20):IN
1590 T(MP(P,Z,1))
1600 PRINT"PROD:"TAB(20):INT(MP(P,Z,2)):
1610 PRINT"BAR PROD:"TAB(20):INT(S)
1620 PRINT"GROSS:"TAB(20):INT(S*M)
1630 PRINT"NET:"TAB(20):INT(S*M-MP(
1640 P,Z,3))
1650 H=MP(P,Z,7):I=MP(P,Z,6):GOSUB2470:P
1660 RINT"CRSRDOWNQUALITY:"TAB(16):INT
1670 (V(1))
1680 PRINT"DEPTH:"TAB(16):INT(V(2)):PRIN
1690 T"ENVIRONMENT:"TAB(16):INT(V(3))
1700 PRINT"CRSRDOWN5CRSRRIGHTTO CONT
1710 INUE - PRESS RETURNHOME"
1720 GOSUB2290:IFASC(F$)<>13THENGOSUB229
1730 0
1740 RETURN
1750 REM ***** PRODUCTION ROUTINES
1760 IFT(P,2)=0THENRETURN
1770 FORZ=1TO5:IFMP(P,Z,5)>0ANDMP(P,Z,5)
1780 <7THEN1270
1790 NEXTZ:RETURN
1800 FORRW=13TO24:POKE251,RW:POKE252,32:
1810 SYS49152:NEXTRW:PRINTDWS="2CRSRDOWN
1820 SELECT ONE:
1830 PRINT"CRSRDOWN(1) SET PRODUCTION R
1840 ATE:PRINT(2) START RECLAMATION"
1850 PRINT(3) EXPAND MILL:PRINT(4) EXIT
1860 GOSUB2290:IFASC(F$)=140ORASC(F$)=52
1870 THENRETURN
1880 IFF<1ORF>3THEN1300
1890 ONFGOSUB1340,1500,1640
1900 GOTO1240
1910 FORRW=14TO24:POKE251,RW:POKE252,32:
1920 SYS49152:NEXTRW
1930 PRINTDWS="2CRSRDOWNWHICH MINE:":G
1940 OSUB2290:IFASC(F$)=13THENRETURN
1950 IFF<1ORF>5THEN1350
1960 PRINTF
1970 Z=F:IFMP(P,Z,5)<4ORMP(P,Z,5)>5THEN1
1980 340
1990 H=MP(P,Z,7):I=MP(P,Z,6):GOSUB2470
2000 PRINTDWS="14:POKE252,32:SYS49152:PRIN
2010 TDWS="2CRSRDOWNPRODUCTION FOR #":Z
2020 PRINT"CRSRDOWNCURR. PROD.:"TAB(20
2030 );INT(MP(P,Z,3))
2040 GOSUB2240:PRINT"BAR./YEAR:"TAB(20):
2050 INT(S)
2060 PRINT"VALUE/BAR.:"TAB(20):INT(M)
2070 PRINT"GROSS:"TAB(20):INT(S*M)
2080 PRINT"NET:"TAB(20):INT(S*M-MP(
2090 P,Z,3))
2100 CRSRLEFT"SHIFT INST":INT(S*M-MP(P
2110 ,Z,3))
2120 PRINT"MAX PROD.:"TAB(20):INT(MP(P,Z
2130 ,1)*.00035)
2140 PRINT"CRSRDOWNNEW PROD.:"TAB(20)
2150 :
2160 XN=7:GOSUB3300:IFASC(A$)=13THENRETU
2170 RN
2180 MP(P,Z,3)=VAL(A$):MP(P,Z,5)=5:RETUR
2190 N
2200 FORRW=13TO24:POKE251,RW:POKE252,32:
2210 SYS49152:NEXTRW
2220 PRINTDWS="2CRSRDOWNSTART RECLAMATI
2230 ON ON":PRINT"WHICH MINE:":
2240 GOSUB2290:IFASC(F$)=13THENRETURN
2250 IFF>5THENRETURN
2260 Z=F:IFMP(P,Z,5)<4ORMP(P,Z,5)>6THENR
2270 ETURN
2280 PRINTZ
2290 D=MP(P,Z,2)*.4
2300 IFT(P,1)-D<0THENPRINTDWS="6CRSRDOWN
2310 NOT ENOUGH MONEY!":GOSUB2320:RETUR
2320 N
2330 T(P,1)=T(P,1)-D
2340 MP(P,Z,5)=7:MP(P,Z,3)=0:MP(P,Z,4)=0
2350 PRINTDWS="6CRSRDOWNRECLAMATION STA
2360 RTED ON #":Z
2370 PRINT"REC. COST:"TAB(20):INT(D)
2380 POKE(1024+MP(P,Z,7)+40*MP(P,Z,6)),1
2390 46
2400 POKE(1024+MP(P,Z,7)+40*MP(P,Z,6)+CL
2410 ),W(P):GOSUB2320:RETURN
2420 FORRW=13TO24:POKE251,RW:POKE252,32:
2430 SYS49152:NEXTRW
2440 PRINTDWS="2CRSRDOWNEXPAND ON WHICH
2450 MINE?":
2460 GOSUB2290:IFASC(F$)=140ORASC(F$)=13
2470 THENRETURN
2480 IFF<1ORF>5THEN1660
2490 Z=F:IFMP(P,Z,5)<2ORMP(P,Z,5)>5THENR
2500 ETURN
2510 PRINTZ
2520 H=MP(P,Z,7):I=MP(P,Z,6):GOSUB2470
2530 PRINT"CRSRDOWNCASH:"TAB(20):INT(T(
2540 P,1))
2550 PRINT"MILL SIZE:"TAB(20):INT(MP(P,Z,
2560 1))
2570 PRINT"MAX PROD.:"TAB(20):INT(MP(P,Z,
2580 1)*.00035)
2590 PRINT"QUALITY:"TAB(16):INT(V(1)):PR
2600 INT"DEPTH:"TAB(16):INT(V(2))
2610 PRINT"ENVIRONMENT:"TAB(16):INT(V(3))
2620 PRINT"INCREASE CONST.:"$":

```

Continued


```

1770 XN=7:GOSUB3300:IFASC(AS)=13THENRETU
1780 N=VAL(AS):IFN=0THENRETURN
1790 GOSUB2270
1800 IFT(P,1)-N<0THENPRINTDWS"11CRSRD
OWN YOU CAN'T AFFORD THAT MUCH HOME
1810 IFT(P,1)-N<0THEN1830
1820 GOTO1840
1830 GOSUB2320:RETURN
1840 MP(P,Z,1)=MP(P,Z,1)+N:T(P,1)=T(P,1)
N-F:MP(P,Z,2)=MP(P,Z,2)+F
MP(P,Z,3)=0:MP(P,Z,5)=3
1850 PRINTDWS"12CRSRDOWN CTRL RVSON R
1860 RETURN CTRL RVSON OFF TO GO TO MENU H
OME
1870 GOSUB2290:IFASC(FS)=13THENRETURN
1880 GOTO1870
1890 REM ***** SURVEY ROUTINE *****
1900 IF T(P,2)=5THENRETURN
1910 FORRW=13TO24:POKE251,RW:POKE252,32:
SYS49152:NEXTRW
1920 PRINTDWS"2CRSRDOWN CASH:"TAB(20);I
NT(T(P,1))
1930 PRINT"CRSRDOWN COST:"PRINT"CRSRD
OWN QUALITY:"PRINT"DEPTH:"PRINT"E
NVIRONMENT:"
1940 PRINT"CRSRDOWN ES DX TO MOVE:"PRINT
"SPACE TO SELECT:"PRINT"RETURN TO K
EEP HOME"
1950 H=16:I=5:HX=144:HY=83:POKESP,HX:POK
ESP+1,HY
1960 POKESP+21,PEEK(SP+21)OR1
1970 GOSUB2290:C=PEEK(1024+H+40*I)
1980 IFFS<>"S"ANDFS<>"E"ANDFS<>"D"ANDFS<
V"X"THEN2000
1990 IFC=19THENPOKE(1024+H+40*I),88:POKE
(1024+H+40*I+CL),6
2000 F=ASC(FS):IFF=69ANDI=>1THENI=I-1:HY
=HY-8:GOTO2070
2010 IFF=88ANDI=<9THENI=I+1:HY=HY+8:GOTO
2070
2020 IFF=83ANDH>0THENH=H-1:HX=HX-8:GOTO2
070
2030 IFF=68ANDH<39THENH=H+1:HX=HX+8:GOTO
2070
2040 IFF=32AND(C=32ORC=88)THEN2090
2050 IFF=13THEN2180
2060 GOTO1970
2070 HR=INT(HX/256):LX=HX-(256*HR)
2080 POKESP,LX:POKE53264,HR:POKESP+1,HY:
GOTO1970
2090 GOSUB2210:IFC=32THEND=V(2)+115+1000
2100 D=1000
2110 IFT(P,1)-D<0THENPRINTDWS"CRSRDOWN
NOT ENOUGH MONEY!!":GOSUB2320:RETUR
N
2120 T(P,1)=T(P,1)-D::PRINTDWS"2CRSRDOW
N CASH:"TAB(20)INT(T(P,1))
2130 PRINT"CRSRDOWN COST:"TAB(20);
6SHIFT CRSRLEFT:INT(D)
2140 PRINT"CRSRDOWN QUALITY:"TAB(20);
3SHIFT CRSRLEFT:INT(V(1))
2150 PRINT"DEPTH:"TAB(20);
4SHIFT
CRSRLEFT:INT(V(2))
2160 PRINT"ENVIRONMENT:"TAB(20);
3S
HIFT CRSRLEFT:INT(V(3)):POKE(1024
+H+40*I),19
2170 POKE(1024+H+40*I+CL),6:GOTO1970
2180 POKESP,PEEK(53264)AND254:POKE532
69,PEEK(53269)AND254:IFC<>19THENRET
URN
2190 T(P,2)=T(P,2)+1
2200 FORZ=1TO5:IFMP(P,Z,5)=0THEN2220
2210 NEXTZ:RETURN
2220 MP(P,Z,5)=1:POKE(1024+H+40*I),Z+176
2230 MP(P,Z,6)=1:MP(P,Z,7)=H:MP(P,Z,8)=V
(4):RETURN
2240 S=(MP(P,Z,3)/(V(2)+(V(3)*.5)+1)):S=
S*((MP(P,Z,8)*.5)+V(1))*0003
2250 IFS<(MP(P,Z,1))*00035)THENS=INT(S):
RETURN
2260 S=INT(MP(P,Z,1)*.00035):RETURN
2270 F=INT((V(3)*.0133)*(MP(P,Z,1)+N))+5
0000
2280 PRINTDWS"11CRSRDOWN RECLAMATION:";
INT(F):RETURN
2290 REM ***** GET *****
2300 GETFS:IFFS=" "THEN2300
2310 F=VAL(FS):RETURN
2320 FORN=1TO800:NEXTN:RETURN
2330 REM ***** QUIT AND SAVE *****
2340 FORRW=12TO24:POKE251,RW:POKE252,32:
SYS49152:NEXTRW
2350 PRINTDWS"2CRSRDOWN 5CRSRRIGHT WARE
YOU SURE? (Y/N)"
2360 GOSUB2290:IFASC(FS)=78THENRETURN
2370 IFASC(FS)<>89THEN2330
2380 FORRW=12TO24:POKE251,RW:POKE252,32:
SYS49152:NEXTRW:PRINT"HOME"13CRSR
DOWN
2390 PRINT"DO YOU WANT TO SAVE THE G
AME:"PRINT"IN PROGRESS? (Y/N)"
2400 GOSUB2290:IFFS="Y"THEN2640
2410 IFFS<>"N"THEN2400

```

```

2420 PRINT"3CRSRDOWN WOULD YOU LIKE TO
PLAY AGAIN (Y/N)?"
2430 GOSUB2290:IFFS="Y"THENRUN
2440 IFFS<>"N"THEN2430
2450 END
2460 REM GET V() ARRAY
2470 FORZ=1TO4:W=B(Z1*2-1)-I:B=B(Z1*2)-
H
2480 V(Z1)=INT(SIN(Z1*.4*SQR(W12+B12))*.5
0+50)*(-(Z1=2)*2+1):NEXTZ1:RETURN
2490 REM ***** DOWN FOR PLAYER *****
2500 DP=DWS"3CRSRDOWN:FORXP=1TOP:DP
$=DP$+"CRSRDOWN":NEXTXP:RETURN
2510 REM ***** DOWN *****
2520 DZ=DWS"2CRSRDOWN:FORXZ=1TOZ:DZ
$=DZ$+"CRSRDOWN":NEXTXZ:RETURN
2530 DATA151,156,154,28
2540 DATA11,4,14,2
2550 DATA000,000,000,000,000,000,000,000,000,000
000,000,000,000,000,000,000,000,000,000
2560 DATA000,129,000,000,000,000,000,000,195,000
000,000,000,000,000,000,000,000,000,000
2570 DATA000,000,000,000,000,000,000,000,000,000
000,129,000,000,195,000,000,000,000,000,000
2580 DATA000,000,000,000,000,000,000,000,000,000
000,000,000,000,000,000,000,000,000,000
2590 DATA SURVEY IN PROGRESS, SURVEY COM
PLETE, UNDER CONSTRUCTION
2600 DATA CONSTRUCTION COMPLETE, IN PROD
UCTION, OUT OF ORE, RECLAMATION STA
RTED
2610 DATA RECLAMATION DONE, MINE CLOSED,
CLOSED FOR CONTAMINATION
2620 DATA169,19,32,210,255,160,0,169,17,
32,210,255,200,196,251,208,248,165,
252
2630 DATA160,0,32,210,255,200,192,39,208
,248,96
2640 FORRW=12TO24:POKE251,RW:POKE252,32:
SYS49152:NEXTRW:PRINT"HOME"13CRSR
DOWN
2650 PRINT"WHAT NAME FOR THE FILE":GOSUB
3390:IFFS=" "THEN2380
2660 FORRW=12TO24:POKE251,RW:POKE252,32:
SYS49152:NEXTRW:PRINT"HOME"13CRSR
DOWN
2670 PRINT"SAVE TO TAPE OR DISK (T/D)"
2680 GOSUB2290:IFFS="T"THEN2700
2690 IFFS<>"D"THEN2680
2700 GOSUB3120:IFERTHEN2380
2710 PRINT#9,E:PRINT#9,J:PRINT#9,M:PRINT
#9,P:PRINT#9,R:PRINT#9,X
2720 FORSA=1TOJ
2730 PRINT#9,PLS(SA)
2740 NEXTSA
2750 FORSA=1TOJ:FORSE=1TO2
2760 PRINT#9,T(SA,SE)
2770 NEXTSB:NEXTSA
2780 FORSA=1TO8:PRINT#9,B(SA):NEXTSA
2790 FORSA=1TOJ:FORSE=1TOT(SA,2):FORSC=1
TO8
2800 PRINT#9,MP(SA,SB,SC)
2810 NEXTSC:NEXTSB:NEXTSA
2820 FORSA=1TO439:LL=PEEK(1024+SA):LL=PEE
K(55296+SA):IFLL<>88THEN2860
2830 PRINT#9,SA
2840 PRINT#9,L
2850 PRINT#9,LL
2860 NEXTSA
2870 PRINT#9,-1
2880 CLOSE9:CLOSE9:PRINT"SHIFT CLR S
EE YOU NEXT TIME...":END
2890 PRINT"SHIFT CLR 3CRSRDOWN WHAT NA
ME WAS USED TO SAVE YOUR GAME?":GO
SUB3300
2900 IFFS=" "THENRUN
2910 PRINT"2CRSRDOWN DID YOU SAVE TO TA
PE OR DISK? (T/D)"
2920 GOSUB2290:IFFS="T"THEN3150
2930 IFFS<>"D"THEN2920
2940 GOSUB3150:IFERTHENRUN
2950 INPUT#9,E:INPUT#9,J:INPUT#9,M:INPUT
#9,P:INPUT#9,R:INPUT#9,X
2960 XS=X:PS=P
2970 FORSA=1TOJ
2980 INPUT#9,PLS(SA)
2990 NEXTSA
3000 FORSA=1TOJ:FORSE=1TO2
3010 INPUT#9,T(SA,SE):NEXTSB:NEXTSA
3020 FORSA=1TO8:INPUT#9,B(SA):NEXTSA
3030 FORSA=1TOJ:FORSE=1TOT(SA,2):FORSC=1
TO8
3040 INPUT#9,MP(SA,SB,SC)
3050 NEXTSC:NEXTSB:NEXTSA
3060 GOSUB3180
3070 FORSA=1TO439:INPUT#9,LL:IFLL=-1THEN
SA=439:GOTO3100
3080 INPUT#9,L:POKE1024+LL,L
3090 INPUT#9,L:POKE55296+LL,L
3100 NEXTSA
3110 CLOSE9:CLOSE15:GOTO540
3120 IFFS="T"GOTO3140
3130 OPEN9,8,9,"@","+S$+",S,W":GOSUB3500:
RETURN
3140 OPEN9,1,1,S$:RETURN
3150 IFFS="T"GOTO3170
3160 OPEN9,8,9,"@","+S$+",S,R":GOSUB3500:
RETURN

```

Continued


```

3170 OPEN9,1,0,S$:GOTO2950
3180 FORRW=1TOTO10:POKE251,RW:POKE252,32:S
YS49152:NEXTRW
3190 FORZ=1464TOTO1503:POKEZ,120:POKEZ+CL
9:NEXTZ
3200 FORRW=12TOTO24:POKE251,RW:POKE252,32:
SYS49152:NEXTRW
3210 FORBP=1TOTOJ:FORBZ=1TOS
3220 QM=MP(BP,BZ,5):IFQM=0THEN3280
3230 QS=(1024+MP(BP,BZ,7)+40*MP(BP,BZ,6)
)
3240 IFQM=10THENPOKEQS,90:POKEQS+CL,2:GO
TO3280
3250 IFQM=9THENPOKEQS,131:POKEQS+CL,W(BP
):GOTO3280
3260 IFQM=8ORQM=7THENPOKEQS,146:POKEQS+C
L,W(BP):GOTO3280
3270 POKEQS,BZ+176:POKEQS+CL,W(BP)
3280 NEXTBZ:NEXTBP:RETURN
3290 REM INPUT FOR NUMERIC DATA
3300 AS="":FORXZ=1TOTOXN
3310 GETFS:IFFS="1"THEN3340
3320 IF ASC(FS)=13THENXZ=XN:AS=AS+CHR$(1
3):GOTO3360
3330 IFASC(FS)=20ANDXZ>1THENPRINT"SHIFT
CRSRLEFT":"SHIFT CRSRLEFT":AS=LE
FTS(AS,LEN(AS)-1):XZ=XZ-1:GOTO3310

```

```

33340 IFASC(F$)<48ORASC(F$)>57THEN3310
33350 AS=AS+F$:PRINTF$;
33360 NEXTXZ:RETURN
33370 REM LOCATE CURSOR ROUTINE
33380 POKE781,MP(P,Z,6):POKE782,MP(P,Z,7)
33390 :POKE783,0:SYS65520:RETURN
33400 REM INPUT FILE NAME
33410 POKE198,0:L=0:S$=""
33420 PRINT"CTRL RVSONE CTRL RVSOFF"SH
33430 LEFT CRSRLEFT"::GOSUB3480:IFK=13THE
33440 NPRINT:RETURN
33450 IFK=20ANDL>0THENS$=LEFT$(S$,LEN(S$)-
33460 1):PRINT"2SHIFT CRSRLEFT"::L=L-
33470 1:GOTO3410
33480 IF K<32ORK>90THEN3410
33490 IFK=34ORK=36ORK=42ORK=44ORK=58=6
33500 3ORK=64THEN3410
33510 IFL=16THEN3410
33520 PRINTK$::S$=S$+K$:L=L+1:GOTO3410
33530 RETURN
33540 GETK$:IFK$=""THEN3480
33550 K=ASC(K$):RETURN
33560 REM READ DISK ERROR
33570 OPEN15,8,15:INPUT#15,ER,ES:IFER<>0T
33580 HEN PRINT" "ES" "":CLOSE9:CLOSE1
33590 5
33600 RETURN

```

MINE OVER MATTER

IBM PC & IBM PCjr

```

1100  ** MINE OVER MATTER **
1110  ** ** ** **
1120  ** ** ** *
1130  COPYRIGTH 1985 PUBLISHING CO.
1140  EMERALD VALLEY K. BALTRAP
1150  BY WILLIAM COMPUTER MAGAZINE
1160  HOME COM 5.4.1
1170  VERS ION 2.1
1180  DOS 2.1
1190  IBM PC r w CARTRIDGE BASIC or
1200  IBM PC r w BASICA and
1210  COLOR/GRAPHICS ADAPTER and
1220  COLOR/MONITOR
1230
1240  SCREEN: 0: WIDTH 40: KEY OFF: RANDOMIZE
1250  DIM PL$(4), MP$(4,6,12), T(4,2), B(4,2)
1260  F1$=""
1270  F2$=""
1280  F3$=""
1290  RESTORE NEXT: 1950: FOR Z=1 TO 4: READ PCOL
1300  (Z): NEXT Z: FOR Z=1 TO 10: READ STATS(Z)
1310  CLS: LOCATE 1,1: PRINT "SELECT ONE": P
1320  RINT: PRINT "2) LOAD AN OLD GAME": P
1330  GOSUB 1860: IF A$=<"3" THEN 320
1340  CLS: LOCATE 1,1: PRINT "MINE OVER MA
1350  TTER (1 TO 4): 2: GOSUB 1860: IF (A$<"1"
1360  OR A$>"4") THEN 3
1370  LOCATE 3,28: PRINT A$: J=VAL(A$): FOR
1380  YEAR=1 TO 7: IF (J=1) THEN 1780: IF (J=2)
1390  THEN 1780: IF (J=3) THEN 1780: IF (J=4)
1400  THEN 1780: IF (J=5) THEN 1780: IF (J=6)
1410  THEN 1780: IF (J=7) THEN 1780: IF (J=8)
1420  THEN 1780: IF (J=9) THEN 1780: IF (J=10)
1430  THEN 1780: IF (J=11) THEN 1780: IF (J=12)
1440  THEN 1780: IF (J=13) THEN 1780: IF (J=14)
1450  THEN 1780: IF (J=15) THEN 1780: IF (J=16)
1460  THEN 1780: IF (J=17) THEN 1780: IF (J=18)
1470  THEN 1780: IF (J=19) THEN 1780: IF (J=20)
1480  THEN 1780: IF (J=21) THEN 1780: IF (J=22)
1490  THEN 1780: IF (J=23) THEN 1780: IF (J=24)
1500  THEN 1780: IF (J=25) THEN 1780: IF (J=26)
1510  THEN 1780: IF (J=27) THEN 1780: IF (J=28)
1520  THEN 1780: IF (J=29) THEN 1780: IF (J=30)
1530  THEN 1780: IF (J=31) THEN 1780: IF (J=32)
1540  THEN 1780: IF (J=33) THEN 1780: IF (J=34)
1550  THEN 1780: IF (J=35) THEN 1780: IF (J=36)
1560  THEN 1780: IF (J=37) THEN 1780: IF (J=38)
1570  THEN 1780: IF (J=39) THEN 1780: IF (J=40)
1580  THEN 1780: IF (J=41) THEN 1780: IF (J=42)
1590  THEN 1780: IF (J=43) THEN 1780: IF (J=44)
1600  THEN 1780: IF (J=45) THEN 1780: IF (J=46)
1610  THEN 1780: IF (J=47) THEN 1780: IF (J=48)
1620  THEN 1780: IF (J=49) THEN 1780: IF (J=50)
1630  THEN 1780: IF (J=51) THEN 1780: IF (J=52)
1640  THEN 1780: IF (J=53) THEN 1780: IF (J=54)
1650  THEN 1780: IF (J=55) THEN 1780: IF (J=56)
1660  THEN 1780: IF (J=57) THEN 1780: IF (J=58)
1670  THEN 1780: IF (J=59) THEN 1780: IF (J=60)
1680  THEN 1780: IF (J=61) THEN 1780: IF (J=62)
1690  THEN 1780: IF (J=63) THEN 1780: IF (J=64)
1700  THEN 1780: IF (J=65) THEN 1780: IF (J=66)
1710  THEN 1780: IF (J=67) THEN 1780: IF (J=68)
1720  THEN 1780: IF (J=69) THEN 1780: IF (J=70)
1730  THEN 1780: IF (J=71) THEN 1780: IF (J=72)
1740  THEN 1780: IF (J=73) THEN 1780: IF (J=74)
1750  THEN 1780: IF (J=75) THEN 1780: IF (J=76)
1760  THEN 1780: IF (J=77) THEN 1780: IF (J=78)
1770  THEN 1780: IF (J=79) THEN 1780: IF (J=80)
1780  THEN 1780: IF (J=81) THEN 1780: IF (J=82)
1790  THEN 1780: IF (J=83) THEN 1780: IF (J=84)
1800  THEN 1780: IF (J=85) THEN 1780: IF (J=86)
1810  THEN 1780: IF (J=87) THEN 1780: IF (J=88)
1820  THEN 1780: IF (J=89) THEN 1780: IF (J=90)
1830  THEN 1780: IF (J=91) THEN 1780: IF (J=92)
1840  THEN 1780: IF (J=93) THEN 1780: IF (J=94)
1850  THEN 1780: IF (J=95) THEN 1780: IF (J=96)
1860  THEN 1780: IF (J=97) THEN 1780: IF (J=98)
1870  THEN 1780: IF (J=99) THEN 1780: IF (J=100)
1880  THEN 1780: IF (J=101) THEN 1780: IF (J=102)
1890  THEN 1780: IF (J=103) THEN 1780: IF (J=104)
1900  THEN 1780: IF (J=105) THEN 1780: IF (J=106)
1910  THEN 1780: IF (J=107) THEN 1780: IF (J=108)
1920  THEN 1780: IF (J=109) THEN 1780: IF (J=110)
1930  THEN 1780: IF (J=111) THEN 1780: IF (J=112)
1940  THEN 1780: IF (J=113) THEN 1780: IF (J=114)
1950  THEN 1780: IF (J=115) THEN 1780: IF (J=116)
1960  THEN 1780: IF (J=117) THEN 1780: IF (J=118)
1970  THEN 1780: IF (J=119) THEN 1780: IF (J=120)
1980  THEN 1780: IF (J=121) THEN 1780: IF (J=122)
1990  THEN 1780: IF (J=123) THEN 1780: IF (J=124)
2000  THEN 1780: IF (J=125) THEN 1780: IF (J=126)
2010  THEN 1780: IF (J=127) THEN 1780: IF (J=128)
2020  THEN 1780: IF (J=129) THEN 1780: IF (J=130)
2030  THEN 1780: IF (J=131) THEN 1780: IF (J=132)
2040  THEN 1780: IF (J=133) THEN 1780: IF (J=134)
2050  THEN 1780: IF (J=135) THEN 1780: IF (J=136)
2060  THEN 1780: IF (J=137) THEN 1780: IF (J=138)
2070  THEN 1780: IF (J=139) THEN 1780: IF (J=140)
2080  THEN 1780: IF (J=141) THEN 1780: IF (J=142)
2090  THEN 1780: IF (J=143) THEN 1780: IF (J=144)
2100  THEN 1780: IF (J=145) THEN 1780: IF (J=146)
2110  THEN 1780: IF (J=147) THEN 1780: IF (J=148)
2120  THEN 1780: IF (J=149) THEN 1780: IF (J=150)
2130  THEN 1780: IF (J=151) THEN 1780: IF (J=152)
2140  THEN 1780: IF (J=153) THEN 1780: IF (J=154)
2150  THEN 1780: IF (J=155) THEN 1780: IF (J=156)
2160  THEN 1780: IF (J=157) THEN 1780: IF (J=158)
2170  THEN 1780: IF (J=159) THEN 1780: IF (J=160)
2180  THEN 1780: IF (J=161) THEN 1780: IF (J=162)
2190  THEN 1780: IF (J=163) THEN 1780: IF (J=164)
2200  THEN 1780: IF (J=165) THEN 1780: IF (J=166)
2210  THEN 1780: IF (J=167) THEN 1780: IF (J=168)
2220  THEN 1780: IF (J=169) THEN 1780: IF (J=170)
2230  THEN 1780: IF (J=171) THEN 1780: IF (J=172)
2240  THEN 1780: IF (J=173) THEN 1780: IF (J=174)
2250  THEN 1780: IF (J=175) THEN 1780: IF (J=176)
2260  THEN 1780: IF (J=177) THEN 1780: IF (J=178)
2270  THEN 1780: IF (J=179) THEN 1780: IF (J=180)
2280  THEN 1780: IF (J=181) THEN 1780: IF (J=182)
2290  THEN 1780: IF (J=183) THEN 1780: IF (J=184)
2300  THEN 1780: IF (J=185) THEN 1780: IF (J=186)
2310  THEN 1780: IF (J=187) THEN 1780: IF (J=188)
2320  THEN 1780: IF (J=189) THEN 1780: IF (J=190)
2330  THEN 1780: IF (J=191) THEN 1780: IF (J=192)
2340  THEN 1780: IF (J=193) THEN 1780: IF (J=194)
2350  THEN 1780: IF (J=195) THEN 1780: IF (J=196)
2360  THEN 1780: IF (J=197) THEN 1780: IF (J=198)
2370  THEN 1780: IF (J=199) THEN 1780: IF (J=200)
2380  THEN 1780: IF (J=201) THEN 1780: IF (J=202)
2390  THEN 1780: IF (J=203) THEN 1780: IF (J=204)
2400  THEN 1780: IF (J=205) THEN 1780: IF (J=206)
2410  THEN 1780: IF (J
```

```

440 NEXT:GOSUB 1410:NEXT:GOSUB 1590:CLE
450 AR:GOTO 240
460 ' MAIN MENU SCREEN -- SCREEN #0
470
480 SCREEN 0,0,0,SC:SC=0:FOR Z=20 TO 25:
:LOCATE Z,1,0:PRINT STRINGS(39,32);
:NEXT
490 COLOR 4:LOCATE 19,1:PRINT STRINGS(4
0,205)::COLOR 0:PRINT TAB(12);"MINE
OVER MATTER":COLOR PCOL(P):PRINT "
PLAYER:":PL$(P)::COLOR 0:PRINT TAB
(27);"YEAR:":YEAR:PRINT (1) SURVEY
:TAB(20);"3) REPORTS":PRINT (2) PRO
DUCTION":TAB(20);
500 PRINT (4) NEXT TURN::SCREEN (5),0
510 GOSUB 1860:IF (AS<"1" OR AS>"5") AND
D AS<>CHRS(27) THEN 510 ELSE IF AS=
CHRS(27) OR AS="4" THEN RETURN ELSE
ON VAL(AS) GOSUB 550,780,1150:GOTO
480
520 ' SURVEY -- SCREEN #0
530
540
550 IF T(P,2)=5 THEN RETURN ELSE SC=0:F
OR Z=20 TO 25:LOCATE Z,1:PRINT STRI
NG$(39,32)::NEXT
560 LOCATE 20,1:PRINT USING F1$;"AVAILA
BLE CASH:":T(P,1):PRINT "COST OF SU
RVEY:":PRINT "QUALITY:":PRINT "DEPT
H:":PRINT "ENVIRONMENT:":LOCATE 25
,1:COLOR PCOL(P):PRINT "PLAYER:":P
L$(P):CURX=20:CURY=10:LOCATE CURY,
CURX,1,0,7
570 AS=INKEY$:IF AS="" THEN 570 ELSE IF
LEN(AS)=2 THEN AS=RIGHT$(AS,1):GOS
UB 590 ELSE IF AS=" " THEN GOSUB 65
0 ELSE IF AS=CHRS(13) OR AS=CHRS(27
) THEN 700
580 GOTO 570
590 DIR=INSTR("HKMP",AS):IF DIR=0 THEN
RETURN ELSE CHAR=SCREEN(CURY,CURX):
IF CHAR=83 THEN LOCATE CURY,CURX,0:
COLOR 0:PRINT CHRS(157);
600 ON DIR GOSUB 610,620,630,640:RETURN
610 IF CURY>1 THEN CURY=CURY-1:LOCATE C
URY,CURX,1:RETURN ELSE RETURN
620 IF CURX>1 THEN CURX=CURX-1:LOCATE C
URY,CURX,1:RETURN ELSE RETURN
630 IF CURX<40 THEN CURX=CURX+1:LOCATE
CURY,CURX,1:RETURN ELSE RETURN
640 IF CURY<18 THEN CURY=CURY+1:LOCATE
CURY,CURX,1:RETURN ELSE RETURN
650 CHAR=SCREEN(CURY,CURX):CCOL=SCREEN(
CURY,CURX,1) MOD 16:IF CCOL<>0 OR T
(P,2)=5 THEN RETURN
660 Z=6:GOSUB 1900:IF CHAR>32 THEN COST
=1000 ELSE COST=MP(P,Z,10)*115+1000
670 IF T(P,1)-COST<0 THEN LOCATE 25,12:
PRINT "COST TOO MUCH!":FOR TD=1 TO
1000:NEXT:LOCATE 25,12:PRINT STRIN
GS(15,32):RETURN
680 T(P,1)=T(P,1)-COST:LOCATE 20,1:COLO
R 0:PRINT USING F1$;"AVAILABLE CASH:
":T(P,1):PRINT USING F1$;"COST OF
SURVEY:":COST:PRINT USING F2$;"QUAL
ITY:":MP(P,6,9):PRINT USING F2$;"DE
PTH:":MP(P,6,10):PRINT USING F2$;"E
NVIRONMENT:":MP(P,6,11);
690 LOCATE CURY,CURX:COLOR PCOL(P):PRIN
T "S":LOCATE CURY,CURX:COLOR 0:RET
URN
700 CHAR=SCREEN(CURY,CURX):IF CHAR<>83
THEN RETURN

```

Continued


```

710 FOR Z=1 TO 5:IF MP(P,Z,5)=0 THEN 73
720 NEXT Z:RETURN
730 MP(P,Z,5)=1:T(P,2)=T(P,2)+1:MP(P,Z,
6)MP=P:CURY=MP(P,Z,7):CURX=MP(P,Z,8):MP
(P,6,12):MP(P,Z,9)=MP(P,6,9):MP(P,Z
,10)=MP(P,6,10):MP(P,Z,11)=MP(P,6,1
1):MP(P,Z,12)=MP(P,6,12):LOCATE CUR
Y,CURX:COLOR PCOL(P):PRINT CHR$(48+
Z):;
740 COLOR 0:LOCATE CURY,CURX:RETURN
750
760 ' PRODUCTION SCREEN -- SCREEN #1
770
780 IF T(P,2)=0 THEN RETURN ELSE FOR Z=
1 TO 5:IF MP(P,Z,5)<2 OR MP(P,Z,5)>
6 THEN NEXT Z:RETURN
790 SCREEN 0,0,1,0:SC=1
800 CLS:LOCATE 1,12:PRINT "MINE OVER MA
TTER":PRINT COLOR PCOL(P):PRINT "PL
AYER":PRINT PL$(P):COLOR 0:PRINT TAB(27
):"YEAR":YEAR:PRINT "PRODUCT
ION OPTIONS":PRINT "1) SET P
RODUCTION RATE":PRINT "2) START REC
LAMATION":PRINT "3) CONSTRUCTION"
810 PRINT "4) RETURN TO MAIN MENU":SCRE
EN 1,1
820 GOSUB 1860:IF (A$<"1" OR A$>"4") AN
D A$<>CHR$(27) THEN 820 ELSE IF A$=
CHR$(27) OR A$="4" THEN RETURN ELSE
ON VAL(A$) GOSUB 860,960,1040:GOTO
800
830
840 ' PRODUCTION -- SET PROD. RATE
850
860 DLN=12:GOSUB 1880
870 GOSUB 1860:IF (A$<"1" OR A$>"5") AN
D A$<>CHR$(27) THEN 870 ELSE IF A$=
CHR$(27) THEN RETURN ELSE Z=VAL(A$)
:IF MP(P,Z,5)<4 OR MP(P,Z,5)>5 THEN
LOCATE 20,1:PRINT A$:" IS NOT CAPA
BLE OF PRODUCTION!!!":GOSUB 1870:RE
TURN
880 CLS:GOSUB 1910
890 LOCATE 1,10:PRINT "SET PRODUCTION R
ATE":PRINT "PLAYER":PL$(P):TAB(30)
):"YEAR":R+X:PRINT "MINE #
Z:PRINT "PRINT USING F1$:"MILL VALUE
":MP(P,Z,1):PRINT USING F1$:"REC
BOND":MP(P,Z,2):PRINT USING F1$:"C
URRENT PROD":MP(P,Z,3)
900 PRINT USING F2$:"CURR. BARR/YEAR:
S:PRINT USING F2$:"MAX BARR/YEAR:
MP(P,Z,1):00035:PRINT USING F1$:"V
ALUE/BARR:":M:PRINT USING F1$:"GROS
S PROFIT:":S*M-MP(P,Z,3):PRINT USING
F2$:"QUALITY":MP(P,Z,9)
910 PRINT USING F2$:"DEPTH:":MP(P,Z,10)
:PRINT USING F2$:"ENVIRONMENT:":MP(
P,Z,11):PRINT "ENTER NOTHING
FOR NO CHANGE":PRINT "OR NEW PRODU
CTION RATE:":SPACES(12):IX=24:IY=2
0:IL=12:GOSUB 1700:IF A$=" " THEN RE
TURN ELSE MP(P,Z,3)=VAL(A$)
920 MP(P,Z,5)=5:GOSUB 1910:GOTO 890
930
940 ' PRODUCTION -- START RECLAMATION
950
960 DLN=12:GOSUB 1880
970 GOSUB 1860:IF (A$<"1" OR A$>"5") AN
D A$<>CHR$(27) THEN 970 ELSE IF A$=
CHR$(27) THEN RETURN ELSE Z=VAL(A$)
:IF MP(P,Z,5)<4 OR MP(P,Z,5)>6 THEN
LOCATE 20,1:PRINT A$:" CAN'T BE RE
CLAIMED NOW!":GOSUB 1870:RETURN
980 COST=MP(P,Z,2)*.4:IF T(P,1)-COST<0
THEN LOCATE 20,1:PRINT "YOU DON'T H
AVE ENOUGH MONEY!!!":GOSUB 1870:RETU
RN
990 T(P,1)=T(P,1)-COST:MP(P,Z,5)=7:MP(P
,Z,3)=0:MP(P,Z,4)=0:LOCATE 20,1:PRI
NT "RECLAMATION STARTED ON #":Z:PRI
NT USING F1$:"RECLAMATION COST:":CO
ST:LOCATE MP(P,Z,6),MP(P,Z,7),0:COL
OR PCOL(P):SCREEN 0,0,0,1:PRINT
:SCREEN 0,0,1,1:COLOR 1:GOSUB 1870
1000 RETURN
1010
1020 ' PRODUCTION -- CONSTRUCTION
1030
1040 DLN=12:GOSUB 1880
1050 GOSUB 1860:IF (A$<"1" OR A$>"5") AN
D A$<>CHR$(27) THEN 1050 ELSE IF A$=
CHR$(27) THEN RETURN ELSE Z=VAL(A$)
:IF MP(P,Z,5)<2 OR MP(P,Z,5)>5 THE
N RETURN
1060 CLS:LOCATE 1,14:PRINT "CONSTRUCTION
":PRINT "PLAYER":PL$(P):TAB(30)
):"YEAR":YEAR:PRINT "PRINT FOR MINE #
Z
1070 LOCATE 6,1:PRINT USING F1$:"AVAILAB
LE CASH:":T(P,1):PRINT USING F1$:"C
URRENT MILL SIZE:":MP(P,Z,1)

```

[illegible]

Continued


```

1420 COLOR = PCOL(P)+16:PRINT USING F3.5;PL
      S(P),**BANKRUPT**FOR Z=1 TO 5:
      IF MP(P,Z,5)>0 THEN LOCATE MP(P,Z,7)
      MP(P,Z,5):COLOR 0:SCREEN 0,0,1,1
      PRINT P,C,L:;SCREEN 0,0,1,1
1430 NEXT
1440 NEXT:FOR P=1 TO J:FOR Z=1 TO 5:CURX
      =MP(P,Z,7):CURY=MP(P,Z,6):IF MP(P,Z
      .5)=0 THEN 1540 ELSE GOSUB 1910
1450 ON MP(P,Z,5) GOTO 1460,1540,1460,15
      40,1470,1500,1460,1520,1520,1520
1460 MP(P,Z,5)=MP(P,Z,5)+1:IF MP(P,Z,5)=
      8 THEN T(P,1)=T(P,1)+MP(P,Z,2):MP(P
      .Z,2)=0:GOTO 1540 ELSE 1540
1470 T(P,1)=T(P,1)+S*M-MP(P,Z,3):MP(P,Z
      .5)=MP(P,Z,8)-S*.001:MP(P,Z,4)=MP(P
      .Z,4)+1:IF MP(P,Z,8)<2 THEN MP(P,Z,5
      )=6
1480 IF INT(MP(P,Z,1)*.00035)=S AND INT(
      RND*.5)=1 THEN 1510
1490 GOTO 1540
1500 IF MP(P,Z,4)>0 THEN MP(P,Z,4)=-4:M
      P(P,Z,3)=0:GOTO 1540 ELSE MP(P,Z,4)
      =MP(P,Z,4)+1:IF MP(P,Z,4)<0 THEN 15
      40
1510 MP(P,Z,5)=10:MP(P,Z,3)=0:SCREEN 0
      .1:LOCATE CURY,CURX:COLOR 4:PRINT C
      HRS(4):;SCREEN 1,1:COLOR 0:GOTO 1
      540
1520 SCREEN 0,1:LOCATE CURY,CURX:IF MP
      (P,Z,5)=8 THEN COLOR PCOL(P):PRINT
      C,L:;ELSE COLOR 4:PRINT CHR$(4):
1530 SCREEN 1,1:FOR Y=1 TO 8:MP(P,Z,Y)
      =0:NEXT:T(P,2)=T(P,2)-1:GOTO 1540
1540 NEXT:NEXT:M=M+INT(RND*.1552)-776:IF
      M<500 THEN M=INT(RND*2000)+3000
1550 COLOR 0:GOSUB 1870:RETURN
1560
1570
1580
1590
1600 SCREEN 0,0,1,1:CLS:LOCATE 3,1:PRINT
      "WOULD YOU LIKE TO SAVE THIS GAME
      (Y/N)?"
1610 GOSUB 1860:IF AS="Y" OR AS="y" THEN
      GOSUB 1330 ELSE IF AS<>"N" AND AS<
      "n" THEN 1600
1620 CLS:LOCATE 3,1:PRINT "ARE YOU SURE
      YOU WANT TO EXIT (Y/N)?"
1630 GOSUB 1860:IF AS="Y" OR AS="y" THEN
      1640 ELSE IF AS<>"N" AND AS<"n" T
      HEN 1620
1640 RETURN
1650 CLS:PRINT TAB(12):"MINE OVER MATTER
      "FOR Z=1 TO J:LOCATE Z*2+5,1:PRINT
      USING F1.5;PLS(Z),T(Z,1):NEXT
1660 LOCATE 22,1:PRINT "PLAY AGAIN (Y/N)
      "
1670 GOSUB 1860:IF AS="Y" OR AS="y" THEN
      RUN ELSE IF AS<>"N" AND AS<"n" TH
      EN 1660 ELSE END
1680
1690
1700
1710
1720
1730
1740
1750
1760
1770
1780
1790
1800
1810
1820
1830
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950

```

```

AS="":CP=1:LOCATE IY,IX,1
K$=INKEY$:IF (K$<"0" OR K$>"9") AND
K$<>"CHR$(8)" AND K$<>"CHR$(13)" AND K
$<>"CHR$(27)" THEN 1710 ELSE IF K$=CH
R$(13) OR K$=CHR$(27) THEN RETURN
IF K$=CHR$(8) THEN IF CP>1 THEN CP=
CP-1:AS=LEFT$(AS,CP):LOCATE IY,IX:P
RINT AS:;LOCATE IY,IX+CP,1:GOTO
1710 ELSE CP=1:AS="":LOCATE IY,IX:P
RINT STRING$(IL,32):;LOCATE IY,IX,1
:GOTO 1710
AS=AS+K$:PRINT K$:CP=CP+1:IF CP>IL
THEN CP=CP-1:LOCATE IY,IX+CP-1,1
:GOTO 1710
ALPHA INPUT ROUTINE
AS="":CP=1:LOCATE IY,IX,1
K$=INKEY$:IF (K$<"0" OR K$>"9") AND
K$<>"CHR$(8)" AND K$<>"CHR$(13)" AND K$<>"CH
R$(13)" AND K$<>"CHR$(27)" THEN 1790 E
LSE IF K$=CHR$(13) OR K$=CHR$(27) T
HEN RETURN
IF K$=CHR$(8) THEN IF CP>1 THEN CP=
CP-1:AS=LEFT$(AS,CP):LOCATE IY,IX:P
RINT AS:;LOCATE IY,IX+CP,1:GOTO
1790 ELSE CP=1:AS="":LOCATE IY,IX:P
RINT STRING$(IL,32):;LOCATE IY,IX,1
:GOTO 1790
AS=AS+K$:PRINT K$:CP=CP+1:IF CP>IL
THEN CP=CP-1:LOCATE IY,IX+CP-1,1
:GOTO 1790
UTILITY ROUTINES
AS="":WHILE AS="":AS=INKEY$:WEND:SO
UND 880.25:RETURN
LOCATE 25,10:PRINT "PRESS ";CHR$(17
):CHR$(217):;TO CONTINUE":GOSUB 1
860:RETURN
Y=0:FOR Z=1 TO 5:IF MP(P,Z,5)>0 THE
N LOCATE DLN+Y,1:PRINT STR$(Z):;
:STAT$(MP(P,Z,5)):Y=Y+1
NEXT:RETURN
FOR Y=1 TO 4:MP(P,Z,8+Y)=INT(SIN(Y*
.4*SQR((B(Y,1)-CURX)^2+(B(Y,2)-CURY
)^2))*50+50)*(-(Y=2)*2+1):NEXT:RETU
RN
S=(MP(P,Z,3)/(MP(P,Z,10)+(MP(P,Z,11
)*.5)+10)):S=S*((MP(P,Z,8)*.5)+MP(P
,Z,9))*0.003:IF S>MP(P,Z,1)*.00035
THEN S=MP(P,Z,1)*.00035:RETURN ELSE
RETURN
PROGRAM DATA
DATA 1,4,10,11,SURVEY IN PROGRESS,S
URVEY COMPLETE,UNDER CONSTRUCTION,C
ONSTRUCTION COMPLETE,IN PRODUCTION,
OUT OF ORE,RECLAMATION STARTED,RECL
AMATION DONE,MINE CLOSED,CLOSED FOR
CONTAMINATION

```

HCM

MINE OVER MATTER

TI-99/4A

```

1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
2000
2100
2200
2300
2400
2500
2600
2700
2800
2900

```

```

300
310
320
330
340
350
360
370
380
390
400
410
420
430
440
450

```

Continued

TYPE-IN LISTINGS

```

910 CALL T(14,A) : "WHICH MINE:" : ACCEPT AT(14,12) : SIZE(1) : VALIDATE(DIGIT) : $ : IF A$="" THEN RETURN ELSE Z=VAL(A$) : IF MP(P,Z,Q)<L OR MP(P,Z,Q)>Q THEN GOSUB 1480 : YEAR/DISPLAY AT(17,A) : USING 1470 : "VALUE/BAR:" : MDISP VV(BS,MP(P,Z,7),MP(P,Z,6),V()) : CALL HCHAR(14,A) : PRODUCING FOR : "Z:" : Z : DISP LAY,AT(16,A) : USING 1470 : "CURR. PROD MP(P,Z,K) : " : GOSUB 1400 : BAR : /YEAR/DISPLAY AT(17,A) : USING 1470 : "GROSS:" : S:M : "S*M-MP(P,Z,K) : " : NET : "S*M-MP(P,Z,K) : " : DISPLAY AT(21,A) : USING 1480 : "MAX PR OD:" : MP(P,Z,A) : .00035 : " : DISPLAY AT(23,A) : NEW,PROD# : " : MP(P,Z,K) : ACCEP T AT(23,12) : SIZE(-9) : VALIDATE(DIGIT) : $ : IF A$="" THEN MP(P,Z,K)=G ELSE MP(P,Z,K)=VAL(A$) : MP(P,Z,Q)=5 : RETURN : CALL HCHAR(13,A,32,384) : DISPLAY AT(14,1) : "START RECLAMATION ON:" : "WHI CH MINE:" : ACCEPT AT(15,12) : SIZE(1) : VALIDATE(DIGIT) : $ : IF A$="" THEN RETURN ELSE Z=VAL(A$) : IF MP(P,Z,Q)<4 OR MP(P,Z,Q)>6 THEN RETURN D=MP(P,Z,B)+.4 : IF T(P,A)-D<0 THEN NDISP Y AT(18,1) : NOT ENOUGH MONEY! : GOSUB 1430 : RETURN ELSE T(P,A)=T(P,A)-D : MP(P,Z,Q)=7 : MP(P,Z,K)=0 : MP(P,Z,L)=0 : DISPLAY AT(18,1) : "RECLAMA TION STARTED ON # : " : Z : DISP LAY AT(20,1) : USING 1470 : "REC. C OST:" : D : "CALL HCHAR(MP(P,Z,7),MP(P,Z,6),102+(Z-1)*8) : " : GOSUB 1430 : RETURN : CALL HCHAR(13,A,32,384) : DISPLAY AT(14,1) : "EXPAND,ON WHICH MINE:" : ACCEPT AT(14,22) : SIZE(1) : VALIDATE(DIGIT) : $ : IF A$="" THEN RETURN ELSE Z=VAL(A$) : IF MP(P,Z,Q)<2 OR MP(P,Z,Q)>5 THEN RETURN MP(P,Z,7),MP(P,Z,6),V() : DISP LAY AT(16,1) : USING 1470 : "CASH:" : T(P,A) : " : DISPLAY AT(17,1) : USING 1470 : "MILL SIZE:" : MP(P,Z,A) : DISP LAY AT(18,A) : USING 1480 : "MAX PR OD:" : MP(P,Z,A) : .00035 : " : DISPLAY AT(19,A) : USING 1480 : "QUALITY:" : V(A) : DISP LAY AT(20,A) : USING 1480 : "DEPTH:" : V(B) : ENVIRONM ENT : " : V(K) : DISP LAY AT(22,1) : "INCREASE CONST. : $ E(DIGIT) : $ : IF A$="" THEN RETURN N=VAL(A$) : GOSUB 1410 : IF T(P,A)-N<F THEN DISP LAY AT(24,1) : "YOU CAN'T AFFORD THAT MUCH!" : GOSUB 1430 : RETURN MP(P,Z,A)=MP(P,Z,A)+N : T(P,A)=T(P,A)-N : F : MP(P,Z,B)=MP(P,Z,B)+F : MP(P,Z,K)=0 : MP(P,Z,Q)=3 : DISP LAY AT(24,1) : ENTER-RETURN TO MENU : GOSUB 1420 : RETURN : ! SURVEY ROUTINE : IF T(P,B)=Q THEN RETURN ELSE CALL HCHAR(13,A,32,384) : "DISPLAY AT(14,A) : USING 1470 : "CASH:" : T(P,A) : DISP LAY AT(16,A) : "COST:" : QUALITY : " : DEPTH:" : ENVIRONMENT : " : ES DX TO MOVE" : SPACE TO SELECT : "ENTER T O KEEP" : I=Q : CALL SPRITE(#A,141,2,H,13,121) : GOSUB 1420 : CALL GCHAR(I,H,C) : I=F C=83 THEN CALL HCHAR(I,H,128) : IF F=69 AND I>A THEN I=I-A : GOTO 1270 : IF F=88 AND I<10 THEN I=I+A : GOTO 1270 : IF F=83 AND H>K THEN H=H-A : GOTO 1270 : IF F=68 AND H<30 THEN H=H+A : GOTO 1270 : IF F=32 AND C=143 OR C=128 THEN 1280 : IF F=13 THEN 1330 ELSE 1200 : CALL LOCATE(#1,(I-A)*8+A,(H-A)*8+A) : GOTO 1200 : CALL VV(BS,H+1,V()) : IF C=143 THEN D=V(B)+.115+1000 ELSE D=1000 : IF T(P,A)-D<0 THEN DISPLAY AT(13,A) BEEP : NOT ENOUGH MONEY!! : GOSUB 1430 : RETURN : T(P,A)=T(P,A)-D : "DISPLAY AT(14,A) : USING 1470 : "CASH:" : T(P,A) : DISPLA Y AT(16,A) : USING 1470 : "COST:" : D : DISP LAY AT(18,A) : USING 1480 : "QUALIT Y:" : V(A) : " : DISPLAY AT(19,A) : USING 1480 : "DEPTH:" : V(B) :

```

Continued


```

1320 DISPLAY "V(K) USING 1480: "ENVIRONMNT: V(K) CALL HCHAR(1,H,83)::
1330 CALL DELSPRITE(#1):: IF C<>83 THEN
1340 RETURN ELSE T(P,B)=T(P,B)+1
1350 FOR Z=A TO Q:: IF MP(P,Z,Q)=0 THEN
1360 NEXT Z:: RETURN
1370 MP(P,Z,Q)=1:: CALL HCHAR(1,H,95+(P-A)*8+Z):: MP(P,Z,6)=1:: MP(P,Z,7)=H:: MP(P,Z,8)=V(L):: RETURN
1380 MISC. ROUTINES & DATA
1390 S=(MP(P,Z,K))/(V(B)+(V(K)*.5)+1)::
1400 S=S*((MP(P,Z,8)*.5)+V(A))*0.0003::
S=INT(MIN(S,MP(P,Z,A)*.00035)):: RETURN
1410 F=INT((V(K)*.0133)*(MP(P,Z,A)+N))+5
0000:: DISPLAY AT(23,1) USING 1470
:: RECLAMATION: F:: RETURN
1420 CALL KEY(G,F,Z):: IF Z=G THEN 1420
ELSE CALL SOUND(1,1200,10):: RETURN
1430 FOR N=A TO 800:: NEXT N:: RETURN

```

```

1440 CALL HCHAR(12,A,32,416):: DISPLAY A
T(16,A):: ARE YOU SURE (Y/N)?N::
ACCEPT AT(16,20) SIZE(-1) VALIDATE("YN"):: AS
1450 IF AS=" " OR AS="N" THEN RETURN
FOR Z=1 TO J:: DISPLAY AT(14+Z,1)::
1460 USING 1470: FL$(Z,A):: NEXT Z::
: DISPLAY AT(24,1):: PRESS ENTER TO
END:: GOSUB 1420:: END
1470 IMAGE:#####
1480 IMAGE:#####
1490 DATA 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,112,113,114,115,116,117,118,119,120,121,122,123,124,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,143,144,145,146,147,148,149,150,151,152,153,154,155,156,157,158,159,160,161,162,163,164,165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,182,183,184,185,186,187,188,189,190,191,192,193,194,195,196,197,198,199,200,201,202,203,204,205,206,207,208,209,210,211,212,213,214,215,216,217,218,219,220,221,222,223,224,225,226,227,228,229,230,231,232,233,234,235,236,237,238,239,240,241,242,243,244,245,246,247,248,249,250,251,252,253,254,255,256,257,258,259,260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,278,279,280,281,282,283,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,302,303,304,305,306,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,325,326,327,328,329,330,331,332,333,334,335,336,337,338,339,340,341,342,343,344,345,346,347,348,349,350,351,352,353,354,355,356,357,358,359,360,361,362,363,364,365,366,367,368,369,370,371,372,373,374,375,376,377,378,379,380,381,382,383,384,385,386,387,388,389,390,391,392,393,394,395,396,397,398,399,400,401,402,403,404,405,406,407,408,409,410,411,412,413,414,415,416,417,418,419,420,421,422,423,424,425,426,427,428,429,430,431,432,433,434,435,436,437,438,439,440,441,442,443,444,445,446,447,448,449,450,451,452,453,454,455,456,457,458,459,460,461,462,463,464,465,466,467,468,469,470,471,472,473,474,475,476,477,478,479,480,481,482,483,484,485,486,487,488,489,490,491,492,493,494,495,496,497,498,499,500,501,502,503,504,505,506,507,508,509,510,511,512,513,514,515,516,517,518,519,520,521,522,523,524,525,526,527,528,529,530,531,532,533,534,535,536,537,538,539,540,541,542,543,544,545,546,547,548,549,550,551,552,553,554,555,556,557,558,559,560,561,562,563,564,565,566,567,568,569,570,571,572,573,574,575,576,577,578,579,580,581,582,583,584,585,586,587,588,589,590,591,592,593,594,595,596,597,598,599,600,601,602,603,604,605,606,607,608,609,610,611,612,613,614,615,616,617,618,619,620,621,622,623,624,625,626,627,628,629,630,631,632,633,634,635,636,637,638,639,640,641,642,643,644,645,646,647,648,649,650,651,652,653,654,655,656,657,658,659,660,661,662,663,664,665,666,667,668,669,670,671,672,673,674,675,676,677,678,679,680,681,682,683,684,685,686,687,688,689,690,691,692,693,694,695,696,697,698,699,700,701,702,703,704,705,706,707,708,709,710,711,712,713,714,715,716,717,718,719,720,721,722,723,724,725,726,727,728,729,730,731,732,733,734,735,736,737,738,739,740,741,742,743,744,745,746,747,748,749,750,751,752,753,754,755,756,757,758,759,760,761,762,763,764,765,766,767,768,769,770,771,772,773,774,775,776,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,796,797,798,799,800,801,802,803,804,805,806,807,808,809,810,811,812,813,814,815,816,817,818,819,820,821,822,823,824,825,826,827,828,829,830,831,832,833,834,835,836,837,838,839,840,841,842,843,844,845,846,847,848,849,850,851,852,853,854,855,856,857,858,859,860,861,862,863,864,865,866,867,868,869,870,871,872,873,874,875,876,877,878,879,880,881,882,883,884,885,886,887,888,889,890,891,892,893,894,895,896,897,898,899,900,901,902,903,904,905,906,907,908,909,910,911,912,913,914,915,916,917,918,919,920,921,922,923,924,925,926,927,928,929,930,931,932,933,934,935,936,937,938,939,940,941,942,943,944,945,946,947,948,949,950,951,952,953,954,955,956,957,958,959,960,961,962,963,964,965,966,967,968,969,970,971,972,973,974,975,976,977,978,979,980,981,982,983,984,985,986,987,988,989,990,991,992,993,994,995,996,997,998,999,1000,1001,1002,1003,1004,1005,1006,1007,1008,1009,1010,1011,1012,1013,1014,1015,1016,1017,1018,1019,1020,1021,1022,1023,1024,1025,1026,1027,1028,1029,1030,1031,1032,1033,1034,1035,1036,1037,1038,1039,1040,1041,1042,1043,1044,1045,1046,1047,1048,1049,1050,1051,1052,1053,1054,1055,1056,1057,1058,1059,1060,1061,1062,1063,1064,1065,1066,1067,1068,1069,1070,1071,1072,1073,1074,1075,1076,1077,1078,1079,1080,1081,1082,1083,1084,1085,1086,1087,1088,1089,1090,1091,1092,1093,1094,1095,1096,1097,1098,1099,1100,1101,1102,1103,1104,1105,1106,1107,1108,1109,1110,1111,1112,1113,1114,1115,1116,1117,1118,1119,1120,1121,1122,1123,1124,1125,1126,1127,1128,1129,1130,1131,1132,1133,1134,1135,1136,1137,1138,1139,1140,1141,1142,1143,1144,1145,1146,1147,1148,1149,1150,1151,1152,1153,1154,1155,1156,1157,1158,1159,1160,1161,1162,1163,1164,1165,1166,1167,1168,1169,1170,1171,1172,1173,1174,1175,1176,1177,1178,1179,1180,1181,1182,1183,1184,1185,1186,1187,1188,1189,1190,1191,1192,1193,1194,1195,1196,1197,1198,1199,1200,1201,1202,1203,1204,1205,1206,1207,1208,1209,1210,1211,1212,1213,1214,1215,1216,1217,1218,1219,1220,1221,1222,1223,1224,1225,1226,1227,1228,1229,1230,1231,1232,1233,1234,1235,1236,1237,1238,1239,1240,1241,1242,1243,1244,1245,1246,1247,1248,1249,1250,1251,1252,1253,1254,1255,1256,1257,1258,1259,1260,1261,1262,1263,1264,1265,1266,1267,1268,1269,1270,1271,1272,1273,1274,1275,1276,1277,1278,1279,1280,1281,1282,1283,1284,1285,1286,1287,1288,1289,1290,1291,1292,1293,1294,1295,1296,1297,1298,1299,1300,1301,1302,1303,1304,1305,1306,1307,1308,1309,1310,1311,1312,1313,1314,1315,1316,1317,1318,1319,1320,1321,1322,1323,1324,1325,1326,1327,1328,1329,1330,1331,1332,1333,1334,1335,1336,1337,1338,1339,1340,1341,1342,1343,1344,1345,1346,1347,1348,1349,1350,1351,1352,1353,1354,1355,1356,1357,1358,1359,1360,1361,1362,1363,1364,1365,1366,1367,1368,1369,1370,1371,1372,1373,1374,1375,1376,1377,1378,1379,1380,1381,1382,1383,1384,1385,1386,1387,1388,1389,1390,1391,1392,1393,1394,1395,1396,1397,1398,1399,1400,1401,1402,1403,1404,1405,1406,1407,1408,1409,1410,1411,1412,1413,1414,1415,1416,1417,1418,1419,1420,1421,1422,1423,1424,1425,1426,1427,1428,1429,1430,1431,1432,1433,1434,1435,1436,1437,1438,1439,1440,1441,1442,1443,1444,1445,1446,1447,1448,1449,1450,1451,1452,1453,1454,1455,1456,1457,1458,1459,1460,1461,1462,1463,1464,1465,1466,1467,1468,1469,1470,1471,1472,1473,1474,1475,1476,1477,1478,1479,1480,1481,1482,1483,1484,1485,1486,1487,1488,1489,1490,1491,1492,1493,1494,1495,1496,1497,1498,1499,1500,1501,1502,1503,1504,1505,1506,1507,1508,1509,1510,1511,1512,1513,1514,1515,1516,1517,1518,1519,1520,1521,1522,1523,1524,1525,1526,1527,1528,1529,1530,1531,1532,1533,1534,1535,1536,1537,1538,1539,1540,1541,1542,1543,1544,1545,1546,1547,1548,1549,1550,1551,1552,1553,1554,1555,1556,1557,1558,1559,1560,1561,1562,1563,1564,1565,1566,1567,1568,1569,1570,1571,1572,1573,1574,1575,1576,1577,1578,1579,1580,1581,1582,1583,1584,1585,1586,1587,1588,1589,1590,1591,1592,1593,1594,1595,1596,1597,1598,1599,1600,1601,1602,1603,1604,1605,1606,1607,1608,1609,1610,1611,1612,1613,1614,1615,1616,1617,1618,1619,1620,1621,1622,1623,1624,1625,1626,1627,1628,1629,1630,1631,1632,1633,1634,1635,1636,1637,1638,1639,1640,1641,1642,1643,1644,1645,1646,1647,1648,1649,1650,1651,1652,1653,1654,1655,1656,1657,1658,1659,1660,1661,1662,1663,1664,1665,1666,1667,1668,1669,1670,1671,1672,1673,1674,1675,1676,1677,1678,1679,1680,1681,1682,1683,1684,1685,1686,1687,1688,1689,1690,1691,1692,1693,1694,1695,1696,1697,1698,1699,1700,1701,1702,1703,1704,1705,1706,1707,1708,1709,1710,1711,1712,1713,1714,1715,1716,1717,1718,1719,1720,1721,1722,1723,1724,1725,1726,1727,1728,1729,1730,1731,1732,1733,1734,1735,1736,1737,1738,1739,1740,1741,1742,1743,1744,1745,1746,1747,1748,1749,1750,1751,1752,1753,1754,1755,1756,1757,1758,1759,1760,1761,1762,1763,1764,1765,1766,1767,1768,1769,1770,1771,1772,1773,1774,1775,1776,1777,1778,1779,1780,1781,1782,1783,1784,1785,1786,1787,1788,1789,1790,1791,1792,1793,1794,1795,1796,1797,1798,1799,1800,1801,1802,1803,1804,1805,1806,1807,1808,1809,1810,1811,1812,1813,1814,1815,1816,1817,1818,1819,1820,1821,1822,1823,1824,1825,1826,1827,1828,1829,1830,1831,1832,1833,1834,1835,1836,1837,1838,1839,1840,1841,1842,1843,1844,1845,1846,1847,1848,1849,1850,1851,1852,1853,1854,1855,1856,1857,1858,1859,1860,1861,1862,1863,1864,1865,1866,1867,1868,1869,1870,1871,1872,1873,1874,1875,1876,1877,1878,1879,1880,1881,1882,1883,1884,1885,1886,1887,1888,1889,1890,1891,1892,1893,1894,1895,1896,1897,1898,1899,1900,1901,1902,1903,1904,1905,1906,1907,1908,1909,1910,1911,1912,1913,1914,1915,1916,1917,1918,1919,1920,1921,1922,1923,1924,1925,1926,1927,1928,1929,1930,1931,1932,1933,1934,1935,1936,1937,1938,1939,1940,1941,1942,1943,1944,1945,1946,1947,1948,1949,1950,1951,1952,1953,1954,1955,1956,1957,1958,1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,1987,1988,1989,1990,1991,1992,1993,1994,1995,1996,1997,1998,1999,2000,2001,2002,2003,2004,2005,2006,2007,2008,2009,2010,2011,2012,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022,2023,2024,2025,2026,2027,2028,2029,2030,2031,2032,2033,2034,2035,2036,2037,2038,2039,2040,2041,2042,2043,2044,2045,2046,2047,2048,2049,2050,2051,2052,2053,2054,2055,2056,2057,2058,2059,2060,2061,2062,2063,2064,2065,2066,2067,2068,2069,2070,2071,2072,2073,2074,2075,2076,2077,2078,2079,2080,2081,2082,2083,2084,2085,2086,2087,2088,2089,2090,2091,2092,2093,2094,2095,2096,2097,2098,2099,2100,2101,2102,2103,2104,2105,2106,2107,2108,2109,2110,2111,2112,2113,2114,2115,2116,2117,2118,2119,2120,2121,2122,2123,2124,2125,2126,2127,2128,2129,2130,2131,2132,2133,2134,2135,2136,2137,2138,2139,2140,2141,2142,2143,2144,2145,2146,2147,2148,2149,2150,2151,2152,2153,2154,2155,2156,2157,2158,2159,2160,2161,2162,2163,2164,2165,2166,2167,2168,2169,2170,2171,2172,2173,2174,2175,2176,2177,2178,2179,2180,2181,2182,2183,2184,2185,2186,2187,2188,2189,2190,2191,2192,2193,2194,2195,2196,2197,2198,2199,2200,2201,2202,2203,2204,2205,2206,2207,2208,2209,2210,2211,2212,2213,2214,2215,2216,2217,2218,2219,2220,2221,2222,2223,2224,2225,2226,2227,2228,2229,2230,2231,2232,2233,2234,2235,2236,2237,2238,2239,2240,2241,2242,2243,2244,2245,2246,2247,2248,2249,2250,2251,2252,2253,2254,2255,2256,2257,2258,2259,2260,2261,2262,2263,2264,2265,2266,2267,2268,2269,2270,2271,2272,2273,2274,2275,2276,2277,2278,2279,2280,2281,2282,2283,2284,2285,2286,2287,2288,2289,2290,2291,2292,2293,2294,2295,2296,2297,2298,2299,2300,2301,2302,2303,2304,2305,2306,2307,2308,2309,2310,2311,2312,2313,2314,2315,2316,2317,2318,2319,2320,2321,2322,2323,2324,2325,2326,2327,2328,2329,2330,2331,2332,2333,2334,2335,2336,2337,2338,2339,2340,2341,2342,2343,2344,2345,2346,2347,2348,2349,2350,2351,2352,2353,2354,2355,2356,2357,2358,2359,2360,2361,2362,2363,2364,2365,2366,2367,2368,2369,2370,2371,2372,2373,2374,2375,2376,2377,2378,2379,2380,2381,2382,2383,2384,2385,2386,2387,2388,2389,2390,2391,2392,2393,2394,2395,2396,2397,2398,2399,2400,2401,2402,2403,2404,2405,2406,2407,2408,2409,2410,2411,2412,2413,2414,2415,2416,2417,2418,2419,2420,2421,2422,2423,2424,2425,2426,2427,2428,2429,2430,2431,2432,2433,2434,2435,2436,2437,2438,2439,2440,2441,2442,2443,2444,2445,2446,2447,2448,2449,2450,2451,2452,2453,2454,2455,2456,2457,2458,2459,2460,2461,2462,2463,2464,2465,2466,2467,2468,2469,2470,2471,2472,2473,2474,2475,2476,2477,2478,2479,2480,2481,2482,2483,2484,2485,2486,2487,2488,2489,2490,2491,2492,2493,2494,2495,2496,2497,2498,2499,2500,2501,2502,2503,2504,2505,2506,2507,2508,2509,2510,2511,2512,2513,2514,2515,2516,2517,2518,2519,2520,2521,2522,2523,2524,2525,2526,2527,2528,2529,2530,2531,2532,2533,2534,2535,2536,2537,2538,2539,2540,2541,2542,2543,2544,2545,2546,2547,2548,2549,2550,2551,2552,2553,2554,2555,2556,2557,2558,2559,2560,2561,2562,2563,2564,2565,2566,2567,25
```



```

710 PRINT "CRSRDOWN" PRESS + TO GO UP
720 PRINT "AVE" PRESS - TO GO DOWN AN OCTA
730 PRINT "CRSRDOWN" SUSTAIN IS OFF.
740 PRINT "S" TO CHANGE. FILTER IS OFF. P
750 PRINT "F1" TO CHANGE. CRSRDOWN
760 PRINT TAB(9) "F3" LOW PASS OFF
770 PRINT TAB(9) "F5" BAND PASS OFF
780 PRINT TAB(9) "F7" HIGH PASS OFF
790 PRINT "CRSRDOWN" CUT OFF=1024
800 PRINT "CRSRDOWN" TO CHANGE.
810 PRINT "3CRSRDOWN" PRESS - TO QUIT
820 RETURN
830 TE=PEEK(VL):GOSUB850:IF TF THEN PRI
840 NT="CTRL RVSON"ON:RETURN
850 POKE 781,R:POKE 782,C:POKE 783,..:SY
860 S 65520:RETURN
870 RETURN
880 S=54272:FL=S:FH=S+1:CR=S+4:AD=S+5:S
890 R=S+6:CU=S+21:RE=S+23:VL=S+24
900 SU=0:FI=0:LF=0:BF=0:HP=0
910 FOR I=S TO VL:POKE I,0:NEXT
920 POKECR,32:POKEAD,21:POKESR,165:POKE
930 VL,15:POKERE,112:POKECU+1,128
940 DEF FN PF(FR)=FR/.06096:DEF FN FR(B
950 DEF FN HF(FR)=INT((FR-256)/(256*
960 DIM NT%(12,7,1)
970 FQ=16.3515977:FOR OC=0 TO 7:FOR NT=
980 1 TO 12
990 NT%(NT,OC,0)=FN LF(FN PF(FQ))
1000 NT%(NT,OC,1)=FN HF(FN PF(FQ))
1010 FQ=FN FR(FQ)=NEXT NT:OC=4
1020 DEF FN H(V)=INT((V*4096/25600)-
1030 DEF FN L(V)=INT((V*4096/25600)-
1040 RETURN
1050 FOR I=49152 TO 49295:READ D:CK=CK+D
1060 :POKE I,D:NEXT
1070 IF CK<>15281 THEN PRINT "** ERROR I
1080 N DATA STATEMENTS **:END
1090 SYS 49152:RETURN
1100 DATA 169,21,141,8,3,169,192,141
1110 DATA 9,3,169,70,3,141,96,169
1120 DATA 192,141,11,3,141,96,169
1130 DATA 32,30,192,76,174,167,165
1140 DATA 96,233,128,144,167,165
1150 DATA 6,76,243,167,76,165
1160 DATA 115,0,32,235,183,32,132
1170 DATA 208,9,32,39,184,164,20
1180 DATA 143,192,96,76,39,184,169
1190 DATA 133,13,32,115,0,201,194,240
1200 DATA 6,32,121,0,76,141,174,32
1210 DATA 115,0,32,241,174,169,107,133
1220 DATA 85,169,192,133,86,32,165,0
1230 DATA 76,141,173,183,32,21,72,165
1240 DATA 72,32,247,183,143,132,192,208
1250 DATA 8,164,20,185,143,192,76,26
1260 DATA 184,76,22,184,165,20,201,96
1270 DATA 176,4,165,21,201,212,96

```

Want to Get Published?

Fame, Fortune, Recognition!

See Your Name in Print!

Home Computer Magazine is looking for articles and programs in all areas of interest relevant to Apple, IBM, Commodore, and Texas Instruments home computers. Here are some of the kinds of material we would like you to submit:

Software

Have you written any programs in the areas of home productivity, education, or entertainment? Perhaps you've created unique software to help monitor personal finances, or a new contribution to computer-assisted instruction (CAI). Maybe you have an unusual new game—or a routine that makes certain computer operations easier to perform. Don't be shy. Even if you think your piece is "unpolished," it may still be a good idea. We will be glad to follow through with your concept—enhancing the program and converting it to work on the other machines we cover.

Product Reviews

Have you recently purchased a piece of hardware or software that hasn't come up to your expectations—or has, on the other hand, impressed you with its performance? We're looking for comprehensive product reviews from different perspectives.

Hardware Tips

Perhaps you've modified your microcomputer or have interfaced it with some unique or useful hardware. Send us your how-to-do-it story, complete with photos and/or diagrams.

Tutorials

Many of our articles are purely instructional. If you have extensive experience in some area of programming or other computer application, put your specialized knowledge down on paper and let us pass it on to our readers.

These are just some ideas. Perhaps you have others. If you're not a professional writer, don't worry. Our friendly editorial staff stands ready to help you polish your manuscripts. And we'll be more than happy to send you a copy of our author guidelines. Here are some comments from happy writers who have already published their work in our magazine:

"The people at Home Computer Magazine are fun to work with. And it's sure nice to get paid for writing about my favorite subject."
—Patricia Swift
Author of "Multiplan Medium" and other articles

"The artwork and layout are creative and contribute a lot to the presentation of my articles."
—Roger Kirchner
Author of "Missionary Impossible" and other articles

"It was gratifying to finally see my name in print after all the work I've done on my computer."
—Brian Lee
Author of "Market Madness"

"I was extremely impressed with the way my program was printed in HCM. It was very interesting to see the way the program was translated into the languages of the other popular computers and to read the comments of the people who reviewed the program. Truly a first class job! Thank you!"
—Craig Blazakis
Author of "Bird Brain"

"I was very pleased with the final presentation of my article. It is gratifying to see such judicious handling of an outside submission. The HCM staff fixed a program bug and expanded the application of the article to three other computers, while preserving the style of the article as submitted. The illustration added to the overall readability."
—Andrew Keith
Author of "Build a LOGO Adventure"

Please send your double-spaced, typed or printed manuscripts, photos, and disks or cassettes (recorded on both sides) if the article includes program material, to:

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



DeBUGS on Display

During the production of every issue, corrections and/or enhancements to our programs are completed and tested in our programming laboratory. As the new version of a program is compared to the *last published version* by our "cross-checking" computer, a listing of all the differences is produced, transmitted to the computerized typesetter, and formatted in the same fashion as our standard listings.

This procedure for "DeBugs on Display" offers two advantages: (1) a standard presentation for updating your HCM programs that is clear and straightforward, and (2) inclusion of all published changes in "update files" which are placed *ON DISK(TM)* at the same time that the corrections appear in print. This is of special significance to *ON DISK(TM)* subscribers, because the correction file can be directly "merged" with the original file—automatically updating it! The procedures for accomplishing this are included with the appropriate media. [TI users take note—the merge command is available in Extended BASIC only.—Ed.]

Good news for Commodore 64 users with Datasets! The utility program for merging files from tape is now available (see the Commodore Tech Note on page 55 of this issue for details).

If you are going to type the corrections from "DeBugs on Display" directly into the original program, follow these steps:

- 1.) Load the original program into your computer's memory.
- 2.) Key-in the corrections as directed in the "Program Typing Guide" at the beginning of the Listings section.
- 3.) Any lines in the listing of corrections that state "****DELETED LINE," are to be deleted from the original program by entering the line number only and pressing either the (ENTER) or (RETURN) key (depending on your computer).

Each set of program corrections is prefaced by an identification bar that tells you: (1) the program name, (2) the volume and number of HCM in which the program was first published, (3) the number of the *last published version*, and (4) the computer brand to which the correction applies. Make sure that you are working with the right listing to ensure satisfactory results.

THE ORGANIZER FILE MANAGER (HCM Vol. 5, No. 1)

The last level of DeBugs published in HCM Vol. 5, No. 2 as version .2

APPLE II Family

```

1180 REM = VERSION 5.1.3
730 FO = 1: GOSUB 1650: GOSUB 1740: RE
940 IF IN$ = ESC$ OR FL$ = " THEN RE
1155 V = 1: GOSUB 3030
1160 GOSUB 1740: GOSUB 1780
1170 IF IN$ = ESC$ OR FL$ = " THEN RE
1185 V = 2: GOSUB 3030
1186 ONERR GOTO 2630
1270 IF IN$ = ESC$ OR FL$ = " THEN RE
1285 GOSUB 3030
2025 IF FL$ = " THEN RETURN
2450 PRINT "PLACE DISK
IN DRIVE 1: V$; DR$; AND PRESS RETURN";
2470 HTAB 1: V$; DR$; CALL 868: GOSUB
1740:
3030 ONERR GOTO 3100
3040 PRINT CHR$(4) "VERIFY"; FL$
3050 IF V < 1 THEN 3070
3060 HTAB 5: V$; V$; 20: CALL 868
3061 PRINT "DO YOU WANT TO REPLACE: "; PR
INT FL$; (Y/N): GOSUB 3200
3062 IF KB = 217 THEN 960
3064 IF KB = 206 THEN 960
3070 IF V < 2 THEN 3090
3080 HTAB 5: V$; V$; 20: CALL 868
3081 PRINT "ARE YOU SURE YOU WANT TO DEL
ETE": PRINT FL$; (Y/N): GOSUB 32
00
3082 IF KB = 217 THEN RETURN
3085 IF KB = 206 THEN RUN
3090 RETURN
3100 X = PEEK (222)

```

```

3110 IF X < 6 AND X < 7 THEN GOTO
3120 IF V = 1 THEN 960
3130 HTAB 5: V$; V$; 20: CALL 868
3131 PRINT "FILE DOES NOT EXIST. PRESS R
ETURN TO CONTINUE."
3140 GOSUB 2500: IF KB < 141 AND KB <
> 155 THEN 3140
3150 RUN
3200 GOSUB 2500: IF (KB < 217) AND (K
B < 206) THEN
3205 IF KB = 155 THEN RUN
3210 RETURN

```

MARKET MADNESS (HCM Vol. 4, No. 4)

The last level of DeBugs published in HCM Vol. 5, No. 2 as version .3

IBM PC & IBM PCjr

```

1600 VERSION 4.4.4
270 LOCATE 23,1: INPUT "HOW MANY WEEKS (
AT LEAST 2)"; AS: GOSUB 1690: IF STA
T=1 THEN 270 ELSE NW=VAL(AS): IF NW<
2 THEN 270
1600 NEXT: LOCATE 15,1: PRINT "THE WINNER
IS "; NAM$(WIN): FOR Z=1 TO NP: IF Z=
WIN THEN 1610 ELSE IF NET(Z)=NET(WI
N) THEN PRINT " AND "; NAM$(Z):

```

Program Alterations for the Tandy 1000

Users of the Tandy 1000, a PC-compatible computer, have written to inform us that many of the IBM PC and PCjr programs published in HCM run without problems on the Tandy machine. We did discover one consistent program change that must be made in order for our programs to run on this PC-compatible. It is listed below. In each forthcoming issue, we will publish a list of line numbers for each program where alterations should be made. Please let us know if you discover any other problems in running our software on your Tandy 1000.

Unlike IBM BASICs, Tandy BASIC's CLS (Clear Screen) does not recognize screen line 25. Thus, the CLS command clears up to and including line 24, leaving line 25 unaltered. To counter this effect, replace each CLS command with this code:

```
CLS:LOCATE 25,1:PRINT SPACES(40)
```

In addition, trying to access line 25 before a KEY OFF statement has been executed causes an illegal function call, so precede the first LOCATE 25,1 command with a KEY OFF statement like this:

```
CLS:KEY OFF:LOCATE 25,1:PRINT SPACES(40)
```

Programs and line numbers to be altered in this issue are as follows:

ARCHEODROID

250, 270, 370, 380, 390, 400, 410, 450, 1440

TRIG-TRIX

250, 750, 830, 850, 860, 870, 880, 1660

MINE OVER MATTER

240, 310, 340, 360, 390, 430, 800, 880, 1060, 1160, 1410, 1590, 1610, 1640

RUN-DAY-VIEW

270, 290, 350, 420, 490, 880, 980, 1460, 1650, 1710, 1750, 1940, 2130, 2210, 2320, 2340, 2470, 2490

EGGS (IBMpressions)

240

CHARACTER GRAPHICS (IBM Tech Note)

260, 270, 310

3 FANTASTIC OFFERS

SGV5540685

HOME COMPUTERTM magazine

QUESTIONNAIRE

Complete and mail to: Home Computer Magazine • P.O. Box 70288 • Eugene, Oregon 97401

FOR ALL READERS

1. Where did you obtain this copy of Home Computer Magazine? ☐Subscriber ☐Supermarket ☐Bookstore
☐Users group ☐Newsstand ☐Computer Store ☐Friend ☐Library ☐Other _____
2. What types of software are you most interested in? ☐Educational ☐Entertainment ☐Computer Literacy
☐Household Management ☐Job-Related Applications ☐Business ☐Other _____
3. Are you ☐Male ☐Female ☐14 or younger ☐15-24 ☐25-34 ☐35-44 ☐45-54 ☐55+
4. Annual Household Income? ☐Under \$10,000 ☐\$10,000-\$14,999 ☐\$15,000-\$19,999 ☐\$20,000-\$24,999 ☐\$25,000-\$29,999
☐\$30,000-\$39,999 ☐\$40,000-\$49,999 ☐\$50,000+
5. Occupation? ☐Professional ☐Management ☐Teacher ☐Student ☐Other _____
6. What is your ZIP code?
7. What is the current month and year? _____
8. Do you presently own a Home Computer? ☐No ☐Yes. It is a ☐TI-99/4A ☐Apple II/II+ /Ile ☐Commodore 64
☐VIC-20 ☐IBM PC ☐PCjr ☐Other _____

FOR READERS WHO PLAN TO BUY A HOME COMPUTER

9. Which model do you think you'll purchase?
☐Apple Ile ☐Commodore 64 ☐VIC-20 ☐IBM PC ☐PCjr ☐TI-99/4A ☐Other _____
10. When do you expect that purchase to be? ☐less than 3 months ☐3-6 months ☐7-12 months ☐at least 1 year
11. What do you anticipate your primary use of a home computer will be? ☐Entertainment ☐Education
☐Computer Literacy ☐Household Management ☐Job-Related Applications ☐Business ☐Other _____

FOR PRESENT HOME COMPUTER USERS

12. Which home computer(s) do you currently own?
☐Apple II/II+ /Ile ☐Commodore 64 ☐VIC-20 ☐IBM PC ☐PCjr ☐TI-99/4A ☐Other _____
13. What is the primary use of your home computer? ☐Entertainment ☐Education ☐Computer Literacy ☐Business
☐Job-Related Applications ☐Household Management ☐Other _____
14. How often is your computer in use?
☐Less than 1 hour per week ☐1-4 hours ☐5-10 hours ☐11-15 hours ☐16-20 hours ☐over 20 hours
15. On the average, about how many program listings in each issue of HCM do you key into your computer and use?
☐None ☐1 ☐2 or 3 ☐4 or more
16. What peripherals do you currently use?
☐Disk System ☐Printer ☐Modem ☐Monochrome/Color Monitor ☐Other _____
17. What do you expect to buy within the next year? ☐Software ☐Disk system ☐Printer ☐Modem ☐Books
☐Magnetic Media ☐Monochrome/Color Monitor ☐Furniture & Accessories
18. How much do you expect to spend on computer-related products during the next year?
☐Less than \$25 ☐\$25-\$49 ☐\$50-\$99 ☐\$100-\$249 ☐\$250-\$490 ☐\$500-\$999 ☐\$1000-\$2499 ☐\$2500 or more

OPTIONAL: If you would like to help us by participating in a telephone interview, please include your telephone number (_____) - _____ here and the most convenient time you can be reached _____: _____ ☐AM ☐PM

FOR OFFICE USE ONLY

NAME _____

ZIP _____

AMOUNT PAID \$ _____

ACCT. _____

DATE _____

DATE SHIPPED _____

BY _____

VIA _____ AIR MAIL

_____ SECOND CLASS

_____ FIRST CLASS

_____ THIRD CLASS

_____ UPS (STANDARD)

_____ AIR CANADA

_____ UPS (BLUE LABEL)

_____ FOURTH CLASS

_____ OTHER _____

_____ FOREIGN SURFACE



COLLECT ALL BACK ISSUES

HOME COMPUTERTM magazine



Please Print

Name _____

Address _____

City _____ State _____ Zip _____

☐ Check or Money Order Enclosed **Total** _____
MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

Bill my ☐ VISA ☐ MasterCard Date Expires _____

Account No. _____

Tel. No. _____ Signature _____

Enclose payment or credit card information & mail with completed form to:

Home Computer Magazine
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

*PLEASE ALLOW 4-6 WEEKS FOR MAGAZINE DELIVERY.

ITEMS	PRICE
Home Computer Magazine Back Issues * (Circle Issues Desired) Vol. 4, No. 1 Vol. 4, No. 2 Vol. 4, No. 3 Vol. 4, No. 4 Vol. 4, No. 5 Vol. 5, No. 1 Vol. 5, No. 2	\$3.95 each — U.S. \$5.95 each — Canada \$5.95 each — Foreign Surface \$8.50 each — Foreign Air
ON DISK & ON TAPE Back Issues (Circle Issues Desired) Vol. 4, No. 1 Vol. 4, No. 2 Vol. 4, No. 3 Vol. 4, No. 4 Vol. 4, No. 5 Vol. 5, No. 1 Vol. 5, No. 2	\$6.95 each — U.S. \$8.95 each — Canada \$10.95 each — Foreign Air
SAVE EVEN MORE—Order Combined Sets (Circle Magazine & Media Sets Desired) Vol. 4, No. 1 Vol. 4, No. 2 Vol. 4, No. 3 Vol. 4, No. 4 Vol. 4, No. 5 Vol. 5, No. 1 Vol. 5, No. 2	\$8.90 each set — U.S. \$12.90 each set — Canada \$12.90 each set — Foreign Surface

Indicate your choice of media: (Check one)

ON TAPETM: ☐ C-64 ☐ TI-99/4A

ON DISKTM: ☐ Apple ☐ C-64 ☐ IBM PC ☐ IBM PCjr ☐ TI-99/4A

Defective media gladly exchanged. NO REFUNDS on media.

For more information see inside front cover.

Offer & Prices Subject To Change Without Notice.

The Best Of 99'er

—Book & Tape Set—

Special Close-Out Offer

See Page 10

See Page 10

Please Print

Name _____

Address _____

City _____ State _____ Zip _____

YES! Please send me *THE BEST OF 99'er ON TAPE*
along with my

FREE copy of the book *The Best Of 99'er*.

and the **SPECIAL BONUS** (while supplies last)

of *Simon's SaucerTM* and the *99'er Programmer's Guide*.

Enclosed is \$35. (Shipping and Handling **FREE** in U.S.!)
(Canada add \$5.)

For more information see page 10.

Offer & Prices Subject To Change Without Notice.

Defective media gladly exchanged. NO REFUNDS on book or media.

☐ Check or Money Order Enclosed

Total _____

MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

Bill my ☐ VISA ☐ MasterCard Date Expires _____

Account No. _____

Tel. No. _____ Signature _____

Enclose payment or credit card information & mail with completed form to:

Home Computer Magazine
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

*PLEASE ALLOW 6-8 WEEKS FOR DELIVERY.



Save \$ \$ \$ On BACK ISSUES of



99'er Home Computer Magazine, Disk, & Tape Back Issues
Exclusively For The TI-99/4A Home Computer

SPECIAL CLOSE-OUT PRICES
FOR MAGAZINES, DISKS & TAPES
NOW IN EFFECT FOR TI-99/4A USERS!

Please Print

ORDERS LIMITED TO QUANTITIES ON HAND.

Name _____

Address _____

City _____ State _____ Zip _____

☐ Check or Money Order Enclosed **Total** _____
MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

Bill my ☐ VISA ☐ MasterCard Date Expires _____

Account No. _____

Tel. No. _____ Signature _____

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

*PLEASE ALLOW 4-6 WEEKS FOR MAGAZINE DELIVERY.

ISSUE	Magazine Only	Mag. & Media Set	Media Only (Disk or Tape)	ISSUE	Magazine Only	Mag. & Media Set	Media Only (Disk or Tape)	ISSUE	Magazine Only	Mag. & Media Set	Media Only (Disk or Tape)
Vol. 1 No. 6	NOT AVAILABLE			Mar. '83				Aug. '83			
Nov. '82	NOT AVAILABLE			Apr. '83				Sept. '83			
Dec. '82				May '83				Oct. '83			
Jan. '83				June '83				Nov. '83			
Feb. '83				July '83							

INDICATE CHOICE OF MEDIA:
☐ DISK ☐ TAPE

Place an "X" in the corresponding box for each item you wish to order.

QTY	MAGAZINE PRICES*			MAG/MEDIA SET PRICES			MEDIA PRICES		
	U.S.	Canada	Foreign†	U.S.	Canada	Foreign†	U.S.	Canada	Foreign†
3	\$6.95	\$9.95	\$9.95	\$14.95	\$19.95	\$21.95	\$10.95	\$12.95	\$14.95
6	\$11.95	\$17.95	\$17.95	\$28.50	\$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

Defective media gladly exchanged.

NO REFUNDS on media.

† Magazines Shipped to Foreign Countries via Surface

Offer & Prices Subject To Change Without Notice.

For more information see inside back cover.

BCV5540685

Offer Expires July 30th 1985

SPECIAL LIMITED-TIME GET-ACQUAINTED OFFER

SAVE \$14.50[†] Over Single-Issue Price



Enter or Renew a 10-issue* ON DISK™ or ON TAPE™ Program Subscription For Only \$45 and Receive Your Choice of One Gift Shown Below

[†] Program subscription price increase goes into effect after July 30, 1985

*10 issues will be added on to any remaining issues you might still be entitled to because of a current program subscription or media given as premiums for subscribing to Home Computer Magazine.

For Apple II Family, C-64, IBM PC & PCjr Users

ON DISK Revue™ Volume 1

- 10 Programs On A Ready-To-Run Disk
- Exciting Activities For The Entire Family—At An Affordable Price
- Entertainment, Productivity & Education Plus Valuable Programming Secrets

Valuable Proof-of-Purchase Coupon For Gift Redemption

An Attractive Library-Storage Slipcase

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



A \$14.95 Premium Value!

Disk Stored in Re-usable Protective Pouch

For TI-99/4A Users



Your Choice of One Gift

4 Tapes or 4 Disks from 1 Group Below:

Group A	Group B	Group C
December, 1982 to March, 1983	April, 1983 to July, 1983	August, 1983 to November, 1983

OR

6 Magazines from 1 Group Below:

Group 1	Group 2
December, 1982 to May, 1983	June, 1983 to November, 1983

OR

Best of 99'er—The 360-Page "Classic" Reference Book.

An \$11.95 to \$19.95* Premium Value!

* 4 Tapes or Disks, \$14.95; 6 Magazines, \$11.95; Best of 99'er, \$19.95

Yes, please enter my 10-issue program subscription immediately. My computer type is:

☐ Apple II Family ☐ IBM PC ☐ IBM PCjr ☐ C-64 (disk) ☐ C-64 (tape)*

* NOTE: C-64 tape subscribers will receive as their premium, the disk-based ON DISK Revue Volume 1.

Please Print Name _____

Address _____

City _____ State _____ Zip _____

☐ Check or Money Order Enclosed ☐ Total _____

MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

Bill my ☐ VISA ☐ MasterCard Date Expires _____

Account No. _____

Tel. No. _____ Signature _____

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

Please allow 4-6 weeks for your first issue.

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. Defective media gladly exchanged. NO REFUNDS on shipped media.

PSV2540685

—TI-OWNERS ONLY—

I Wish To Subscribe To:

☐ ON TAPE, or ☐ ON DISK

Select Your Premium Below:

(Mark only one (1) box for your choice)

Either

99'er Home Computer Magazine[†]

ON DISK ON TAPE

Group A	<input type="checkbox"/>	<input type="checkbox"/>
Group B	<input type="checkbox"/>	<input type="checkbox"/>
Group C	<input type="checkbox"/>	<input type="checkbox"/>

OR

6 Issues of 99'er Home Computer Magazine[†]

- ☐ Group 1
☐ Group 2

OR

Best of 99'er ☐

[†]For editorial and program information, refer to the inside back cover of this magazine.

Canada add \$5 for software subscription.

Software subscription not available in other countries at this time.

NOVEMBER, 1983 (Partial Contents)

- Education with your Home Computer • Five Creative Learning Activities for Children • Let's Build Something We Have No Fear, Assembly Language Won't Byte, Part 2
- Squeezing the Most Out of TI BASIC • TiWRITER Tutorial • LOGO Lexicon • Interview with Dale Osborne • TiWRITER: At Home In The Office • PLATO's Progress
- Les Izmore and Debug • The Multiphan Medium • Computer Arrested Instruction—99'er Interviews the Kid • Gameware Buffet: Taco Man and Robo Chase • Game Reviews of Jail Break and Arithmetex • Hall of Fame • 99'er Digest • and much, much more.

THESE 2 MAGAZINES OUT OF PRINT
TAPES AND DISKS ARE STILL AVAILABLE

ALL PROGRAMS IN THIS MAGAZINE



ONLY \$4.95* ON DISK OR TAPE!

The same high-quality Apple, 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!"

To Order, Use The Bind-In Card Inside Rear Cover. Offer & Prices Subject To Change Without Notice.

* Current Single-Issue Price Only — See Center Bind-In Card For Back-Issue Prices & Special Program Subscription Offer.