Software Giveaway with Purchase of this Magazine

See bound-in card behind front cover

HOME COMPUTER magazine™

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

Vol.4 No.1

$3.00 in USA
$3.50 in Canada

PCjr: How to Buy, Operate, and Program the Peanut

Spritely Tricks to Pep Up Commodore Programs

Creating a Real Data Base on the Unexpanded VIC-20

Apple 3-D Graphics Made Easy

Super Sound & Music on the TI-99/4A

- LOGO Poetry 'n Motion
- A Bonanza of Key-In-and-Run Software for Apple II, C-64, VIC-20, IBM PCjr and TI-99/4A
- Computer Assisted Savings
- Game Reviews Galore
- Word Processing at Home

FREE Bound-in Home Compter Digest

THE SECRETS OF PERFECT MEMORY: ONE AND ONE HALF EARTH DOLLARS

AT LAST: THE WHOLE TRUTH ABOUT FLOPPIES.

Amazing book reveals all!

How to keep from brainwashing your disk so it never loses it's memory.

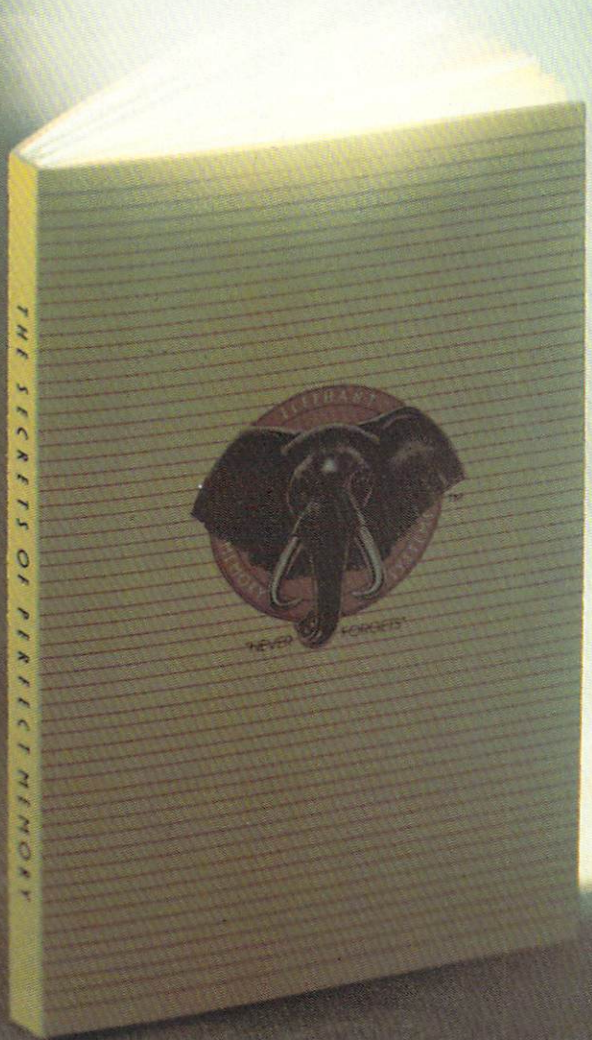
How fingerprints can actually damage disks. Unretouched Kirlian photographs of UFO's (Unidentified Floppy Objects)! The incredible importance of making copies: the Department of Redundancy Department—and what goes on when it goes on! Powerful secret methods that scientists claim can actually prevent computer amnesia! All this, and much more . . .

In short, it's an 80-page plain-English, graphically stunning, pocket-sized definitive guide to the care and feeding of flexible disks.

For The Book, ask your nearest computer store that sells Elephant™ disks, and bring along one and one half earth dollars.

For the name of the store, ask us.

ELEPHANT MEMORY SYSTEMS® Marketed exclusively by Leading Edge Products, Inc., Information Systems and Supplies Division, 55 Providence Highway, Norwood, MA 02062. Call toll free 1-800-343-8413. In Massachusetts, call collect (617) 769-8150, Telex 951-624.



2049
XXX
ROSS
Miner 2049er™
TI99/4A
TIGERVISION™
"Help Bounty Bob™ mine an abandoned uranium mine loaded with radioactive waste, deadly mutant organisms and perform many dangerous tasks on **eight different** play fields."
TIGERVISION™
Tiger Electronic Toys, Inc.
909 Orchard
Mundelein, IL 60060
(312) 949-8100
"TI99" and "4A" are registered trademarks of Texas Instruments.
Miner 2049er is a registered trademark of Big 5 Software
Licensed through Compu-Vid, No. Hollywood, CA

T&F SOFTWARE

CheckEase!™

Professional-Quality Checkbook:

For VIC-20 and Commodore 64

Now you have the power of a professional quality Check Register System. Maintain multiple checking accounts, complete with full checkbook reconciliation and 16 budget categories. Change or delete any check, check or deposit amount, or deduction and CheckEase! will automatically update all balance figures. Review checks forward, backward or by check number. Configure for RS232 or compatable Commodore printer. Post checks as they clear the bank. Upgrade data from cassette to disk. Print by check number, category or if item is tax deductable. Commodore 64 and VIC-20 users can even save months worth of check data in a format compatable with Commodore's *Personal Finance* package for later analyzation.

$24.95 cassette (VIC-20 min. 8K),

$29.95 cassette: Commodore 64, *Atari 400®/800®/1200 XL®

$34.95 disk: Commodore 64, *Atari 400®/800®/1200 XL®, *IBM PC, *APPLE II/IIplus/IIe®

SearchSeries™

First with Arcade-Quality Games:

For VIC-20 and Commodore 64

There are 374 letters on the screen. Concealed within are 20 words: 10 across and 10 down. You have 10 minutes. When you've found a hidden word, it changes color. Every game features a new screen. Over 300 different words and thousands of new games possible. **$19.95** on cassette.

WordSearch 3 categories: Capitals, Jumbled and Animals.

SportSearch 3 categories: Pro Teams, College Teams and Sport Games.

ArcadeSearch 3 categories: Home Video Games, Arcade Video Games and Famous Video Game Characters.

Space Sentinel™

For Commodore 64

Planet Earth is under attack by ruthless aliens who hurl heat missles at our polar ice caps. Will the Earth flood? As the orbiting Space Sentinel, the Earth's fate is up to you. If you can hold out against the merciless attackers, Earth's population will have time to escape and colonize a new home planet. Complete sprite & character graphics with 3-voice sound. **$29.95** on disk. Joystick, Diskdrive & Commodore 64® required.

*AVAILABLE 4TH QUARTER '83

Available at finer Software Stores everywhere.

Or Call (213) 501-5845 for the name of your local dealer or distributor.

T&F SOFTWARE

T&F Software Company 10902 Riverside Drive / North Hollywood, California 91602. (213) **501-5845**

On Screen

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

It was the best of times;
It was the worst of times.
—Charles Dickens, *A Tale of Two Cities*

"...a young, volatile world in which *change* is the only constant... an unforgiving world where the life and death of products depend on time-critical windows of opportunity."

It was the worst of times for Texas Instruments to exit the consumer computer marketplace: Their Home Computer was, at last, positioned correctly; an extremely visible, no-nonsense network TV campaign presaged strong holiday sales; dozens of the industry's finest software titles had been "cherry-picked," converted, and scheduled for pre-Christmas release; and a new generation of powerful, upwardly compatible hardware was about to be launched.

Ironically, it was also the best of times—that is, the best of times for hundreds of thousands of holiday shoppers looking for quality, affordable gifts or virtually no-risk tickets of entry to the intriguing world of home computing.

In the young, volatile world of the home computer industry, *change* is the only constant. It's an unforgiving world where the life and death of products depend on time-critical windows of opportunity—a place where even the smallest slip can prove fatal to the weak, and where a mountain of silica possessed by the strong must eventually drain through the corporate hour-glass of fiscal responsibility and stockholder impatience.

In such a dynamic environment, the worst thing any participant can do is stand still. Computer publications are no exception. A user magazine must be flexible enough to accommodate new reader interests, and perceptive enough to know when to alter its course.

This magazine is now on an exciting new heading. What started as *99'er Magazine* nearly four years ago—and later became *99'er Home Computer Magazine*—is now, simply, *Home Computer Magazine*. Our heritage is reflected in the name change: We pioneered the concept of a magazine exclusively for *home* users of computers, and nurtured it to become the world's leading machine-specific computer magazine. Now, we have broadened our TI-exclusive beat to include the three brands which we feel will be most in demand for home use during the remainder of the 80's: Apple, Commodore, and IBM. And as evidenced by this issue, we haven't sacrificed one bit of the quantity, quality, or comprehensiveness of our traditional 99/4A-related content.

Over the last several years, a close rapport and mutual respect has developed between us and our readers. We've listened to you, and learned. Today, with millions of published words behind us, the magazine's editorial, technical, and design teams have the requisite knowledge and sensitivity to make *Home Computer Magazine* the most valuable information resource—bar none—for present and future home users.

I dare to make this claim—and the editorial promise to keep it valid—because of the care and planning that go into each issue. Crafting a quality magazine with enduring reference value is our life's work; it is far removed from the prevalent practice of belching out prodigious masses of paper that are thrown together as "catalogs" *solely* for the purpose of carrying advertising messages for a span of thirty days.

We absolutely have no room within our pages for editorial "filler." The fat stays out. Instead, you get a well-balanced diet of facts—not fluff! You'll find carefully written and illustrated articles that are interrelated to, and integrated with other material in the same issue. Our goal is to produce a well-orchestrated learning environment that encourages self-paced exploration, and is, above all, *fun*.

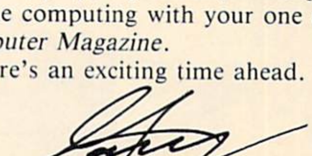
Then, of course, there is our software. Although we do have more key-in-and-run programs than anyone else, it is the unique way we present program listings—for maximum clarity and minimum typing errors—that has made us famous.

Even though many thousands of readers have indicated that they both enjoy and learn from the type-in-and-save process, we know that there are many more of you who don't have the time for it. *Home Computer Magazine* has a practical solution: You may order all of an issue's programs for your particular computer ON TAPE™ or ON DISK™ for an unbelievably low price—less than $4.

With our exclusive, in-depth coverage of Apple, Commodore, IBM and Texas Instruments, each issue of *Home Computer Magazine* is really *four* system-specific magazines in one wrapper—harnessing the power of "synergy" to make the value of the *whole* greater than the sum of its component parts. This unique approach allows readers to better understand their own computers through comparison to other machines. The contents of each issue is carefully balanced to deliver the proverbial "something for everyone"—translated for users of all four computer brands—whether absolute beginner or seasoned pro.

For our old as well as many new readers, I expect the coming years to be the best of times—the best of times to own a computer, and to learn, play, experience, and explore the magical world of home computing with your one essential resource, *Home Computer Magazine*.

So stay with us; there's an exciting time ahead.



Group Grapevine

News, information and upcoming events of home computer users groups around the world.

Looking to join a users group, exchange newsletters or software, increase your users group's membership or pep up your next meeting's agenda? For the latest users group news, put your ear to the Group Grapevine. And if you have a message to put out to other groups, if you are starting a new group, or have an interesting item to share, send a note or picture—or better yet, a group newsletter—to the Users Group Editor, Home Computer Magazine, 1500 Valley Drive, Suite 250, Eugene, OR 97401, (503) 485-8796

The phone lines and electronic bulletin boards have been buzzing as users groups around the world speculate, compare notes, and try to get more information on the fate of their machines. Groups responded to TI's announcement in various ways. Some called emergency meetings, others manned TI product availability hotlines. ("Ten consoles at $49.95 spotted at J.C. Penney. Unconfirmed report of peripheral expansion box at K-Mart.") Others went out into the streets to offer assistance to the droves of potential new TI-owners eager to get in on the "deal of the century." Everyone we talked to, however, agreed upon one thing: Now, more than ever, users groups will play a major role in user support.

Right now, the main quest is for information. To fill that need, Don Veith of TEX-BUG is forming an organization of users group presidents in the Western U.S., Western Canada, and Australia. The **Western Users Group Association (WRUGA)** will serve as a clearinghouse for information—to stop some of those wild rumors that are circulating, provide information on third-party releases, and more. To join the group, contact Don at 3535 So. H St. #93, Bakersfield, CA 93304. Hopefully, an enterprising easterner will start a similar group for the eastern half of the TI world.

Many of the new user groups starting up are hungry for basic information—how to go about starting up a group, writing by-laws, contacting prospective members, etc. There is good news for you brand new groups: TI's excellent start-up packet is still available. You can receive it by calling the toll-free TI CARES number (800) 842-2737. If you can't get through at first, don't despair...remember, lots of people are phoning that line.

Meanwhile, our letters and calls tell us that record numbers of TI users are looking for groups to join, and starting up new groups of their own. The influx of new owners eager for basic information and old-timers hungry for product news means more and bigger users groups all over the world.

Among the new groups we have heard from is the **Nittany Users of TI (NUTI)** in Pennsylvania. They are starting the first TI organization in what was, heretofore, exclusively Apple country. Interested parties in that neck of the woods can contact Linda Becker at 625 Berkshire Dr., State College, PA. A group has also sprung up in the streets of El Paso. To get in touch with the **Sun City 99/4A Computer Club of El Paso**, contact their president, Michael Elsner, in care of the Sun City Computer Club, P.O. Box 6966, El Paso, TX 79906.

The Brevard Users Group a new group located near the Kennedy Space Center, sports a space shuttle logo on their newsletter. The club owns three TI systems, a software library of 100+ programs, and promises programming classes, stimulating guest speakers, low dues, group software/hardware discounts, and a good time. So, for a good time, write B.U.G., P.O. Box 1402, Palm Bay, FL 32906.

Anyone who has ever felt himself at the mercy of a capricious, all-powerful Home Computer will appreciate the **Salt Lake and Valley 99'er Users Group's** acronym. Members of SLaVe (who refer to themselves as SLaVes) sent us a copy of their newsletter this month. Among the interesting tips inside was a subroutine to let those of you with Speech Synthesizers and Terminal Emulators list your programs with speech. This sounds great for double-checking those long, type-in-and-run listings without getting terminal eyestrain. Potential SLaVes can contact the group through Brent Case at 1874 West Homestead Farms #4, West Valley, UT 84119, or (801) 973-8480.

The Holidays have meant increased activity for Commodore groups. What with eager shoppers and holiday sales, it seemed the perfect time to exchange an old piece of software or machine accessory for that shiny new peripheral you've been coveting. The **Commodore Houston Users Group** (CHUG) had a bright idea for some holiday shopping and swapping. They held a swap meet to let their members buy, sell, or trade software and hardware. Sounds like a painless way to cut down on that last minute desperation shopping, trade in that well-meant (but all wrong) birthday present, and start the new year with some inspiring new software. CHUG has been in existence for about two years and boasts about 500 members (we didn't ask how many show up for meetings). According to group spokesman, John Walker, they are an informal group who prefer talking with one another to bringing in outside speakers. They have held BASIC, hardware, and Tech Talk clinics to focus on topics of special interest. You can contact this group at 8738 Wildforest, Houston, TX 77088, (713) 999-3650.

The **VIC Commodore User Club** in Mariesville, MI, was pleased to receive a $50 donation from the Mariesville Police Department at their last meeting. The reason? The group recently developed a program to help their men in blue keep track of scores and divisions at a police target shooting meet. An ultra-quick sort subroutine was used to provide instant information on past and present shooting scores. The Mariesville group is one of that rare breed of hybrid organizations—users groups who welcome any member regardless of machine type. Acording to M. Gautier, one of their more active members, the multi-machine mix works out surprisingly well. You can reach this polyglot group at 486 Michigan Ave. Mariesville, MI 48040, (313) 364-6804.

The mammoth **Toronto Commodore User Group** is already making plans for their annual conference, slated for May 18. For the few Commodore people who have never heard of this 10,000 member group, we should mention their gargantuan software library (containing some 5000-6000 programs) which any group or individual can join. Library Members receive monthly diskettes of public domain software. Anyone interested in joining either the library or group can contact Chris Bennet at 381 Lawrence Ave. West, Toronto, Ontario, Canada, M5M 1B9, (416) 782-9252.

Less huge, but equally active groups have also been busy this month. The **Napa Valley Commodore Computer Club** got together to take a look at the VIC Tree cartridge and Easy Script Word Processing Package . You can contact this group at 2680 Jefferson St. Napa, CA 94558, (707) 252-6281. In Virginia, the **Peninsula Commodore 64 Users Group** were treated to an impressive graphics demonstration. A commercial artist showed how the Doodle program could be used with a track ball to move lines on the screen in a manner similar to a light pen or artist's brush. According to Richard Wilmoth, the combination of artistic talent and technical innovation made for some dazzling displays. For more information on the group, contact him at 124 Burnham Place, Newport News, VA 23606, (804) 595-7315.

One of our newer groups, the 6-month-old **Commodore User Group** in Santa Cruz, CA, met this month to watch a local computer dealer demonstrate how to clean your disk drive. Nothing like bringing in an expert to get the inside track on proper maintenance. You can contact the group through Bud Massey at 2301 Mission St., Santa Cruz, CA 95060, (408) 425-8054.

It may not seem like much of a problem, but some of the larger Apple groups seem to be finding themselves with a surplus of money these days. When you think about it, a group of 1000 that charges $28 per year in dues is pulling in a healthy income of $28,000 yearly. Add to that the extra revenue from diskette sales, etc., and you can understand why these big Apple groups have something of a reverse cash flow problem. What to do? Of course, you can just let the money sit there or print a 4-color newsletter, but most groups prefer to put their money to use to benefit the members.

The **Dallas Apple Corps** is one of the more well-heeled groups that are considering some exciting new ways to put their cash to work. One radical new idea that has been suggested is paying the officers a mini-salary. While some may balk at the idea, consider all the time your program chairman puts into planning each meeting, wheedling people to make presentations and lining up speakers. And what about the poor soul in charge of the newsletter—when was the last time you sent in some unsolicited copy to help him out? A small salary would be a just reward for these hard-working people who really keep the group going, and it would offer incentive to improve performance—chances are a lot more people would be interested in a paid position. . . and members could expect first-rate performance from their newly commisioned officers.

Another option for the wealthy group is paying to bring in speakers—why not spring for a plane ticket and bring in Steve Wozniak to explain, once and for all, how to edit program lines? Or how about a more concrete investment? The Dallas Apple Corps has already put some of their surplus cash to good use by purchasing some equipment—2 monitors and a 10-foot diagonal projector. These lone-star Apple barons are as active as they are prosperous: They have special interest groups on business and personal software, spreadsheets, advanced topics, the C language, advanced Applesoft, beginning on the Apple, Pascal, CP/M, education, and science & technology. At their last meeting, the Dallas Apple Corps people enjoyed a presentation on a 68000 board (with a quarter megabyte) for the Apple. You can reach this dynamic group in care of Apple Corps of Dallas, P.O. Box 5537, Richardson, TX 75080.

Education is a strong interest for the **Birmingham Apple Corps** members. It was such an important topic for them that they started a spin-off group, **Apple Blossom**, which is exclusively for Apple-oriented educators. That little offshoot now numbers over 100 members. The original group still retains an educational bent, however, providing regular classes in BASIC and assembly language. At their last meeting, the group discussed the Z-80 chip and listened to a representative from 3M discuss the properties of different disks. You can reach the Birmingham Apple Corps at P.O. Box 55421, Birmingham, AL 35255.

The **Mini'App'les** (in Minneapolis, of course) were treated to a presentation on artificial intelligence by two professors from the University of Minnesota. This large conglomerate of small Mineapolis groups includes special interest groups on Pascal, business applications, investment applications, education, medical applications, Visi-Calc, CP/M, telecommunications, and applications for the handicapped. You can contact the group by writing to Mini'App'les, P.O. Box 796, Hopkins, MN 55343.

Users groups for the PCjr? Not yet, but if the **IBM PC Users Group** in Eugene, Oregon is any indication, the PCjr will find a warm spot waiting in the hearts of PC groups across the country. At their last meeting, Rocky Mertin, a representative from IBM, showed slides and answered questions about this long-awaited addition to the IBM family. One thing is for sure, the interest in this new machine is high, and all signs point to a rash of "extended family" PC groups, spin-offs, and brand new groups exclusively for the new PCjr home user. We hope to hear from these new groups as soon as they spring up—and we'll be closely covering their rapid progress in each monthly installment of Group Grapevine. HCM

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 with:

- Typeset listings with numbers in boldface type.
- 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) to 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

Before You Begin

Since *HCM* publishes programs 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 Series**
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 I for the number 1, the letter S for the $, and the uppercase B for the number 8. This last error is especially likely when working in hexadecimal numbers which are composed of 0-9 and the uppercase letters A-F.

The listings in *HCM* are always the same number of characters wide, but, the number of characters put on any line of the video display will vary from computer to computer. Don't try to make your listing *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 Listings for the C-64 and VIC-20

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.

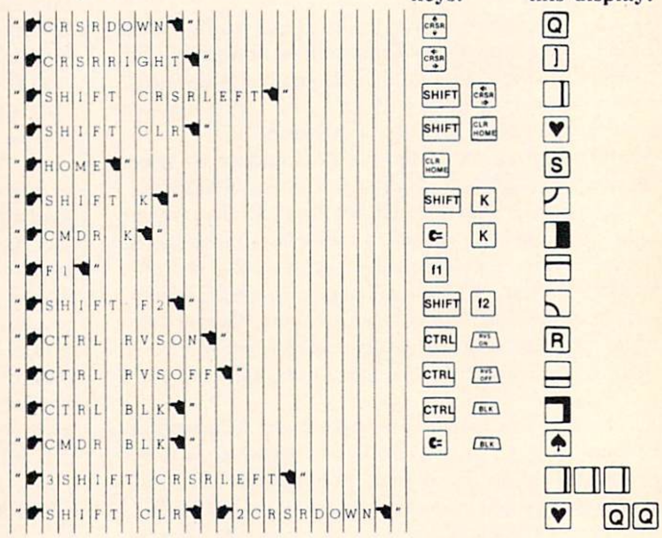
When you come to the hand symbols, remember:

- 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.
- Not to enter spaces or anything else *within* a set of hands.
- Everything between a pair of hand symbols is set in a different typeface.

In Figure 2 below, we have included a chart showing you a representative sample of the symbols that appear when you use keystrokes enclosed by the hand symbols.

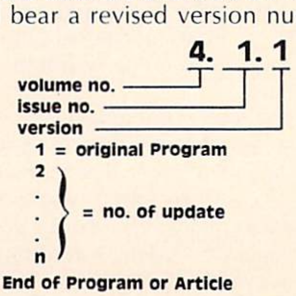
Figure 2: SPECIAL SYMBOLS CHART FOR C-64 and VIC 20

When you see: | **Press the keys:** | **To get this display:**



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.



HCM

HOME COMPUTER magazine

Home Computer Magazine is published monthly by Emerald Valley Publishing Co., P.O. Box 5537, Eugene, OR 97405. 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 $7 per year. Other foreign countries $43 for one year surface mail. Inquire for air delivery. Single copy price in U.S. and its possessions is $3.00, and $3.50 in Canada and Mexico. Foreign subscription payment should be in United States funds drawn on a U.S. bank. Application to mail at second-class postage rates is pending at Eugene, OR 97401.

POSTMASTER: Send all address changes to **Home Computer Magazine**, P. O. Box 5537, Eugene, OR 97405. 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 "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 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.

Outside HCM

The airbrush artistry of Eric Martin depicts our commitment to the future. A computer that begins humbly on a design grid may rise rapidly to the top in this dynamic industry. Four home computers stand out on the high-tech horizon. In *Home Computer Magazine* we chart their progress, reflect the evolution of the industry, and interpret how these new developments and applications will affect you, the reader.

Inside HCM

This month we bring you a home computer concert—written in four-part harmony. Center stage, the principal players are warming up for a performance you can enjoy in your own home. The program has been specially selected to expand your home computer repertoire.

To guide you through this issue, here are a few notes on a musical theme. *Uncle Larry's Fiddle Tunes* will have you toe-tappin' in time to the familiar strains of "Arkansas Traveler" and nine other fiddlin' favorites. Go ahead and hum along as you key in the complete BASIC program listing for your own computer.

With these country classics still ringing in your ears, head back (not Bach) in time, to discover Professor Holl's *Pocket Canon*, an electronic version of Pachelbel's Canon in D. And if you have a more current beat in mind, we have two programs for budding composers. *I Write the Songs*, in Extended BASIC, and *Just Assemble Melody*, in Assembly Language, let you write your own music and see it displayed on the screen. If you are still a bit intimidated by Assembly Language, be sure to check out part 3 of our tutorial, *Have No Fear: Assembly Language Won't Byte*.

Another language makes a command performance in *Lyrical LOGO*. Your computer can become a poet, lyricist, or just a total LOGO-phile. Or it can help you build skyscrapers in *What Is LOGO?* Then LOGO blasts off in *LOGO Shoots for the Moon*. The tips in our latest *TI-WRITER Tutorial* will help you record your lunar impressions with snappy word processing techniques to speed up and simplify the literary process.

Back on Earth, a symphony of applications awaits us. In *Porsches and Other Pipedreams*, you will find new ways to save money for that dream purchase—a Steinway, or a super stereo . . .

Making music isn't the only way to use notes and measures — menu planners use them too. In *Moveable Feasts*, you can plan a menu that makes a banquet of ordinary leftovers.

Apple devotees can get right to the core of the matter with three pieces written especially for them. *Biting Into Your Apple* is a theme and variations approach to home computing. Elaborate graphics programming is made simple in *Easy As Apple Pie*, and *3D-IIe* plots and rotates three-dimensional graphics on your screen.

Those who believe in a more hands-on approach to graphics will enjoy *Art at Your Fingertips*—our triple-treat review (with programs) of a powerful graphics system.

Next, strike up the band with a parade of tutorials. We lead off with *66 Keys to Graphics Success*, which explains how your first steps with your Commodore 64 or VIC-20 can create some instantly impressive graphics.

For those with a more spritely disposition, there are two special features to perk up your programs. *Don't Be a SlowPOKE* speeds up sprite routines, and *Down Memory Lane* will help you maximize character design with a minimal use of BASIC memory. And bringing up the rear, *Simon's Basic* and the *Super Expander 64* team up in *Bigger Better BASIC* to give you a wider range of commands than ever before.

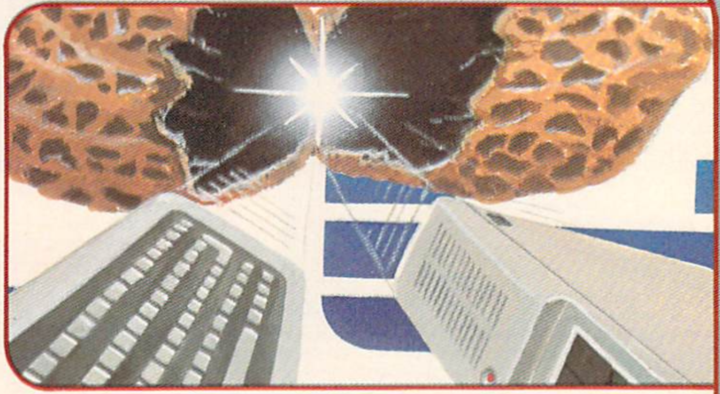
The IBM PCjr is featured in its own 12-page solo section. *A Detailed Look Inside the Peanut's Shell* provides extensive, never-before-published reference information on this long-awaited micro.

Our orchestral finale comes in a flourish of games for all instruments. TI users can try their luck at *Slots* or save the world in *Meltdown*. And for game lovers of all machine persuasions, *Flak Attack* and *Tower of Hanoi* are certain to be challenges.

So strike the keys and join the chorus! With *Home Computer Magazine* as your conductor, you'll find the melody lingers on and on and on.

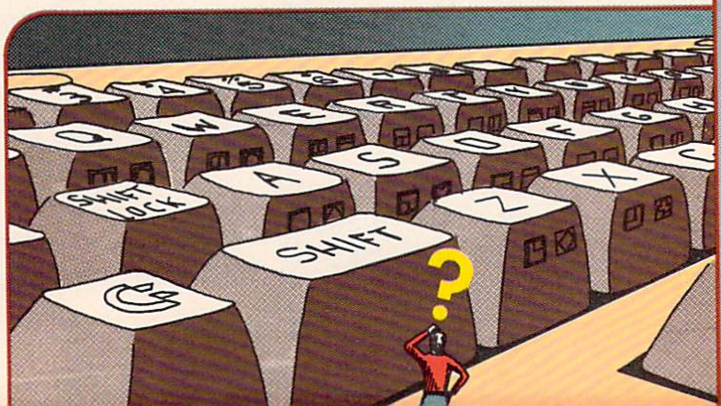
Until next month, have fun reading, learning and RUNing HCM

HOME COMPUTER magazine™



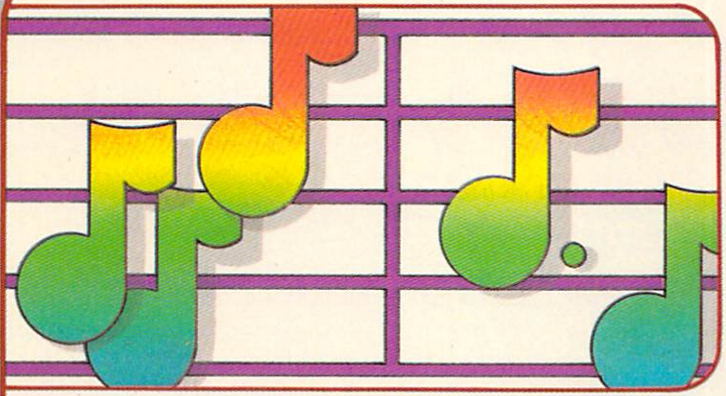
FEATURES

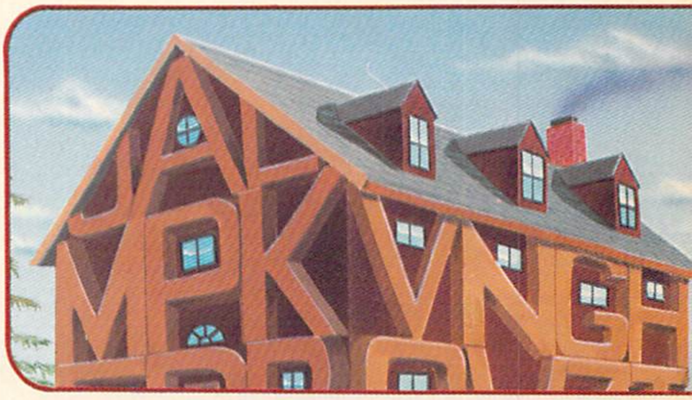
LOGO TIMES™



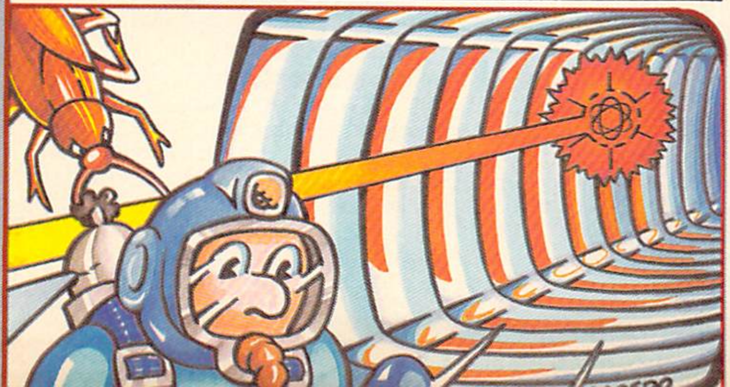
CONTENTS

VOLUME 4 NUMBER 1





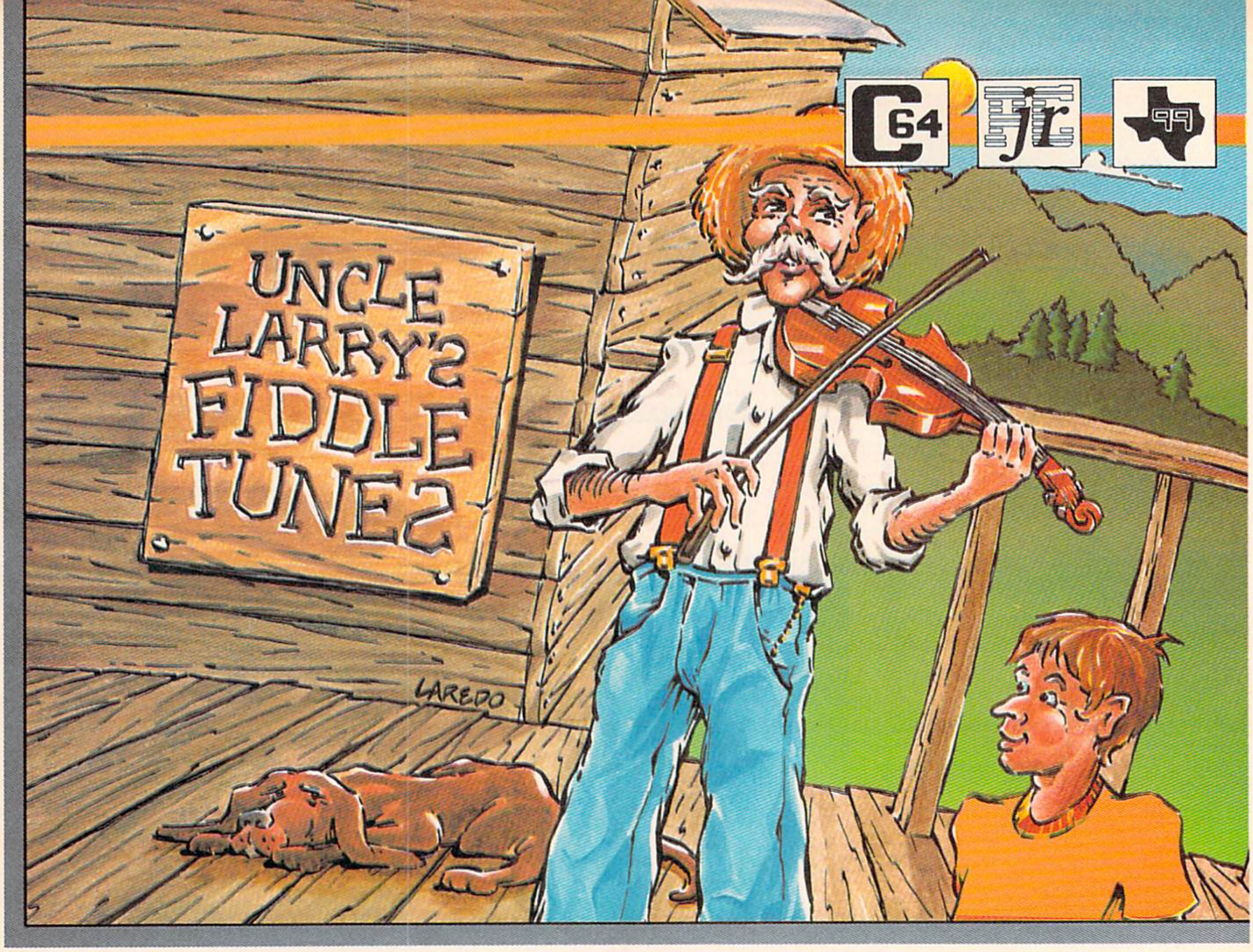
PRODUCT REVIEWS



GAMEWARE BUFFET

DEPARTMENTS

HOME COMPUTER DIGEST™
News and Happenings in the Home Computer World
Bound in between Pages 98 & 99



by J. Larry Schott
and the HCM Staff

There's an inheritance I received from a favorite relative that I'd like to share with you through the Home Computer. When I was a kid in Kentucky, we always looked forward to family gatherings with Great-Uncle Larry. Wearing a white starched shirt, this ancient and weathered farmer would ceremoniously tune up his old fiddle, and standing ramrod straight on our shady porch, he'd play and play. As I followed the melodies flowing from his coarse hands, veined as the tobacco he tended, I picked up those traditional tunes. Twenty years later, thinking about long-gone Uncle Larry late one night in the glow of my monitor, it dawned on me what I could write with the programming skills I had learned. I set about transcribing those songs he left me, feeling that the tough part of this program was behind me. For me, figuring out *how* to program is easy compared to deciding *what* to program.

You'll recognize these melodies from animated cartoons, old Civil War movies, even commercials (if not folk festivals). Though they were passed to me on a fiddle, you can pick them out on any instrument. Mandolins are tuned exactly the same way as fiddles, flutes have the same range as fiddles, and even guitarists like David Bromberg perform fiddle tunes. The computer sounds a lot like a clarinet when it plays them. This software is flexible, so you can use it as you wish. *Larry's Ten Fiddle Tunes* can be slowed down for instruction on any instrument; it can be your partner while playing a duet; or it can simply be run solo for some entertaining examples of computer music. You can also borrow a selection for your own programs; these songs are so time-worn that they are public domain.

How exactly did I cram ten traditional fiddle tunes into a single Home Computer program in less than 16K? Read on.

Two things about all the music programs I had typed on my TI computer frustrated me. One was the endless repetition of typing CALL SOUND on every line. The other was that after all that time at the keyboard, the reward of a few seconds of music seemed too small. This program addresses both of those problems.

The first solution grew out of a hint that a friend of mine, Dave Seymour, gave me. Dave has been involved with computers since he helped solder together one of the first ones, the ENIAC, in Philadelphia in the 1940's. His golden advice: "Anything you repeat a lot, you can probably put into a loop. Let the computer do the hack work. That's what they're good for." Thus developed my plan to use only one CALL SOUND statement containing variables and continually read DATA into it. That insight was probably reinventing the wheel, as they say, but as a neophyte computer user, I was pleased with myself for thinking of it. I believe this is the first mention of such a coding system in *HCM*.

The second problem was solved by the same strategy. Since DATA takes up less room than interminable commands, the loop opened up room for more music. Six, then eight, and finally ten tunes fit into a single program. That seemed too good to be possible. However, there is a corollary to Murphy's Law: Once you solve one problem, another shows up. By filling the available RAM with program, you eliminate much of the room for processing the DATA. This causes the music to "hiccup"—i.e., pause while the computer does its internal housekeeping—throwing the tune slightly out of rhythm. [Because the program will play in either BASIC or Extended BASIC, it can face different system configurations, and it reacts differently on those configurations. It hiccups *most* when running in BASIC with the disk system attached; entering CALL FILES (1) before running the program deletes some of the disk buffers and gives the system more workspace. In BASIC with the cassette system attached, or in

Extended BASIC with the disk system attached, it runs better. The programs run best in Extended BASIC with memory expansion attached.—Ed.]

Despite this almost human imperfection (which actually sounds a lot like the way Uncle Larry played), you can play along with the computer. For those who want to accompany the fiddling TI on rhythm guitar or whatever, the musical key and the starting note for each song are printed at the bottom of the screen. All the fiddle tunes are easy to follow with two or three chords once you have an idea where to start.

Special Program Features

As I mentioned, only one CALL SOUND statement—line 2280—plays every bit of the music. Here's how the play routine works. Line 2220 checks for the end of the melody line flag (99), and if it finds the flag, the program returns to the tune's subroutine. The next line checks to see if the variable D it just read is really a duration (less than 110) or is instead a frequency (necessarily over 110). If it is a duration, the program reads the next DATA as the frequency—line 2240—and goes right to the CALL SOUND to play the note. If not, the frequency is set equal to that D, the duration is assumed to be 1, and *then* the note is played. This process is used because the overwhelming number of notes are duration 1. One little routine saves typing the duration for all those notes. The 1 can be thought of as a sixteenth note, 2 is a quarter note, and so on. The decimal .7 roughly approximates a note of a triplet. It's actually a little long, but it sounds right.

The actual speed is determined by multiplying the duration by the variable SP. You can permanently set the value of SP in line 200 at the beginning of the program, or you can adjust it with menu choice (12). The suggested lower parameter of 50 is not locked into the program, and you can go as low as 2 without blowing it up. However, because of the time needed to read and run through the routine, all values below 50 sound pretty much alike—rather like the mechanical rock group DEVO. Hiccups show up less at slower speeds, which are also better for learning the tunes.

All other possible input errors are locked out of the program at lines 560 and 1590, making the program very user-friendly while running. I also designed the program to be user-friendly during the long, boring, mistake-prone process of typing in DATA. First of all, each DATA section of melody is clearly marked by a REM with the initials of the song. Each second melody line is additionally identified by a 2. For example, SJ is the first line on "Soldier's Joy"; SJ2 is the second line. This should make it easy to find the mistyped DATA if something sounds wrong or the program stops in the play routine.

To make it less likely that you have a typo to begin with, each line of DATA contains *sixteen* numbers. If there is an odd amount of DATA, there will be a few numbers right before the next REM statement. Even when the remainder was just one number, the end of section flag 99, I gave the flag its own line to maintain consistency and make typing in the DATA easy. As additional insurance against accidentally changing something in the DATA (I suspect it has happened), only the following numbers appear in these songs: durations .5,.6,1.5,2,3; frequencies from 220 to 1175 (all are numbers from the note chart in

Continued on p. 14

The one factor that sets the IBM PCjr version of this program apart from the others is the way in which we create the sound. The PCjr uses the Microsoft BASIC PLAY command, but has the ability to "PLAY" with three voices. The PLAY command allows you to compose music with notes and scales, rather than having to figure out the frequencies. The music is composed by building a string of subcommands for the PLAY statement to execute. You can designate a note with the letters A through F, and even designate sharps with the # sign. The L command is used to set the length of the note. For example L8 would make the music play eighth notes, while L4 would play quarter notes. The T command sets the tempo of the music in beats per minute. For example T120 would play 120 beats per minute which is about normal, while T40 would play at 40 beats per minute and be three times slower.

There are eight octaves available in the range of the sound chip. [This is the same Texas Instruments sound chip found in the TI-99/4A—Ed.] There are two ways you can specify the octave you are using. The first way should always be one of the first commands in your PLAY command string. The O command, followed by a number from 0 to 7, will select an octave for all of the following notes to be played at until the octave is changed. You should always place this command before any notes are played in each string so the computer will know which octave to use.

The second way to change the octave is with the < (less than) and > (greater than) signs. The < sign lowers the octave and the > raises it. The following string could be interpreted like this:

"O2 BAB>BAB<BAB>>DEF#"

The octave is set to 2 and the notes B, A and B are played. The octave is then raised to 3 and the notes are repeated. The octave is then lowered back to 2 and the same notes are played again. The octave is then raised to 4 with the >> and the notes D, E, and F# (F sharp) are played.

The last command used in this program is P. This command causes a pause of one beat. —*W. K. Balthrop*

Continued on p. 16

One of the subroutines in *Larry's Ten Fiddle Tunes* can be adapted for your own programs when you want them to play music.

In order to play a note on the Commodore 64 we must first look up the two corresponding values to be POKEd (or moved) into two registers of the sound chip to generate a particular note. These values can be found in the table of musical notes in the Appendix of the *Commodore 64 User's Manual*. The two registers are actually combined to form one sixteen-bit number that corresponds to the frequency of the note. If we want a succession of different notes (as in a melody), we need a way to input these POKE values to play the tune. We could put all the necessary POKE values directly into DATA statements, read them in and POKE them into the sound chip, but there is a better way.

It is often more practical to deal with musical notes by relating each note to its corresponding audio frequency rather than to a high or low POKE value that is meaningless outside the world of the Commodore 64. This will also shorten the DATA statements since we will have only one number instead of two for each note. *Larry's Ten Fiddle Tunes* uses a method based on these practical considerations to generate the notes for each of the ten tunes. If, for example, you want to play the A above middle C, first find its corresponding audio frequency (440 hertz) and put this into a DATA statement along with frequencies for the other notes to be played in sequence by the Play Note subroutine.

The Play Note subroutine will do several chores for us. First it calculates the corresponding high and low POKE values; then it sets up all the other parameters associated with playing a note on the Commodore 64. Next it turns the sound chip on and goes into a FOR-NEXT loop for a designated interval (equivalent to the duration of the note). Then it turns the sound chip off and returns to the calling program. The Play Note subroutine starts on line 8600 and returns on line 8660. Before using this routine we must set the variable G equal to 54272 (the beginning address of the sound chip registers). It is also a good practice to clear all of the sound chip registers whenever a new type of tone is used. Lines 270 and 280 perform these two steps.

Continued on p. 15

Uncle Larry

the instruction manual); flag 99, which signals the end of a musical phrase; and the frequency 22222, used as a musical rest since it is too high to hear and easy to type.

Another innovation to help spot problems during debugging is the spread of DATA lines throughout the subroutines as remarks. These DATA lines are used to print the menu each time, in lines 290-500. However, their placement in lines 630-1440 indicates which subroutine controls which song. This was done to keep from entering redundant REM statements. I wanted to document the program thoroughly but I hated to waste RAM.

The second part of each of these DATA lines (everything after the comma) is stored in the array A$. This information is printed as M$ after the menu selection is made, displaying the musical key and the first note and string number. To help you understand the rest of the program, the variables are obvious: **D**uration, **F**requency, and **SP**eed in the CALL SOUND; **R**ow and **C**olumn (think of **C** as a TAB) in the PRINT M$ (any message); **N**ote (the ASCII of G, D, A or E for tuning), **T$** to pull title information out of array A$, and the ubiquitous all-purpose counter **X**. The **X** counts characters printed during the PRINT routine and keeps track of where the program is during the play routine. Since it never prints and plays at the same time, the same counter variable can be reused for each section.

Extended BASIC

The program was originally written in Extended BASIC, but I broke it down to plain vanilla TI BASIC to reach the widest possible audience. Some of you will want to change it back. In Extended BASIC, the menu was written very differently, taking advantage of DISPLAY AT and ACCEPT AT. Though that code was much shorter, the display was more static without the drama of scrolling the menu each time. You might want to make cosmetic changes if you have Extended BASIC. Stacked statements in Extended make the whole program much neater; for instance:

```
650 RESTORE 2340 : : GOSUB 2210 : : IF X<2 THEN 650
680 RESTORE 2410 : : GOSUB 2210 : : IF X<4 THEN 680 :
    :RETURN
```

Two stacked lines replace seven BASIC lines. This is a good place to see how each subroutine to play the tunes works. The format for each tune is traditionally the same—AABB—where the first line is played twice, followed by the second section twice. The counter X, incremented in line 2300, keeps track of where we are in that formula. The play routine can be dramatically condensed from eleven lines to three, as shown in lines 2210 to 2300:

LARRY'S TEN FIDDLE TUNES (TI)
Explanation of the Program

Line Nos.	
100-180	Program header.
190-250	Initialization.
260-280	Change color of lower case letters.
290-500	Print menu.
510-620	Select from menu; print pertinent information from array to bottom of screen.
630-1430	Ten subroutines for the ten songs.
1440	Print a blank line in menu for readability.
1450-1570	Draw head of instrument and strings for tuning utility.
1580-1610	Choose string to tune.
1620	Leave tuning section when 0 is pressed.
1630-1770	Note, frequency and row placement information for tuning peg.
1780	CALL SOUND for tuning (negative duration allows interrupt).
1790-1930	Draw peg to turn.
1940	Display letter of note on peg.
1950	Loop to stay in tuning routine.
1960-2030	Change playing speed.
2040-2140	End message.
2150-2190	Print message subroutine.
2200-2310	Play music subroutine.
2320-3340	DATA for music.

```
2210 READ D : : IF D=99 THEN 2300 ELSE IF D<110 THEN
     READ F ELSE F=D : : D=1
2280 CALL SOUND (INT(SP*D),F,1) : : GOTO 2210
2300 X=X+1 : : RETURN
```

The information in the tuning subroutine can be similarly compacted, and DISPLAY ATs can be used to draw the pegs and name the notes. You can also replace the whole PRINT M$ routine (lines 2150-2190) with a single DISPLAY AT statement. Of course, the most important and lengthy part of the program, the mass of DATA, is unchanged in the switch from BASIC to Extended BASIC. Although you could cram more numbers on each line, I believe the increased possibility of typing errors outweighs the savings.

Play it Again, Uncle Larry

So now you can put into your computer a part of my uncle and hundreds of other folk musicians stretching back through centuries. I'm not sure what my taciturn tutor would have said about my mechanical method of teaching you his tunes. All I know is that he always smiled when he played. These songs made him happy, and if passing them along in this new-fangled way makes any of you smile, then I guess he'd be in favor of it.

Now hit the electricity and rosin up the bow!

TI-99/4A

```
100 REM  *****************
110 REM  *   LARRY'S TEN    *
120 REM  *   FIDDLE TUNES   *
130 REM  *****************
140 REM  BY LARRY SCHOTT
150 REM  HOME COMPUTER MAGAZINE
160 REM  VERSION 4.1.1
170 REM  TI BASIC
180 REM
190 REM  INITIALIZATION
200 SP=175
210 CALL SCREEN(8)
220 CALL CHAR(33,"E7E7E7E7E7E7E7E7")
230 CALL CHAR(44,"FFFFFFFFFFFFFFFF")
240 CALL CHAR(95,"08080808868F8F060")
250 DIM A$(14)
260 FOR X=9 TO 11
270 CALL COLOR(X,5,1)
```

Continued on p. 30

C64 *Uncle Larry . . .* from p. 13

Now let's analyze the subroutine itself. The central segment of the program, lines 8600 to 8660, plays the notes for *Larry's Ten Fiddle Tunes*. Line 8600 will derive the high and low Poke values necessary to drive the sound chip in the correct tone:

8600 FD=INT(F/.06097) :HF=(FD/256) :LF=FD−(256*HF)

HF equals the high frequency; LF equals the low frequency. FD is an interim variable that is used only in the calculation. F equals the note frequency in hertz.

8610 POKE G+5,184:POKE G+6,169

sets the attack/decay and the sustain/release values to simulate the tone of a fiddle.

8620 POKE G+24,15

sets the volume level to 15 (maximum).

8630 POKE G+1,HF:POKE G,LF

POKEs the values derived by line 8600.

8640 POKE G+4,33

does two things: It generates a sawtooth wave form for voice 1, and it initiates the attack/decay cycle.

8650 FOR ZX=1 TO INT(SD*D): NEXT

determines the length of time the note will play. The two control variables are SD and D. The value of SD is constant throughout the program unless changed by Option 12 of the menu (Change Speed). The variable D is used to simulate note values, such as eighth notes, quarter notes, and so on.

8660 POKE G+4,32:RETURN

initiates the release cycle and effectively turns the sound chip off at a speed determined by the value poked into G+6 in line 8610. We also return from the subroutine here.

—John Thrasher

Continued on p. 174

Uncle Larry . . . **from p. 13**

LARRY'S TEN FIDDLE TUNES (IBM PCjr)
Explanation of the Program

Line nos.	
100-170	Program header.
180-230	Initialize the program, and display the title screen.
240-300	Display options page, and input option.
310-330	Branch to the subroutine chosen, then return to the menu screen.
340-530	Data for the menu screen, key, and first note. Also selects the data for each tune, and then branches to the subroutine to play that data.
540-690	Tune the fiddle.
700-810	Change the speed.
820-850	End the program.
860	Reads the data and plays the music.
870-880	Data for Soldier's Joy.
890-900	Data for Arkansas Traveler.
910-920	Data for Cincinnati Hornpipe.
930-940	Data for Pop Goes The Weasel.
950-960	Data for Garry Owen.
970-980	Data for Cock And Hen.
990-1000	Data for Tom And Jerry Reel.
1010-1020	Data for Irish Washerwoman.
1030-1040	Data for Mc Donald's.
1050-1060	Data for Two Forty Reel.
1070	End of the program.

IBM PCjr

```
100 REM ****************
110 REM * LARRY'S TEN   *
120 REM * FIDDLE TUNES  *
130 REM ****************
140 REM BY LARRY SCHOTT AND HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.1.1
170 REM IBM PCjr
180 DIM A(14):SP=8
190 CLS:LOCATE 1,1
200 BEEP OFF:SOUND ON
210 FOR X=1 TO 24:FOR Y=1 TO 40:PRINT C
    HRS(14);:NEXT Y:NEXT X
```

IBM PCjr

```
220 LOCATE 12,7:PRINT "  LARRY'S TEN FID
    DLE TUNES ":LOCATE 23,13:PRINT "   PR
    ESS RETURN ";
230 Z$=INKEY$:IF Z$="" THEN GOTO 230
240 CLS:LOCATE 1,1
250 PRINT "        LARRY'S TEN FIDDLE T
    UNES":PRINT:PRINT
260 RESTORE:FOR X=1 TO 13:READ T$,A$(X)
    :PRINT T$:NEXT X
270 LOCATE 23,1:INPUT "YOUR CHOICE (1-1
    3)";B$
280 IF B$="" OR B$=CHR$(13) THEN GOTO 2
    70
290 IF ASC(B$)<49 OR ASC(B$)>57 THEN GO
    TO 270
300 IF LEN(B$)>2 THEN GOTO 270 ELSE IF
    LEN(B$)=2 THEN IF ASC(MID$(B$,2,1))
    <48 OR ASC(MID$(B$,2,1))>51 THEN 27
    0
310 S=VAL(B$):LOCATE 18,1:PRINT "SELECT
    ION:    KEY    FIRST NOTE":LOCATE 19,1
    :PRINT S;TAB(14);A$(S)
320 ON S GOSUB 340,360,380,400,420,440,
    460,480,500,520,540,700,820
330 CLS:LOCATE 1,1:GOTO 250
340 DATA "  (1) SOLDIER'S JOY",D      F#
    ON 3
350 RESTORE 870:GOTO 860
360 DATA "  (2) ARKANSAS TRAVELER",D
      A ON 4
370 RESTORE 890:GOTO 860
380 DATA "  (3) CINCINNATI HORNPIPE",D
      D/OPEN 3
390 RESTORE 910:GOTO 860
400 DATA "  (4) POP GOES THE WEASEL",G
        G ON 3
410 RESTORE 930:GOTO 860
420 DATA "  (5) GARRY OWEN",G        G ON
    1
430 RESTORE 950:GOTO 860
440 DATA "  (6) COCK AND HEN",G        B O
    N 2
450 RESTORE 970:GOTO 860
460 DATA "  (7) TOM AND JERRY REEL",D
     A/OPEN 2
470 RESTORE 990:GOTO 860
480 DATA "  (8) IRISH WASHERWOMAN",G
       D ON 2
490 RESTORE 1010:GOTO 860
500 DATA "  (9) MC DONALD'S",G         B ON
     2
```

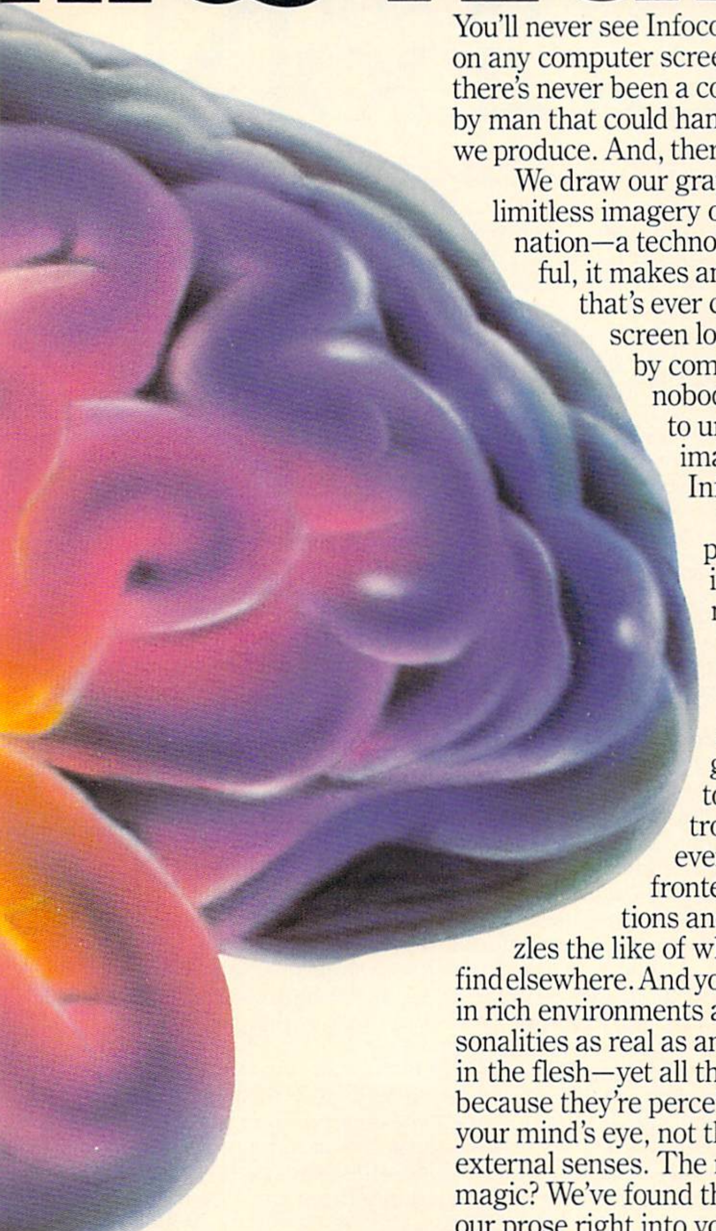
IBM PCjr

```
510 RESTORE 1030:GOTO 860
520 DATA "(10) TWO FORTY REEL",G    G/O
    PEN 3
530 RESTORE 1050:GOTO 860
540 DATA "(11) Tune the fiddle",""
550 SCREEN(1)
560 CLS:LOCATE 2,22:PRINT "3        2
    ":LOCATE 4,22:PRINT "4          1":
    LOCATE 21,1:PRINT "PRESS STRING NUM
    BER TO BE TUNED."
570 PRINT:PRINT "PRESS (0) TO RETURN TO
     THE MENU."
580 DRAW "BM 200,130;S8R3NU52R2NU58R2NU
    61R2NU55R3U46EU2E2U13L18D13F2D2FD46
    "
590 SC1$="R2ERD2LH":SC2$="L2HLD2RE"
600 DRAW "BM 192,26;XSC2$;BM 192,14;XSC
    2$;BM 232,20;XSC1$;BM 232,8;XSC1$;"
610 Z$=INKEY$:IF Z$="" OR Z$=CHR$(13) T
    HEN GOTO 610
620 IF Z$<"0" OR Z$>"4" THEN GOTO 610 E
    LSE Z=VAL(Z$)
630 IF Z=0 THEN CLS:LOCATE 1,1:GOTO 250
640 IF Z=1 THEN F=659:GOTO 680
650 IF Z=2 THEN F=440:GOTO 680
660 IF Z=3 THEN F=294:GOTO 680
670 IF Z=4 THEN F=196
680 SOUND F,32/SP
690 GOTO 610
700 DATA "(12) Change speed",""
710 CLS:LOCATE 1,1:PRINT "CHANGE PLAY S
    PEED (1 TO 64):"
720 PRINT:PRINT "SOME RECOMENDED VALUES
    :"
730 PRINT:PRINT " 1  VERY VERY SLOW"
740 PRINT:PRINT " 2  VERY SLOW"
750 PRINT:PRINT " 4  SLOW"
760 PRINT:PRINT " 8  NORMAL SPEED"
770 PRINT:PRINT "12  FAST"
780 PRINT:PRINT "16  VERY FAST"
790 PRINT:PRINT "24  VERY VERY FAST"
800 PRINT:PRINT:PRINT "ANY VALUE ABOVE
    THIS WILL BE HARD TO   UNDERSTAND
    MUCH LESS PLAY BACK."
810 PRINT:PRINT:INPUT "YOUR CHOICE";SP:
    CLS:LOCATE 1,1:GOTO 250
820 DATA "(13) End the program",""
830 P1$="T60 O3 D T171 O2 AABA;P=SP;":P
    2$=""
840 PLAY "XP1$;XP1$;":PLAY "V14 O2 T60
    A V10 T32 D","V15 O3 T60 C# T32 D"
850 CLS:GOTO 1070
860 READ P1$,P2$:PLAY "L=SP;V15;XP1$;XP
    1$;XP2$;XP2$;XP2$;XP2$;","L=SP;V12;
    XP1$;XP1$;XP2$;XP2$;XP2$;XP2$;","L=
    SP;V8;XP1$;XP1$;XP2$;XP2$;XP2$;XP2$
    ;":RETURN
870 DATA "T120 O2 F#GAF#DF#AF#DF# T60 A
    >DD T120 E<BAF#DF#AF#DF# T60 GEE T1
    20 F#GAF#DF#AF#DF# T60 A>DD T120 EG
    F#AF#DEGEC# T60 DDD T120"
880 DATA "O3 DEF#EDEF#AGF#EDC#<B T60 A>
    G T120 F#EDEF#GAF#EDC#DEF#GEF#AF#DE
    GEC# T60 DDD T120"
890 DATA "T120 O1 AB>C#DF#ED T60 <BBAA
    T32 >D T120 EE T60 E T120 F#F# T60
    F# T120 DF#ED T60 <BA T120 >DF#ED T
    60 <BBAA T32 >D T120 >DC#D<AB>D<AGF
    #EDC# T60 D T120"
900 DATA "O3 AGF#AGF#EGF#EDF#EDC#<A>DC#
    DF#EDEGF#EDF# T60 E T120 F#GAGF#AGF
    #EGF#EDF#EDC#<A>DCD<AB>D<AGF#EDC# T
    32 D T120"
910 DATA "T120 O2 DAF#ADAF#A>D<A>F#<A>E
    <A>F#<A>G<A>F#<A>EDC#DEDC#<BAGF#EDA
    F#ADAF#A>D<A>F#<A>E<A>F#<A>G<A>F#<A
    >EDC#<B>C#DEGF#D T60 D T120"
920 DATA "E O2 AAA>F#<AAA>G<AAA>F#<AAA>
    E<A>F#<A>G<A>F#<A>EDC#<BAGF#ED>DDDC
    #EEEDF#F#F#EGGGF#GAF#BGEC# T60 DD T
    40 D T120"
930 DATA "T60 O2 G T120 G T60 A T120 AB
    >D<B T60 G;P=SP;GGA T120 >C< T40 B
    T60 G;P=SP;G T120 G T60 A T120 AB>D
    <B T60 G;P=SP;T40 >E< T60 A T120 >C
    < T40 B T60 G"
940 DATA "O3 T40 G T60 E T120 GF#AF# T4
    0 D T60 G T120 G T60 E T120 G T40 F
    # T60 D T120 <B> T60 C T120 <B> T60
     C T120 D T60 E T120 F# T60 G;P=SP;
    T120 E T60 ;P=SP;<A>C T40 <B T60 G>
    "
950 DATA "O3 T40 GF# T120 EDC<BAGBEBB>G
    F#EDC<BAGABAA>GF#EDC<BAGB>C<B T60 B
     T120 >CDEF#GD<BABA T40 A T120"
960 DATA "T240 B O3 C T60 D T120 <B> T6
    0 D T120 <B> T60 D T120 <B>DGF# T60
     E T120 C T60 E T120 C T60 E T120 C
     T60 E T120 F# T60 G T120 A T60 B T
    120 AGF#EDC<B>DEF#GD<BAB T40 A T120
    "
970 DATA "O2 T120 B>C<BBAB T60 >GE<B>C<
    BBAB>DC<BB>C<BBAB T60 >G T120 E T60
     F# T120 D T60 E T120 <B>D<B T60 A
    T120"
```

Continued on p. 30

WE UNLEASH TH
POWERFUL GRAP

E WORLD'S MOST
HICS TECHNOLOGY.



You'll never see Infocom's graphics on any computer screen. Because there's never been a computer built by man that could handle the images we produce. And, there never will be.

We draw our graphics from the limitless imagery of your imagination—a technology so powerful, it makes any picture that's ever come out of a screen look like graffiti by comparison. And nobody knows how to unleash your imagination like Infocom.

Through our prose, your imagination makes you part of our stories, in control of what you do and where you go—yet unable to predict or control the course of events. You're confronted with situations and logical puzzles the like of which you won't find elsewhere. And you're immersed in rich environments alive with personalities as real as any you'll meet in the flesh—yet all the more vivid because they're perceived directly by your mind's eye, not through your external senses. The method to this magic? We've found the way to plug our prose right into your psyche, and catapult you into a whole new dimension.

Take some tough critics' words about our words. SOFTALK, for example, called ZORK® III's prose "far more graphic than any depiction yet achieved by an adventure with graphics." And the NEW YORK TIMES saw fit to print that our DEADLINE™ is "an amazing feat of programming." Even a journal as video-oriented as ELECTRONIC GAMES found Infocom prose to be such an eye-opener, they named one of our games their Best Adventure of 1983.

Better still, bring an Infocom game home with you. Discover firsthand why thousands upon thousands of discriminating game players keep turning everything we write into instantaneous bestsellers.

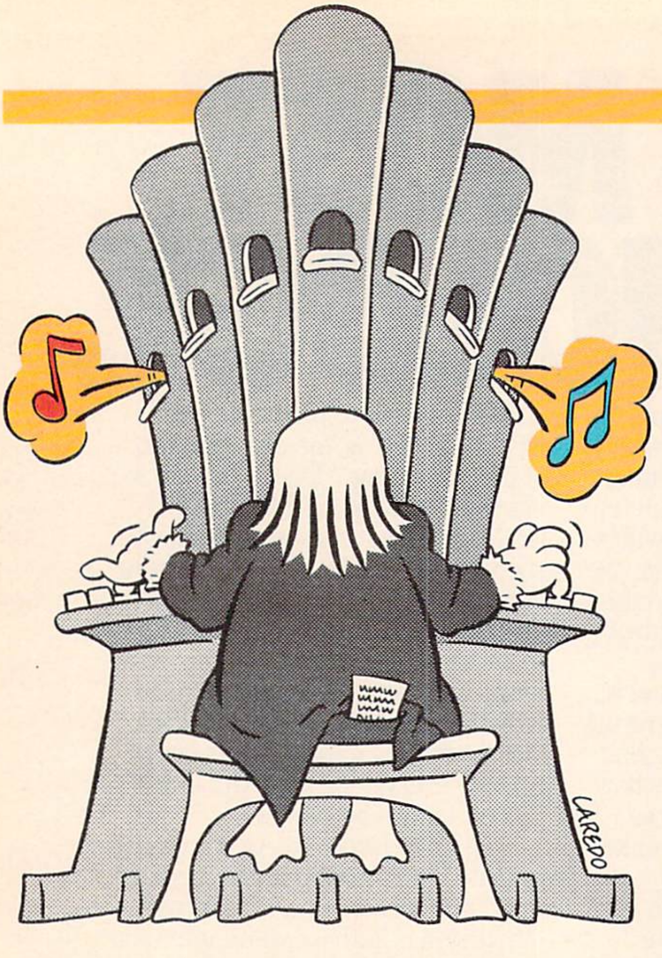
Step up to Infocom. All words. No graffiti. The secret reaches of your mind are beckoning. A whole new dimension is in there waiting for you.

(For more information on Infocom games contact: Infocom, Inc., P.O. Box 855, Garden City, NY 11530.)

INFOCOM™

The next dimension.

For your: Apple II, Atari, Commodore 64, CP/M 8,* DEC Rainbow, DEC RT-11, IBM, MS-DOS 2.0, NEC APC, NEC PC-8000, Osborne, TI Professional, TI 99/4A, TRS-80 Model I, TRS-80 Model III.



PROFESSOR HOLL'S

Pocket CanoN

by S. T. Holl

Be seated at the keyboard, Maestro,and play this little composition scored for the TI-99/4A computer:

```
100 REM  ***************
110 REM  *  POCKET CANON *
120 REM  *  BY PACHELBEL *
130 REM  ***************
140 REM  BY S. T. HOLL
150 REM  HOME COMPUTER MAGAZINE
160 REM  VERSION 4.1.1
170 REM  TI BASIC
180 REM
190 DIM F(7)
200 DATA 0,0,262,196,220,165,175,131,17
    5,196
210 READ N,V,F(0),F(1),F(2),F(3),F(4),F
    (5),F(6),F(7)
220 DEF M(X)=INT(X)-8*INT(X/8)
230 CALL SOUND(500,F(M(N/4)),3,1.5*F(M(
    N/2)),30+29*(V>31),2*F(M(N)),30+30*
    (V>63))
240 N=N+1+32*(N>31)
250 V=V+1
260 GO TO 230
```

If this doesn't work the first time you run it, and the error message is INCORRECT STATEMENT IN 230, then check line 230 carefully. Did you notice that there are two closing parentheses at the end of the line? How about that single period—not comma—in 1.5? If the error message is DATA ERROR IN 210, be sure to check line 200 too—it contains ten entries separated by commas; because some of these entries are duplicates of others, it is not difficult to lose your place when typing them in.

Now that you have the program running, let us discuss what it is you are hearing. A *canon** is a musical form in which the melody of the leading voice is imitated in some fashion by the other voice or voices. The imitating voices can be at the same tempo and pitch as the leader, or they can vary either or both. When you started this program, you heard the theme alone, played very slowly by the first voice; then the second voice joined the first, playing the theme twice as fast and a perfect fifth higher; and finally the third voice chimed in four times as fast and a full octave higher than the lead. A *round* like "Frere Jacques" or "Row, Row, Row Your Boat" is a simple canon in which the imitating voices use the same tempo and pitch as the lead but start at different times.

Now, how did we pack all that into eight program lines? The melody itself is stored in the third through tenth entries in the data statement. You could substitute your own eight-note theme by replacing those numbers with new ones you have looked up in the musical frequencies table on page III-7 of your *User's Reference Guide*. This particular theme comes from a celebrated canon of Johann Pachelbel. His harmonization is different—more complex, and much more beautiful—from our childhood rounds. Pachelbel's Canon in D is the perfect accompaniment for golden sunrises, but well worth the hearing any time.

RCA Vector

Line 210 initiates two counters we'll need later, and then reads the frequencies of the notes into the vector F. You may think of a vector (or array) as a series of numbers all using the same name. To specify the first one we use F(0), the second one under that alias is F(1), and so forth. CAUTION: When using vectors, always declare them first with a DIM (for DIMension) statement which tells the machine things it needs to know to manage them properly.

The advantage of using a vector to store our tune is that we can use an expression like N/2 as the index to the vector(e.g., F(N/2)),and then when the value of the expression changes, it refers to a different value of the vector F.

Canons play themes over and over again. So we want to make an index for the F vector go from zero to seven over and over again in a natural manner as our program executes, and we would like to be able to do this at varying speeds.

The mechanisms which make this music box play Pachelbel's theme are compacted into three lines: numbers 220,230, and 240. Line 220 is a DEFine statement, which establishes a base 8 modulo function for use within this program. We all routinely use base 12 and base 60 modulo arithmetic when we do problems involving time. For example, to find out what time it will be 19 hours from now, we add 19 to the present time and then subtract twelves until the hour is 12 or under. Since there are 8 notes indexed 0 through 7 in this theme, we need a transformation which will turn any number into its modulo 8 equivalent, then round that down to an integer. Line 220 defines such a function, M( ). Try a few different values of X; you will see that this is true. Incidentally, this definition of M( ) works for negative numbers as well as positive ones. We'll not make use of that generality here, but it comes free, and if you pursue program-

*Ordnance enthusiasts who have managed to read this far without remarking about the single n in the title should turn immediately to the Gameware Buffet, where they will find the usual fine laser and photon torpedo smorgasbord.

ming, you're certain to need it again somewhere else. So jot down the function (or maybe this magazine issue and page) on a 3x5 card and file it under Modulo Arithmetic.

The purpose of line 240 is to increment by one the counter N each time it is executed until you reach 32. At this point N is reset to zero and the incrementing continues. The expression in parentheses is relational; during execution that expression is replaced with zero when it is false, with minus one when true.

Line 230 is the one actually making the tones. The first of its seven arguments is the duration of the sound. Five hundred milliseconds is about as short as the notes can be without producing gaps between them while the computer finishes the other things it needs to do in each iteration of the program. The remaining six arguments, separated by commas, are the frequencies and volumes of the three voices, in pairs. Volume can range from loud (zero) down to inaudible (30). The volume for the first voice, which is the one playing the slowest tune, is set at three. The volumes for voices two and three start at 30 but contain relational expressions which "turn up the volume" when the counter V (initially zero and incremented by one every iteration) reaches 32 and 64, respectively. This corresponds to one and two complete playings of the theme by the first voice.

Perfecting the Pitch

Now let us look at how the three frequency entries work. In line 230 they are:

first voice: F(M(N/4))
second voice: 1.5*F(M(N/2))
third voice: 2*F(M(N))

The array is loaded in line 210 with the frequencies of the notes of the theme in the DATA statement in line 200. Voice 1 uses those values of F directly. Voice 2 uses the same frequencies increased by 50 percent. If you try this for several of the frequencies listed on page III-7 of your manual, you will see that adding half again to the frequency will make it five whole notes higher. (There will, perhaps, be a difference of 1 in the last place due to roundoff in the table.) Voice 3 uses twice the frequencies in F. Can you tell from page III-7 what the effect of this is?

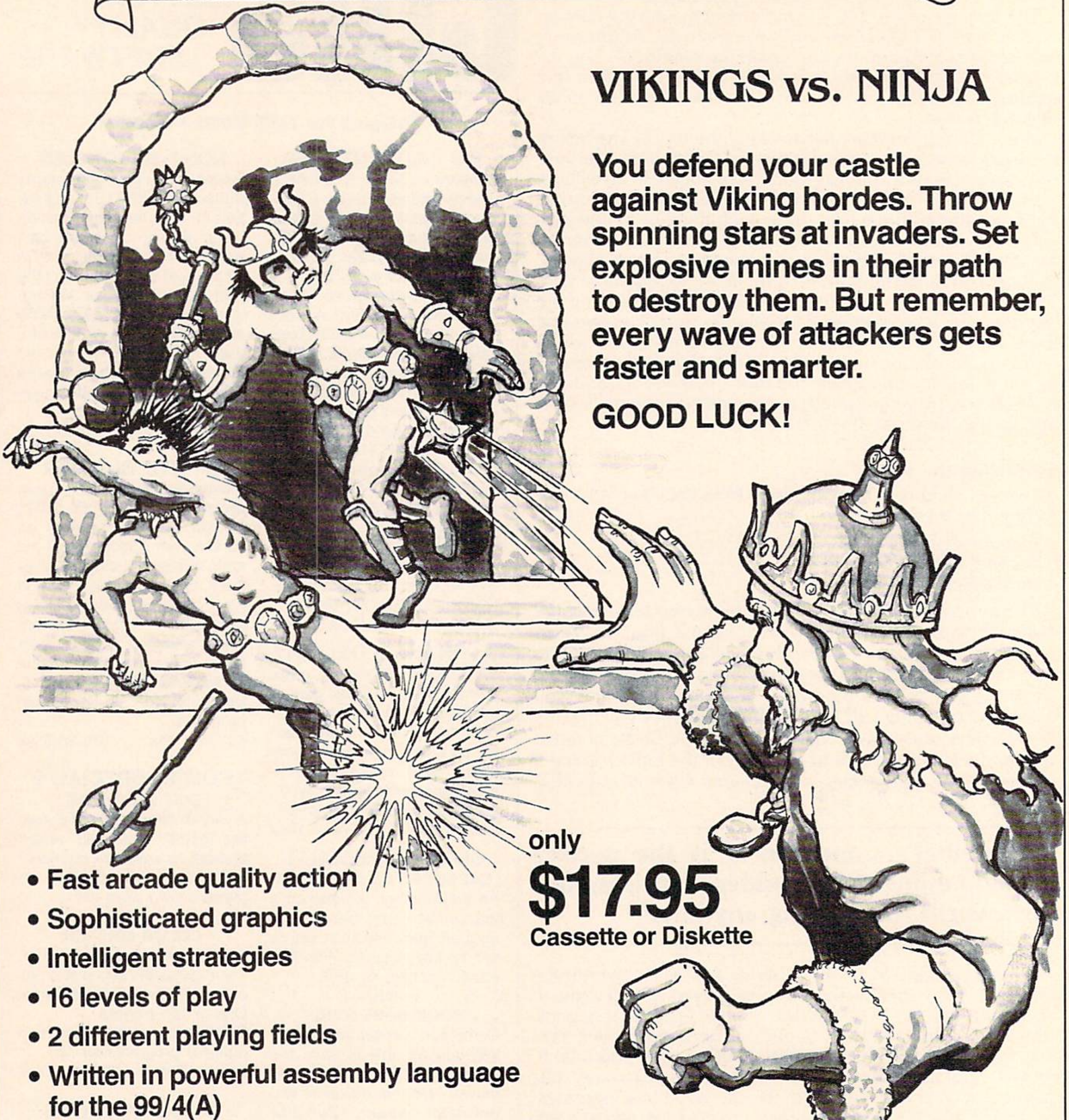
"Pachelbel's Canon in D is the perfect accompaniment for golden sunrises, but well worth the hearing anytime."

The expressions M (N/4), M (N/2) and M(N) determine which note from the vector F is to be played during a given invocation of CALL SOUND. We already know that N goes from zero up to 31 in steps of one and then starts over. We also know that the function M( ) returns the modulo 8 equivalent of its argument. In the case of the first voice, this means that as N goes from 0 to 31, the value N/4 increases by 1/4's: 0, 1/4, 1/2, 3/4, 1, 1¼, 1½ . . . up to 7¾, and M(N/4) takes the values 0, 0, 0, 0, 1, 1, 1, 1, 2, 2, etc., since M( ) rounds down to whole numbers modulo 8. In the case of the second voice, we have M(N/2) running 0, 0, 1, 1, . . . up to 7, 7 and then starting over at 0 again as N passes its halfway point. The third voice goes 0, 1, 2, . . . up to 7 four times as N goes from 0 to 31.

Now that you know where the theme is hidden, how the relative pitches of the three voices are set, and how the relative speeds of the three voices are determined, you should be able to write 8-note canons of your own. Themes of different length will require a bit more modification (to the modulo function, for example), but not much more. Now go ahead and blast your canon!

[Editor's note: Send in your best canons and fugues and we will publish as many as space in our letters column permits. Would anyone like to write an article and program to play Ragtime?] HCM

KING of the CASTLE



VIKINGS vs. NINJA

You defend your castle against Viking hordes. Throw spinning stars at invaders. Set explosive mines in their path to destroy them. But remember, every wave of attackers gets faster and smarter.

GOOD LUCK!

only
$17.95
Cassette or Diskette

- Fast arcade quality action
- Sophisticated graphics
- Intelligent strategies
- 16 levels of play
- 2 different playing fields
- Written in powerful assembly language for the 99/4(A)

MasterCard VISA **Order toll free:** 1-800-538-9787 except California. In California call direct 408/446-2800 and deduct $2.00 from the purchase price. Operators standing by 24 HRS.

KING of the CASTLE $17.95

Specify: ☐ Cassette ☐ Diskette
☐ Check or M.O. enclosed
Charge to VISA ☐ MASTERCARD ☐

____
Expiration date Account Number

Signature ______________________
Name ______________________
Address ______________________
City ____________ State ________ Zip ____________

CYDEX SOFTWARE
P.O. Box 14471
Portland, Oregon 97214

NO RISK GUARANTEE

If you are not totally satisfied with the excellent quality of this software, return it within 15 days for a full refund.

Letters

to the Editor

Slick Solutions

In your September, 1983 issue, Donald Beck presented an interesting problem. He challenged readers to write a BASIC program which found all the four-digit numbers that equal their original value after you divide them by two, add together the halves, then square the sum . His answer is inefficient, however. Here's a BASIC program that does the job over a hundred times faster:

```
100 CALL CLEAR
110 XHI=INT(SQR(9999))
120 XLO=INT(SQR(1000))+1
130 FOR X=XLO TO XHI
140 Y=X*X
150 Z=INT(Y/100)
160 IF (Y-Z*99)<>X THEN 180
170 PRINT Y
180 NEXT X
190 END
```

Woodrow Wilson
Olivenhain, CA

That's a very slick and fast solution to Mr. Beck's problem. Soon we'll have more problems for readers to work on, as you may have seen from the note in the November issue.

Memory Not Jarred

I wish to comment on two items mentioned in your September issue.

The first is a comment on the Foundation memory card. I got the 128K version and upon installation found that the card is about a sixteenth of an inch higher than the TI cards. This pressed it very firmly into the top pads of the PES box and precluded any movement, even when some varlet dropped the box on its end. This card not only survived this incident, but it has survived some very rugged trips to my North Carolina summer home in either the back of a Toyota Land Cruiser or in a little baggage trailer that I use. I do recommend that it be installed between the PES card and the RS232 card. My PES box was in a special carrier that purposely has little padding but is constructed of thick plywood. I am testing one system for possible use in Central America for portability under rural transportation conditions.

The second has to do with the letter by Clifford Parms on page 49 of the September issue concerning two drives in one. I ordered a pair in March from Western Micro Systems and received them in late June. What they sent me were two TEAC disk drives in the exact configuration that I ordered. I have not had any problems with the power supply in the PES but do recommend the added power supply that they sell as an extra. They, too, survived the summer trips.

This is my experience with these two items for whatever it may be worth.

I do like the magazine very much.

Jasper Pierce
Sumter, SC

It sounds like your equipment has taken as rough a beating as any is likely to get, and it's encouraging to hear that everything held up.

Generation Gap

I have been using the IBM PC for quite a while at the office and have amassed a great deal of software for it. I'm currently interested in purchasing the IBM PCjr for my home and am curious as to whether or not my PC software will work on the PCjr. Or will I have to buy all new software for my PCjr at home? I understand that the languages the two systems use are different as well.

Arlene Delmonico
Chicago, IL

Many of the programs written for the IBM PC will also run on the PCjr, because IBM has made an attempt at keeping the two systems compatible. A problem you may run into will be with software which requires more than the maximum 128K of memory in the PCjr including the screen memory—or uses special machine language routines that are not part of the regular BIOS calls. [See related article in this issue]. Also, because the PCjr has only one disk drive, software which requires two drives will not work unless it can be adapted to a single drive system. To get a comprehensive list of programs which are compatible with both systems, have your local IBM dealer show you the PCjr software compatibility chart.

As far as languages go, the PC's Advanced BASIC (BASICA) will not run on the IBM PCjr because it occupies the same area of memory used by Cartridge BASIC. Cartridge BASIC is located in ROM (Read Only Memory) at the cartridge port. But don't despair. Any program written in BASICA will run under Cartridge BASIC because Cartridge BASIC is a superset of BASICA. Programs written in Cartridge BASIC on the PCjr may not run on the PC under BASICA, however, because they may have Cartridge BASIC commands that aren't in BASICA.

A Difference in Assembly

I would like to commend you on a publication that I find useful and far more informative than any other in the small computer field. Your subject matter is well treated and the range of coverage that you provide on the Texas Instruments brand of computers is unmatched by any other computer magazine that I know of today. However, I do have a bone to pick with you.

Most Assembly Language programs that you publish are written for the Mini Memory Module. Many of us would like to convert these to run with the Editor/Assembler package but are unsure of just exactly how to modify them. As they now stand, they definitely do not run with the Editor/Assembler and 32K card; I know, I've tried.

David L. Ramsey
Woodbridge, VA

There are three major differences between programs for the Mini Memory and the Editor-Assembler cartridges:

Location of utility routines. They reside at different locations in the two systems. Wherever a Mini Memory program refers to a utility routine

HCM Review Criteria

Each month, *HCM* reviews software packages for the IBM PC and PCjr, Apple II, II+ and IIe,TI- 99/4A, and Commodore 64 and VIC-20 computers.These reviews take a detailed look at the quality of commercially available third-party software for these home computers.

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

- **Performance**—how well the activity responds to the player's commands; how well the sound effects, music, or speech are integrated with the software.
- **Documentation**—the quality of the printed matter that comes with the software: whether the instructions are clear and comprehensive; whether the machine configuration requirements are spelled out. Information such as how to load the program, use the keyboard, and restart the activity contributes to the documentation rating, as do tips on performance peculiarities.
- **Engrossment**—whether the game or activity has that intangible quality that holds the player on the edge of his seat while the hours tick by unnoticed.
- **Ease of Use**—the degree to which a user can interact with the software 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.

Education-Specific Criteria

Educational software may also be evaluated in the following areas:

- **Concept Presentation**—whether the concepts are presented clearly, in logical order, and in enough depth for the learner to be able to apply the learning to other situations.
- **Rewards**—whether the audio-visual rewards are motivating and whether they are appropriate to the activity.
- **Graphics**—rates the quality of the graphics and whether they enhance or detract from the educational purposes of the activity.

Letters to the Editor

(by address unless you EQUate that address to a label), you should substitute an Editor/Assembler mnemonic for that routine.

Entries in the REF/DEF table. In Mini Memory, you must insert the information necessary—program title and entry point—into the proper memory locations, and then change the starting address of the table. The Editor/Assembler does this automatically with the REF and DEF assembler directives at the start of the program.

Memory use. The space occupied by the Command Cartridge is not available for use with the Editor/Assembler. Thus any references to addresses between >7000 and >7FFF must be changed. The best way is to change all those address references to label references, which can be relocated.

As far as editorial balance is concerned, we want to cover as many aspects of the world of the Home Computer as we can, while still providing material that the greatest number of our readers can use. Those two requirements sometimes conflict; we try to strike a happy medium.

C-64 Keyboard Buffer

I am having a problem with the keyboard buffer on my Commodore 64. I have to be very careful that I don't accidentally hit any keys while running my program, because if I do, these keys get entered into the keyboard buffer, and when the program starts executing code that accesses the buffer, it reads these garbage inputs as input data. What can I do about this?

Jeff Strong
Canton, OH

There are a couple of ways to correct this problem. One method would be to clear the buffer just before accessing it. The code would look like this:

```
100 GET A$:IF A$<>"" THEN 100
```

The next line of code would then read:

```
110 GET A$:IF A$="" THEN 110
```

You can correct this another way by not accessing the keyboard buffer at all. There is a byte, maintained by the keyboard scan routine, that contains the unique code of the key currently being pressed. This byte can be read by PEEKing address 197. The code that will be returned for each key pressed is not part of any standard coding, e.g., ASCII, EBCDIC, etc. If you want to develop a table that matches codes to key presses, try inputting the following BASIC line in the command mode:

```
FOR X=1 TO 3:Z=PEEK(197):PRINT Z:X=1:NEXT
```

To find the code for each key, simply press the desired key and watch the code scroll up the screen. You can use this method in a program by inserting the following BASIC statement:

```
100 Z=PEEK(197):IF Z=64 THEN 100
```

64 is the code that will be placed in address 197 when no key is being pressed. Good luck, Jeff.

Memory Pages

I have been trying to learn more about the inner workings of my Commodore, because I want to start some machine language programming. But explanations of machine language programming are hard for a beginner to understand, partly because everyone uses terms without defining them. One term that everyone seems to use without explanation is "page." What is a "page" of memory?

Henry J. Feingold
Roanoke, VA

If you were to write a story, you would think of the work as a whole, but if the story were printed, the physical structure of the book would break the story up into pages. Similarly, we would like to deal with our computer's memory as a whole, but the hardware breaks the memory up into pages. In BASIC programming, for instance, Commodore handles virtually all the difficulties of going from page to page for you.

When you write assembly language programs, the situation is a bit different, and in that case, the zero page *will be important. A page is 256 memory locations, each one byte long, and each of those 256 bytes has an associated memory address. These memory addresses run from 0 to 65535. Most of these addresses (addresses,* not *memory locations) are two bytes long. The addresses on the zero page (the first 256 bytes of memory) are only one byte long. The Commodore operating system uses the zero page for many operations where speed is important because it evaluates these one-byte addresses more quickly than the addresses on other pages.*

Apple Text Files

When I write programs that create text files on my Apple IIe, I sometimes have problems with them. If I have a program that opens and writes to the same file with the same filename each time it runs, sometimes I get garbage at the end of that file, and I can't figure out why. Sometimes it looks like text from previous runs of the program. I hope you can help me find a solution to this problem.

Horst Wiener
Pompano Beach, FL

This is a common problem that all Apple programmers have to cope with. If a new text file is shorter than an old one with the same name, it will have part of the old file on the end. (See page 66 of Apple's DOS Programmer's Manual.*) To avoid this problem, always use the following statements when a program routinely creates and reuses a text file:*

```
200 PRINT CHR$(4);"OPEN TEXTFILE"
210 PRINT CHR$(4);"DELETE TEXTFILE"
220 PRINT CHR$(4);"OPEN TEXTFILE"
```

This first opens the old file, deletes it, and then opens it again. You can't just delete it because this will return an error—and stop your program—if you're using a diskette on which that file hasn't already been created.

This is a bit cumbersome, and we've heard that Apple's new operating system (ProDOS, to be released later this year) will address this problem, as well as some of the other Apple idiosyncracies

Consistency Problems

After a lot of soul searching, I finally managed to justify the purchase of an APPLE IIe. Since this purchase, I've spent many enjoyable hours gaming and hacking. Since the editing facilities on the APPLE IIe don't have total screen control, one of my first purchases was an editing utility—the *Global Program Line Editor*. While using this utility, I've noticed that my programs don't produce consistent results. Is there something I don't know?

Annette Weidler
Greenville, OH

The APPLE IIe is a very flexible computer. To really understand this flexibility, you must know how your software is affecting various memory locations. Some utilities, such as the Global Program Line Editor, *modify locations in the hardware I/O address area. For instance, if your program reads an input from the keyboard, APPLESOFT will add 128 to the ASCII code before placing this value in location −16384. However, certain utilities (such as the editor you use) interfere with this process. The result is that memory location −16384 contains the ASCII code for the keyboard input without the addition of 128. Try running your program immediately after booting APPLE DOS. If the problem was caused by an editing program, your program will operate as it should.*

Temporary Amnesia

One of your readers in Texas notified me of a problem with my program, *Cash Flow* (August, 1983), of which we were both unaware: The program will not run without the memory expansion card. Unless the card is inserted, an error message (MEMORY FULL IN 150) is generated.

The program was composed on my machine (which has a memory card) and as it was not a long program, the issue never arose. I suppose that the machine on which it was edited was also so equipped.

I have attempted getting around the problem through conserving memory (CALL FILES(1); omitting REMs; shortening variable names; allowing only 10 entries instead of 16) all to no avail.

I think, therefore, that you should mention that the article requires memory expansion.

Happily, I have heard this comment from only one reader (the rest called or wrote to express their delight). Hopefully the universe of those with Extended BASIC and disk drives but without memory expansion is small.

Thanks for an excellent job of editing and a typo-less printing job on the program. Should I find the time to do a third article, I will be in touch.

Joel S. Moskowitz
Rancho Cordova, CA

Thanks for your letter, Joel. You're right—we inadvertently edited your program on a machine with a memory expansion card, and the fact that memory expansion was required escaped us. We hope this didn't inconvenience our readers unduly. At the moment, however, we are investigating the possibility of program compaction as well, so that perhaps more TI users can take advantage of your excellent program. HCM

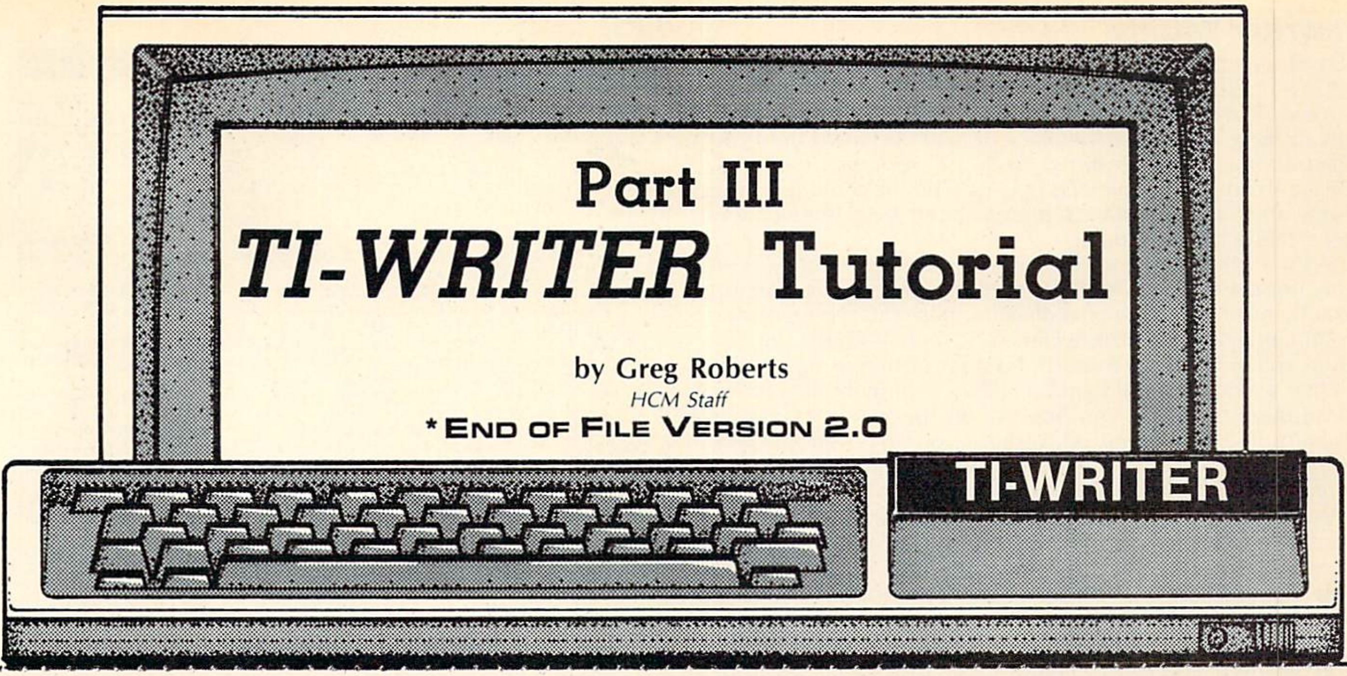
NOW YOUR COMPUTER FITS THE ORIGINAL ARCADE HITS.
COMMODORE 64
COMMODORE VIC 20
IBM PC
DIG DUG
FROM ATARISOFT
STARGATE
FROM ATARISOFT
ROBOTRON: 2084
FROM ATARISOFT
CENTIPEDE
FROM ATARISOFT
DONKEY KONG
BY NINTENDO®


TI-99/4A
THE EVERYTHING BOOK
For The TEXAS INSTRUMENTS 99/4A HOME COMPUTER
• HOW-TO Information on expanding your system
• GAMES - The newest, the best, the most
• GIANT Software Directory over 500 descriptions
• Peripherals and Accessories Hundreds of items: printers, joysticks, disk drives, plotters, modems, monitors & much more!
• DISCOUNTS! And Easy Order Info
GET IT FREE!
Just Call Us Toll-Free
1-800-348-2778
Or send your name and address to:
TENEX, Box 6578, South Bend, IN 46660
(219) 277-7726
© TENEX Computer Marketing Systems, Box 6578, South Bend, Indiana 46660

Part III
TI-WRITER Tutorial

by Greg Roberts
HCM Staff

***END OF FILE VERSION 2.0**



This month we cover the remaining "frills" in the *TI-WRITER* word processing program. These features may not be essential to your writing, but they certainly can speed it along.

You may already have some of your writings on diskette. So let's start out by loading a file. Press [FCTN] [9], type LF and [ENTER]. Note that you do not have to select 'Files' from the command line. Just go directly to LF and enter DSK1. plus the name of the file. You don't have an old file to play with? Then just put a few lines of gibberish on the screen so that we can get started.

Home Cursor [CTRL] [L] puts the cursor in the upper left-hand corner of the screen, saving you the trouble of moving it with the arrow keys. An even more specific operation, [CTRL] [V], moves the cursor to the beginning of the line you are working on. A similar time-saver is Last Paragraph or [CTRL] [H]. It shoots the cursor back to the first letter of the previous paragraph.

More minor tactics: Duplicate Line lets you reproduce the line appearing above your cursor. Press [CTRL] [5]. Delete Line, [FCTN] [3] or [CTRL] [N] , takes out an entire line and closes up the text. You can even Delete End of Line by using [CTRL][K] to erase the character under your cursor, and everything to the right of it.

By this time you may be questioning the value of learning such restricted plays. After all, when would you ever have to copy the exact line you've just written? It is, admittedly, an obscure device belonging on the same shelf with electric carving knives and patio bug zappers. The only place I can think of using Duplicate Line would be in song writing. I suppose the Beatles might have found a use for it in setting down that immortal line

"I get by with a little help from my friends."

That is, they repeat the line with just a minor variation. Duplicate Line would have saved them the trouble of typing it twice. Some of the other operations will, I hope, more readily show their value. At first some of these may not seem as convenient as using the arrow keys, but eventually they will prove their worth. If you practice the commands, you will soon be using all of them without having to refer to the strip at the top of the keyboard.

Crazyquilt Commands

Some of the difficulty in learning these operations comes from their haphazard arrangement on the keyboard. They may be triggered by the Control key, the Function key, with letters or numbers. Many operations can be activated by two combinations of keys, adding to the confusion. It is unfortunate that some of the operations aren't tied to keys that would help in memorizing them—for example, [CTRL] [H] for Home Cursor, or [CTRL] [O] for Oops! But such is not the case with the present version of *TI-WRITER*. Nevertheless, nearly all the operations described in the manual can be time-savers, once you get them under your thumb.

Several of these operations are so close to the Delete Line feature ([FCTN][3]) that they can easily cause you to blank out a line of text. But you don't necessarily have to retype the lost line. If you press [CTRL] [1] or [CTRL] [Z], otherwise known as Oops!, you regain the line. You must, however, press Oops! immediately after the mistake occurs. If you have already started another line, the program cannot recover the lost line.

> **"You are forced to retype DSK1. and the name for a deletion, to give you time to think before you plunge that masterwork into oblivion."**

Now on to large-scale operations. DeleteFile certainly is not in the frill category. This command lets you clear your diskette of unwanted programs. When you press [FCTN] [9], then DF, note that the program does not automatically display the filename. You are forced to retype DSK1. and the name for a deletion, to give you time to think before you plunge that masterwork into oblivion.

The Purge command is like the Delete command, but without its finality. Use [FCTN] [9], then P, to clear the contents of the screen after you have saved the material on diskette. You can then conveniently start your next file. This one also carries a safeguard: The program asks PURGE FILE ARE YOU SURE (YES OR NO)? It gives you time to make sure you have entered SaveFile before you clear your work from the screen. Beyond this safeguard, you get yet another chance to recoup from having accidentally purged a file. RecoverEdit is a command that comes from pressing [FCTN] [9] followed by RE. You must access this command as soon as possible after accidentally purging your file, or the program will not be able to recover it.

But even with these safety features, it is easy to destroy your work and not realize it until much later. For example, you purge a file, start a new one, then go to save the second file with the SF command. Here it is very easy to unwittingly hit

Continued on p. 30

TI-Writer Tutorial

SF without changing the file name that automatically appears on the screen—the file name of the previous work already purged. The disk drive spins, and it's *adios Carlos*, as a two-line mailing label takes the place of a 20,000 word thesis on *Frustration-Induced Trauma Among Computer Operators*. This problem can be avoided only by working cautiously with Purge and Save commands, and probably by suffering a few devastating losses of data.

With all this Deleting, Purging, and RecoverEditing, you may not always be sure of the contents of a file. To find out exactly what you have on a diskette, push [FCTN] [9] for the command line, SD for ShowDirectory, then [ENTER]. The program will ask you for the disk number (a simple numeral is all that's required) and then list all files in alphabetical order, in addition to telling you how much space is left on your diskette (0-358 sectors). Make sure your *TI-WRITER* Command Cartridge is in the cartridge port when you ShowDirectory, or your program will crash.

Wait, you say. What an absurdly obvious remark. Of course you have to keep your Command Cartridge in the machine in order to use the program. Surprise! Such is not the case. Once you set up your word processing program to the point of entering TEXT EDITOR from the main menu, you can pull out the cartridge, and *TI-WRITER* just keeps on breathing. You can then lend it to someone across the office, classroom or street. With just your user diskette you can load, save, and print files. Do not, however, try to ShowDirectory or use RecoverEdit without the cartridge.

Once you learn these refinements, you will be in control of all the editing power of the *TI-WRITER* word processor. But that is just the beginning. Next time we will explore some of the printing and formating options available in this program.

HCM

Uncle Larry . . . from p. 17

IBM PCjr

```
 980 DATA "O3 T120 AF#DDEF# T60 G T120 E
     AF#DDEF#GF#EAF#DDEF# T60 G T120 EF#
     ED T60 E T120"
 990 DATA "T60 O2 F#>D T120 F#DEF#GE T60
      D T120 F#DEF#GEDF#A>DC#<BAGF#GABAG
     F#E T60 D T120 F#DEF#GE T60 D T120
     F#DEF#GEDF#A>DC#<BAGF#GEF# T60 D"
1000 DATA "T120 O2 AGF#A>D<AF#A>D<AGB>D<
     BGB>D<BAB>D<BAB>C#DEF#GEF#GEF#DC#<B
     AF#A>D<AF#A>D<AGB>D<BGB>D<BAB>C#DEF
     #GEF#GEF# T40 D"
1010 DATA "T240 O3 DC T120<BGGDGGBGB>DC<
     B>C<AADAA>C<A>CEDC<BGGDGGBGB>DC<B>C
     <B>C<A>DC<BGG T60 G"
1020 DATA "T240 O3 GA T120 BGGDGGBGBBAGA
     F#F#DF#F#F#DF#AGF#C#GGDGGC<BA>DC<BG
     G T60 G"
1030 DATA "T120 O2 B T40 >D T120 ED <B>G
     <B>C<B>G<B>A<AAB T40 >D T120 ED<B>G
     <BA>C<BABGGBB>DDED<B>G<BA>C<BABGG"
1040 DATA "O2 B>DGBGAGBGDGBGAGEGDGBGAGBG
     DEDC<BGGEDGBGAGBGDGBGAGEGBAGF#GF#ED
     EABGG"
1050 DATA "T171 O2 DEF# T60 G T120 BG>C<
     GBG>C<GBGADEF# T60 G T120 BF>C<GBGA
     >DC<AG"
1060 DATA "O3 T171 DEF# T60 G T171 AGF#G
      T120 D<B>D T60 G T171 AGF# T120 GA
     BA T60 G T171 AGF# T120 GD<B>DEGF#A
      T60 G"
1070 END
```

HCM

Uncle Larry . . . from p. 15

TI-99/4A

```
280 NEXT X
290 REM   PRINT MENU
300 FOR X=1 TO 17 STEP 8
310 CALL HCHAR(X,1,95,256)
320 NEXT X
330 RESTORE
340 CALL HCHAR(24,1,32,32)
350 FOR X=1 TO 3
360 IF X<>2 THEN 390
370 PRINT TAB(5);"LARRY'S FIDDLE TUNES
    "
380 GOTO 400
390 PRINT
400 CALL HCHAR(23,1,95,4)
410 CALL HCHAR(23,29,95,4)
420 NEXT X
430 FOR X=1 TO 14
440 READ T$,M$
450 A$(X)=M$
460 PRINT TAB(3);T$
470 CALL HCHAR(23,1,95,4)
480 CALL HCHAR(23,29,95,4)
490 NEXT X
500 PRINT ::
510 REM  SELECT FROM MENU
520 M$="SELECTION:    KEY:   1ST NOTE:"
530 C=2
540 GOSUB 2150
550 INPUT "    ":S
560 IF (S<1)+(S>13)THEN 550
570 M$=A$(S)
580 C=16
590 GOSUB 2150
600 X=0
610 ON S GOSUB 640,720,800,880,960,1040
    ,1120,1200,1280,1360,1460,1970,2050
620 GOTO 290
630 REM  SUBROUTINES FOR THE 10 TUNES
640 DATA "  (1) SOLDIER'S JOY",D      F#
    ON 3
650 RESTORE 2340
660 GOSUB 2210
670 IF X<2 THEN 650
680 RESTORE 2410
690 GOSUB 2210
700 IF X<4 THEN 680
710 RETURN
720 DATA "  (2) ARKANSAS TRAVELLER",D
        A ON 4
730 RESTORE 2470
740 GOSUB 2210
750 IF X<2 THEN 730
760 RESTORE 2520
770 GOSUB 2210
780 IF X<4 THEN 760
790 RETURN
800 DATA "  (3) CINCINNATI HORNPIPE",D
      D/OPEN 3
810 RESTORE 2570
820 GOSUB 2210
830 IF X<2 THEN 810
840 RESTORE 2630
850 GOSUB 2210
860 IF X<4 THEN 840
870 RETURN
880 DATA "  (4) POP GOES THE WEASEL",G
          G ON 3
890 RESTORE 2680
900 GOSUB 2210
```

```
910 IF X<2 THEN 890
920 RESTORE 2720
930 GOSUB 2210
940 IF X<4 THEN 920
950 RETURN
960 DATA " (5) GARRY OWEN",G     G ON
    1
970 RESTORE 2760
980 GOSUB 2210
990 IF X<2 THEN 970
1000 RESTORE 2810
1010 GOSUB 2210
1020 IF X<4 THEN 1000
1030 RETURN
1040 DATA " (6) COCK AND HEN",G      B O
    N 2
1050 RESTORE 2860
1060 GOSUB 2210
1070 IF X<2 THEN 1050
1080 RESTORE 2900
1090 GOSUB 2210
1100 IF X<4 THEN 1080
1110 RETURN
1120 DATA " (7) TOM & JERRY REEL",D     A
    /OPEN 2
1130 RESTORE 2940
1140 GOSUB 2210
1150 IF X<2 THEN 1130
1160 RESTORE 3000
1170 GOSUB 2210
1180 IF X<4 THEN 1160
1190 RETURN
1200 DATA " (8) IRISH WASHERWOMAN",G
      D ON 2
1210 RESTORE 3060
1220 GOSUB 2210
1230 IF X<2 THEN 1210
1240 RESTORE 3110
1250 GOSUB 2210
1260 IF X<4 THEN 1240
1270 RETURN
1280 DATA " (9) MC DONALD'S",G      B ON
     2
1290 RESTORE 3160
1300 GOSUB 2210
1310 IF X<2 THEN 1290
1320 RESTORE 3210
1330 GOSUB 2210
1340 IF X<4 THEN 1320
1350 RETURN
1360 DATA "(10) TWO FORTY REEL",G     D/O
    PEN 3
1370 RESTORE 3250
1380 GOSUB 2210
1390 IF X<2 THEN 1370
1400 RESTORE 3290
1410 GOSUB 2210
1420 IF X<4 THEN 1400
1430 RETURN
1440 DATA "",""
1450 REM
1460 DATA (11) tune the fiddle,""
1470 REM
1480 CALL CLEAR
1490 PRINT " PRESS NUMBER":" OF STRING
    TO":" BE TUNED.":::
1500 PRINT " PRESS (0)":" FOR MENU.":::
    :::
1510 CALL VCHAR(1,22,33,96)
1520 CALL VCHAR(1,22,44,10)
1530 CALL VCHAR(1,23,44,6)
1540 CALL VCHAR(1,24,44,4)
1550 CALL VCHAR(1,25,44,8)
1560 CALL HCHAR(14,22,44,4)
1570 PRINT TAB(20);"4321"
1580 CALL KEY(0,K,S)
1590 IF (S=0)+(K<48)+(K>52)THEN 1580
1600 ON (K-47)GOSUB 1620,1630,1670,1710,
    1750
1610 GOTO 1580
1620 GOTO 300
1630 N=69
1640 F=659
1650 R=7
1660 GOTO 1780
1670 N=65
1680 F=440
1690 R=3
1700 GOTO 1780
1710 N=68
1720 F=294
1730 R=5
1740 GOTO 1780
1750 N=71
1760 F=196
1770 R=9
1780 CALL SOUND(-SP*20,F,1)
1790 IF K>50 THEN 1840
1800 C=27
1810 GOSUB 1880
1820 CALL HCHAR(R-2,C-1,44)
1830 GOTO 1940
1840 C=18
1850 GOSUB 1880
1860 CALL HCHAR(R-2,C+3,44)
1870 GOTO 1940
1880 REM DISPLAY PEG
1890 FOR X=1 TO 3
```

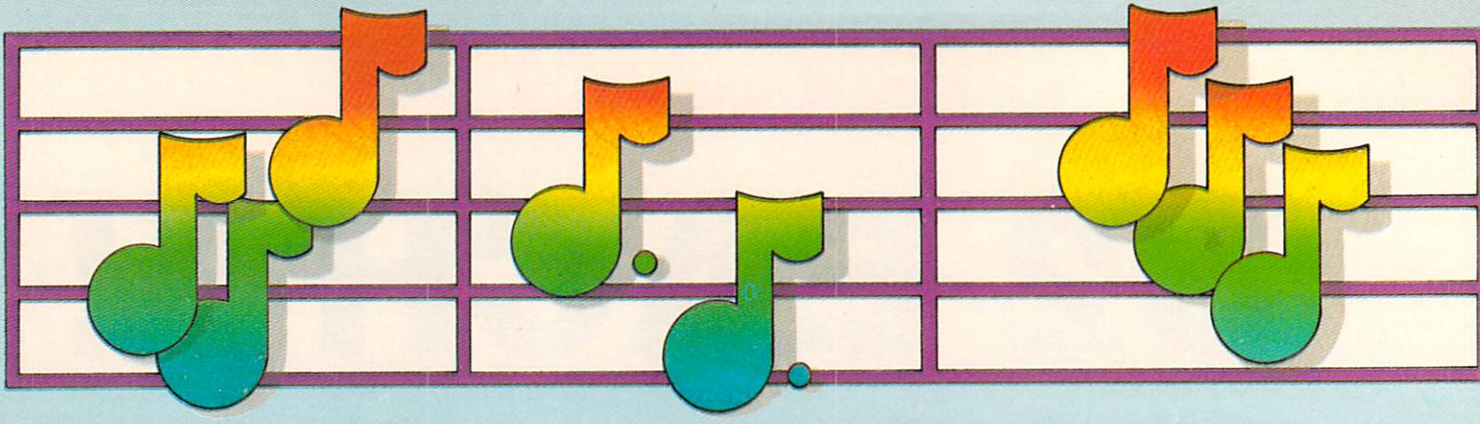
Continued on p. 32

TI-99/4A

```
1900 CALL HCHAR(R,C,44,3)
1910 R=R+1
1920 NEXT X
1930 RETURN
1940 CALL HCHAR(R-2,C+1,N)
1950 GOTO 1580
1960 REM
1970 DATA (12) change speed,""
1980 REM
1990 M$="SPEED 50-500 (HIGHER=SLOWER)"
2000 C=2
2010 GOSUB 2160
2020 INPUT "    ":SP
2030 RETURN
2040 REM
2050 DATA (13) end,""
2060 REM
2070 CALL CLEAR
2080 PRINT "_ NEXT TIME - DOUBLE NOTES _
":::::::::
2090 RESTORE 3340
2100 GOSUB 2210
2110 CALL SOUND(INT(SP*2),440,2,554,0)
2120 CALL SOUND(INT(SP*8),294,8,587,0)
2130 CALL CLEAR
2140 STOP
2150 REM PRINT M$ ROUTINE
2160 FOR X=1 TO LEN(M$)
2170 CALL HCHAR(23,C+X,ASC(SEG$(M$,X,1)
)
2180 NEXT X
2190 RETURN
2200 REM PLAY ROUTINE
2210 READ D
2220 IF D=99 THEN 2300
2230 IF D<110 THEN 2240 ELSE 2260
2240 READ F
2250 GOTO 2280
2260 F=D
2270 D=1
2280 CALL SOUND(INT(SP*D),F,1)
2290 GOTO 2210
2300 X=X+1
2310 RETURN
2320 REM THERE ARE 16 NUMBERS ON EACH D
ATA LINE
2330 REM EXCEPT FOR REMAINDERS AT THE E
ND OF A MUSICAL PHRASE
2340 REM SJ
2350 DATA 370,392,440,370,294,370,440,37
0,294,370,2,440,2,587,2,587
2360 DATA 554,494,440,370,294,370,440,37
0,294,370,2,392,2,330,2,330
2370 DATA 370,392,440,370,294,370,440,37
0,294,370,2,440,2,587,2,587
2380 DATA 659,784,740,880,740,587,659,78
4,659,554,2,587,2,587,2,587
2390 DATA 99
2400 REM SJ2
2410 DATA 587,659,740,659,587,659,740,88
0,784,740,659,587,554,587,659,740
2420 DATA 784,659,740,659,587,659,740,78
4,880,740,659,587,554,494,2,440
2430 DATA 2,784,740,659,587,659,740,784,
880,740,659,587,554,587,659,740
2440 DATA 784,659,740,880,740,587,659,78
4,659,554,2,587,2,587,2,587
2450 DATA 99
2460 REM AT
2470 DATA 220,247,277,294,370,330,294,2,
247,2,247,2,220,2,220,4
2480 DATA 294,330,330,2,330,370,370,2,37
0,294,370,330,294,2,247,2
2490 DATA 220,294,370,330,294,2,247,2,24
7,2,220,2,220,4,294,587
2500 DATA 554,587,440,494,587,440,392,37
0,330,294,277,2,294,99
2510 REM AT2
2520 DATA 880,784,740,880,784,740,659,78
4,740,659,587,740,659,587,554,440
2530 DATA 587,554,587,740,659,587,659,78
4,740,659,587,740,2,659,740,784
2540 DATA 880,784,740,880,784,740,659,78
4,740,659,587,740,659,587,554,440
2550 DATA 587,554,587,440,494,587,440,39
2,370,330,294,277,4,294,99
2560 REM CH
2570 DATA 294,440,370,440,294,440,370,44
0,587,440,740,440,659,440,740,440
2580 DATA 784,440,740,440,659,587,554,58
7,659,587,554,494,440,392,370,330
2590 DATA 294,440,370,440,294,440,370,44
0,587,440,740,440,659,440,740,440
2600 DATA 784,440,740,440,659,587,554,49
4,554,587,659,784,740,587,2,587
2610 DATA 99
2620 REM CH2
2630 DATA 659,440,440,440,740,440,440,44
0,784,440,440,440,740,440,440,440
2640 DATA 659,440,740,440,784,440,740,44
0,659,587,554,494,440,392,370,330
2650 DATA 294,587,587,587,554,659,659,65
9,587,740,740,740,659,784,784,784
2660 DATA 740,784,880,740,988,784,659,55
4,2,587,2,587,3,587,99
2670 REM PGTW
2680 DATA 2,392,392,2,440,440,494,587,49
4,2,392,22222,2,392,392,2
2690 DATA 440,523,3,494,2,392,22222,2,39
2,392,2,440,440,494,587,494
```

TI-99/4A

```
2700 DATA 2,392,22222,3,659,2,440,523,3,
494,2,392,99
2710 REM PGTW2
2720 DATA 3,784,2,659,784,740,880,740,3,
587,2,784,784,2,659,784
2730 DATA 3,740,2,587,494,2,523,494,2,52
3,587,2,659,740,2,784
2740 DATA 22222,659,2,22222,2,440,523,3,
494,2,392,99
2750 REM GO
2760 DATA .5,784,.5,740,659,587,523,494,
440,392,494,523,494,494,784,740
2770 DATA 659,587,523,494,440,392,440,49
4,440,440,784,740,659,587,523,494
2780 DATA 440,392,494,523,494,2,494,523,
587,659,740,784,587,494,440,494
2790 DATA 440,3,440,99
2800 REM GO2
2810 DATA .5,494,.5,523,2,587,494,2,587,
494,2,587,494,587,784,740
2820 DATA 2,659,523,2,659,523,2,659,523,
2,659,740,2,784,880,2
2830 DATA 988,880,784,740,659,587,523,49
4,587,659,740,784,587,494,440,494
2840 DATA 3,440,99
2850 REM C&H
2860 DATA 494,523,494,494,440,494,2,794,
659,494,523,494,494,440,494,587
2870 DATA 523,494,494,523,494,494,440,49
4,2,784,659,2,740,587,2,659
2880 DATA 494,587,494,2,440,99
2890 REM C&H2
2900 DATA 880,740,587,587,659,740,2,784,
659,880,740,587,587,659,740,784
2910 DATA 740,659,880,740,587,587,659,74
0,2,784,659,740,659,587,2,659
2920 DATA 494,587,494,2,440,99
2930 REM T&J
2940 DATA 2,440,2,587,740,587,659,740,78
4,659,2,587,740,587,659,740
2950 DATA 784,659,587,740,880,1175,1109,
988,880,784,740,784,880,988,880,784
2960 DATA 740,659,2,587,740,587,659,740,
784,659,2,587,740,587,659,740
2970 DATA 784,659,587,740,880,1175,1109,
988,880,784,740,784,659,740,2,587
2980 DATA 99
2990 REM T&J2
3000 DATA 440,392,370,440,587,440,370,44
0,587,440,392,494,587,494,392,494
3010 DATA 587,494,440,494,554,587,659,74
0,784,659,740,784,659,740,587,554
3020 DATA 494,440,370,440,587,440,370,44
0,587,440,392,494,587,494,392,494
3030 DATA 587,494,440,494,554,587,659,74
0,784,659,740,784,659,740,3,587
3040 DATA 99
3050 REM IW
3060 DATA .5,587,.5,523,494,392,392,294,
392,392,494,392,494,587,523,494
3070 DATA 523,440,440,294,440,440,523,44
0,523,659,587,523,494,392,392,294
3080 DATA 392,392,494,392,494,587,523,49
4,523,494,523,440,587,523,494,392
3090 DATA 392,2,392,99
3100 REM IW2
3110 DATA .5,784,.5,880,988,784,784,587,
784,784,988,784,988,988,880,784
3120 DATA 880,740,740,587,740,740,740,58
7,740,880,784,740,659,784,784,587
3130 DATA 784,784,523,784,784,494,784,78
4,587,523,494,440,587,523,494,392
3140 DATA 392,2,392,99
3150 REM MCD
3160 DATA 494,3,587,659,587,494,784,494,
587,494,784,494,880,440,440,494
3170 DATA 3,587,659,587,494,784,494,440,
523,494,440,494,392,392,494,494
3180 DATA 587,587,659,587,494,784,494,58
7,494,784,494,880,440,440,494,494
3190 DATA 587,587,659,587,494,784,494,44
0,523,494,440,494,392,392,99
3200 REM MCD2
3210 DATA 494,587,784,988,784,880,784,98
8,784,587,784,988,784,880,784,659,7
84,587,784,988,784,880,784,988,784
3220 DATA 587,659,587,523,494,392,392,33
0,294,392,494,392,440,392,494,392,2
94,392,494,392,440,392,330,392
3230 DATA 494,440,392,370,392,370,330,29
4,330,370,393,440,494,392,392,99
3240 REM TFR
3250 DATA .7,294,.7,330,.7,370,2,392,494
,392,523,392,494,392,523,392
3260 DATA 494,392,440,294,330,370,2,392,
494,392,523,392,494,392,440,587
3270 DATA 523,440,392,99
3280 REM TFR2
3290 DATA .7,587,.7,659,.7,740,2,784,.7,
880,.7,784,.7,740,784,587
3300 DATA 494,587,2,784,.7,880,.7,784,.7
,740,784,880,988,880,2,784
3310 DATA .7,880,.7,784,.7,740,784,587,4
94,587,659,784,740,880,2,784
3320 DATA 99
3330 REM END
3340 DATA 2,587,.7,440,.7,440,2,494,2,44
0,2,22222,99
```



I Write the Songs:

Electronic Sheet Music

Program by Carol Burris

and the HCM Staff

Like many Home Computer users, I was amazed to discover the 99/4A's extensive sound, color and graphics capabilities. After finding a formula to generate a musical scale in Herbert Peckham's book, *Programming BASIC*,* I decided to try some musical programming. I wanted a program to not only play music, but to display the notes on a staff as well. The *magic* in the title of my resulting program refers to the fact that in writing it, I really learned BASIC.

Music Magic is an easy-to-use Extended BASIC program which lets you play, display and save music. Songs can be composed of notes or chords within a two-octave range (A below middle C to high G# [no Ab] above high C) and can be up to 43 notes or chords in length.

Getting Started

After a short introduction, the first menu appears. You may choose either 1) Recorded Song—which will load and play a song previously saved on tape or diskette—or 2) New Song. Naturally, the first time you use *Music Magic* you will press 2. A second menu then appears which asks you to select either Single Note Entry, Two-Part Harmony, or Three-Part Harmony. After you make this choice, a third menu asks if the key you're writing in has any sharps. If you type N (No), the next menu will appear, asking if your piece has any flats. If it does, after you type Y (Yes), the next screen will ask, HOW MANY? You need to tell the computer the letter name of each flat (or sharp). If we were writing in the key of F we would tell the computer that we have one flat, B flat. In this program you will not be able to cancel the flat and play a B natural, because there is no provision for accidentals (sharps, flats, and natural signs that appear next to the note and not in the key signature).

When your choices are made, you can begin entering your song. One staff of music appears, ready to receive your notes, and the message NOTE 1 prompts your first entry. After each note is entered, it is played and displayed on the staff.

If you decide that the note (or chord) is not what you wanted, press "R", and the computer will allow you to enter it again. Throughout the program, whether you have just selected the key signature, or have selected Recorded Song instead of New Song, entering "R" will let you backtrack.

The staff displays approximately 13 notes, and then a new staff will begin.

Our Song

In order to get a feel for what this program can do for you, let's arrange the popular carol, "Joy to the World" for the 99/4A using *Music Magic* so you can pick up music writing hints along the way. We will write our version in the key of C—with no sharps and no flats.

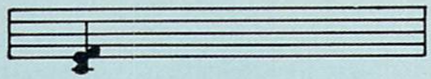
Entering single notes (without harmony) is a simple matter of typing in the letter name of the note. In the two- or three-part option, things get a little tricky. If you want a single note to appear on the staff you will need to enter the same note two or three times in succession. You must also remember to enter the lowest note of the dyad or triad first.

For example, to enter the first dyad in our song properly, you should type the E first and then the C. To indicate that this is the C above Middle C, however, you must also type [FCTN][O] for the single quote. Your second entry will be displayed as C' on the screen, and the note will be printed in the third space from the bottom.

This takes care of the first note in our song; next comes the prompt to enter the duration of the note. An eighth note (♪) is the shortest note, and the whole note (𝅝) is the longest note available. Durations are entered by typing in a number according to the note you want from the chart below.

NOTE VALUE:		TYPE IN:
Eighth	♪	1
Quarter	♩	2
Half	𝅗𝅥	4
Whole	𝅝	8

Before you put the rest of "Joy to the World" on the staff, here are some more hints about entering notes. If you put in a three-part combination like this:



it is counted as two notes, not one. The note that looks like it is on the wrong side of the stem (but isn't!) counts as one extra note space. Therefore, one less space is available for other notes in your song. In other words, instead of having room for 43 notes you will, after entering this combination as your first note, have 41 note spaces left. If you have chosen 2- or 3-part harmony and wish to play a single note, simply enter the same note for all of the note prompts for that space. To enter a 2-note combination in 3-part mode, simply type in the higher note of the two, twice. For example, E, C', C' results in E and C' being printed and played.

You can keep track of song length with the note counter located in the lower left corner of the screen. If you accidentally exceed 43 notes, the entry stops, and your song thus far is played. To end a song of less than 43 notes, enter Q after the NOTE 1 prompt.

The next menu screen will offer the self-explanatory choices of: 1) PLAY 2) RECORD 3) NEW SONG or 4) EXIT. Now, choose the New Song option and enter all the notes for the first seven measures

Continued on p. 36

*Texas Instruments and McGraw Hill, 1979.

Just Assemble Melody:

Music in Mini Memory

by Cleon Chapen

For the past year or so I've been writing Assembly Language programs for my TI, and after tackling such sticky subjects as bit-mode graphics, sprites, and data files, the time has come to face the music.

As a musician by profession and a compulsive TI devotee by inclination, one would think I could dive right in and make beautiful music. But after taking a long hard look at the Editor/Assembler manual's chapter on sound, I was, quite frankly, intimidated. The detailed bit-picking needed to format correct sound lists seemed like an awful lot of work. When you create sound data, you are writing machine language programs for the sound processor to execute. The natural thing to do, then, is to write a program that will remember all those details for you. After all, isn't that what computers are for?

This program does for the sound processor what the *Line-by-Line Assembler* does for the CPU: translates a symbolic assembly code (in this case, very much like TI BASIC CALL SOUND statements) into a machine-executable form. In addition, it will (if you choose) load and run a short machine language subprogram called PLAY which executes the sound list through TI BASIC.

If you haven't already typed the program in, I'll wait here for a few minutes while you do so . . . Be extra careful in typing the DATA statements marked MACHINE ROUTINE. A code in the *Music Assembler* checks this data, but some errors may not be detected.

Before you run the program, type CALL INIT to clear the contents of the Mini Memory so that plenty of memory is available for the sound lists. The program will begin putting in the lists at location >701C (the FFM pointer). If there is not enough room for the sound list and the PLAY subprogram to live together, the *Music Assembler* will issue a ****MEMORY FULL**** error and stop.

Type RUN, and *Music Assembler* will initialize while the title screen is displayed. An initial prompt will be displayed, telling you how to stop the assembler. Enter sound data after each "?" prompt. To assemble your first sound list, consider the following TI BASIC sound statement:

Example 1

```
CALL SOUND(1000,440,15,554,15,659,15,-5,20)
```

Notice that the code between parentheses consists of nine fields separated by commas, each of which has a certain meaning depending on its position. *Music Assembler's* version of this statement looks like this:

Example 2

```
1000.440.15 .554.15.659.15.-5.20.
```

The meaning of each number remains the same, but the statement is slightly more compact. Since TI BASIC has an aversion to the comma in INPUT statements, we use the period instead. The only other difference is that each statement must end with a period. If you forget, the program will make you do it over again. Politely.

Here is another version of the same statement which will produce the same machine code:

Sound List 1

```
(1) 1000.A1.15.C#1.15.E2.15.-5.20.
```

You may specify notes by name as well as frequency. There are several advantages to this. The most obvious is that you don't have to look up frequencies any more. In addition, the frequencies that the program uses are slightly more accurate than whole numbers, with improved intonation for the ultra-fastidious ear (like mine).

The note names that *Music Assembler* recognizes correspond to those in the Editor/Assembler manual (section 20.3), minus the generator number. They extend from A0 (110) to F6(5587.65). To simplify the search routine, only sharps are used. To specify Bb3, for instance, use A#3 instead. Of course, you may specify notes by the range of frequencies allowable in TI BASIC.

Type in Sound List 1 above, then press [ENTER]. *Music Assembler* will assemble the line, checking to be sure all the values are within the proper ranges, then load the data into memory. It will then ask for your next line with the "?" prompt. If you discover that you mistyped something (3000 instead of 1000, for example) just type REDO and press [ENTER]. The *Music Assembler* will back up one line, forget what you just typed, and ask you to re-enter the line. If you type in the wrong designation for a note—FI instead of F1, for instance—it will prompt you to reenter the *note* designation only—F1—not the whole sound list. Since your first sound list should be a little more interesting, try entering these lines:

Sound List 1 (continued)

```
(2) 1000.A1.15.D2.15.F#2.15.
(3) 1000.A1.15.D2.15.G2.15.
(4) 1000.D2.15.F#2.15.A2.15.
```

When the last line is finished assembling, stop the process by pressing [ENTER]. The *Music Assembler* will add some code to the end of your list to make the sound processor shut off when it finishes the list, and reset the First Free Memory pointer to the next word boundary (even-numbered address) after the sound list. This is where the next list will be loaded (assuming you do not issue a CALL INIT command) the next time you run the program. This stacking of lists can be very useful if you are assembling a number of them. You can locate them anywhere you want by presetting >701C via the EASY BUG program.

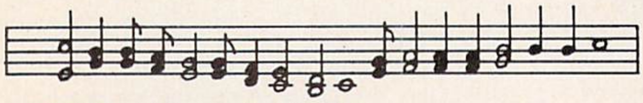
If you want to hear it played (an understandable desire), type Y in answer to the question PLAY LIST? (Y/N). The *Music Assembler* will then load the PLAY subprogram and execute it.

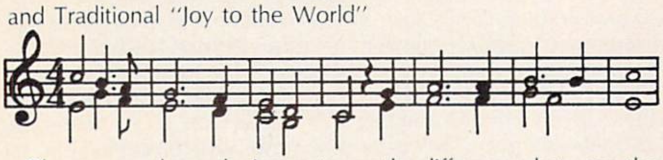
After the music is over, press [ENTER], and *Music Assembler* will tell you (in decimal and hexadecimal notation) where the sound

Continued on p. 37

I Write the Songs . . . from p. 34

of "Joy to the World". We can compare the electronic sheet music version with the way it looks on traditional sheet music.

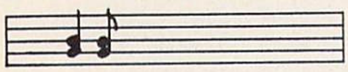
Music Magic "Joy to the World"

and Traditional "Joy to the World"



These examples make it easy to see the differences between the two notations. In the *Music Magic* version:

1. There are no measure bars.
2. The notes are not spaced on the staff according to their durations.
3. All of the stems go up and are on the right-hand side of the notes.
4. There are no dotted notes or rests.

The first three points are merely cosmetic differences, but not having dotted notes or rests can be a serious musical limitation. Actually, you can compensate for their absence by using some simple values in music. All of the notes have a relative duration. That is, an eighth note always lasts half as long as a quarter note, which lasts half as long as a half note, and so on. With 4/4 time, in which there are four beats to a measure, a whole note (𝅝) is 4 counts, a half note (𝅗𝅥) is 2 counts, a quarter note (♩) is 1 count, and an eighth note (♪) is a half count. When a dot appears next to a note, it increases that note's duration by one half. A dotted quarter note, then, would last for 1½ counts (in 4/4 time), or as long as a quarter note and an eighth note combined. When you see a dotted note, such as the second one in "Joy to the World" (traditional version), it would "translate" into this:



Rests are handled in a similar way. Each rest has a value that corresponds to a note of equal duration.

An Eighth rest (𝄾) = ♪
A Quarter rest (𝄽) = ♩
A Half rest (𝄼) = 𝅗𝅥
A Whole rest (𝄻) = 𝅝

Our solution to the rest problem was to make the note value before the rest *longer* and the note following it *shorter* than they would be in the traditional version.

So,  becomes 

You will, no doubt, make some interesting innovations of your own as you finish this song and create new ones.

The Record option allows you to put a melody you've written on either cassette or diskette. To get a full 43-note melody on cassette, it will have to be longer than five minutes. If you choose to record on diskette, be sure your diskette is in disk drive one.

Expanding the Program

This kind of program lends itself to personalizing, adapting and modifying. As amazing as the 99/4A is, it is still a microcomputer with the BASIC interpreter. If you ask it to do too much during play and display, your sound will drag.

If you have memory expansion (the program, as is, only leaves about 200 bytes free if the disk system is connected), you could add some other editing features. You could, for example, devise a way to write tunes longer than 43 notes, or a way to handle accidentals. You could even add a treble clef graphic to make your *Music Magic* versions look more authentic. If you are successful with any of these innovations let us all know in a Letter-to-the-Editor! *Music Magic* is another example of what can happen when you lend your inventive mind to your computer—the two of you can make beautiful music together.

MUSIC MAGIC
Explanation of the Program

Line Nos.	
100-170	Program header.
180-400	Redefine characters and set up arrays.
410-430	New or saved song menu.
440-460	Select cassette or disk option to retrieve a song.
470-480	Code for inputting song.
790-910	Song playback and note display.
920-1020	Option menu.
1030-1060	Cassette or disk option for recording a song.
1070-1100	Draw staff subroutine.
1110-1220	Save song subroutine.
1230-1320	Play recorded song subroutine.
1330-1340	Error recovery.
1350-1410	Menu display subroutine.
1420-1570	Sharps and flats subroutine.
1580-1800	Note printing subroutine.
1810-1850	Adjust notes for sharp or flat.
1860	Eliminate deleted note from the screen.

Continued on p. 54

Just Assemble Melody . . . from p. 35

Using the Machine Language Routine

This program was intended to facilitate the construction of sound lists for Assembly Language applications, but it may also be used in conjunction with TI BASIC programs if some care is exercised. Because the sound list will play in its entirety, even while the BASIC interpreter continues, your program can go on to other things while the music plays. However, because TI BASIC has no idea what is going on, there may be unpredictable side effects, especially when BASIC is manipulating strings or displaying graphics on the screen. [See the Editor/Assembler manual, page 312, for more information.—Ed.] Any CALL SOUND statement will stop the execution of a sound list. Here is a sample program which will play a sound list over and over until a key is pressed, then interrupt it (much like the techniques used in TI's *Tombstone City*):

```
100 CALL LINK("PLAY")
110 CALL PEEK(-31794,N)       Is it done?
120 IF N=0 THEN 100           Yes, play it again.
130 CALL KEY(0,K,S)           Key pressed?
140 IF S=0 THEN 110           No, check sound again.
150 CALL SOUND(-1,-1,30)      Stop the sound list.
 .
 . program continues
 .
```

The PEEK statement (line 110) reads a byte at >83CE. This will be zero when the interrupt routine has finished the sound list.

You can save and reload the PLAY subprogram and any sound list you generate with EASYBUG. Be sure to save everything from >7000 to >7FFF. This ensures that you have the entry point to the PLAY subprogram, and can play the sound list using the short program segment above.

Here, in the form used with the *Line-by-Line Assembler*, is an Assembly Language listing of the "PLAY" subprogram that the *Music Assembler* loads:

```
ST   EQU    >837C
     AORG   >7FBE
TA   DATA   0             Any value poked into this address
                          will be used as the address of the
                          sound list.
H1   BYTE   >01
     EVEN

PL   MOV    @TA,R1        Get table length pointer.
     MOV    R1+,R2        R1 is now start of sound list.
     MOV    R2,@>830C     Copy length to GPL parameter
                          address.

     BLWP   @>6018        Get string space routine.
     DATA   >0038         Get address of allocated space.
     MOV    @>831C,R0     Move sound list to VDP RAM.
     BLWP   @>6028

     LIMI   0             Play list
     MOV    R0,@>83CC
     SOCB   @H1,@>83FD
     MOVB   @H1,@>83CE
     LIMI   2
     CLR    @ST           Return to BASIC.
     RT
     TEXT   'PLAY
     DATA   PL
     AORG   >701C
     DATA   >7FF8, >7FF8
     END
```

It would be possible to put the program title in the REF/DEF table in this way because the program begins at >7FBE and uses up all but the last eight bytes of memory. The last two lines set *both* the First *and* Last Free Addresses in Medium Memory to >7FF8. The Last Free Address indicates the start of the REF/DEF table.

list now resides in memory. The first word of the sound list tells its length. This is used by the PLAY subprogram when allocating memory in VDP RAM for the list. It is a good idea to write these numbers down for future reference. To hear the music again, type CALL LINK("PLAY").

If you take a closer look at the sound statements above, you may notice that fields 8 and 9 (the noise specification and volume) are missing from lines 2-4, but that the noise continues. This is because the sound, once started, continues until it is explicitly stopped. The *Music Assembler* assumes that you want the sound of channel X to continue unless you tell it otherwise. TI BASIC, you will remember, works slightly differently, turning off all sound as it finishes each CALL SOUND statement. So a program like this:

```
100 FOR X=1 TO 10
110 CALL SOUND(500,440,2)
120 NEXT X
```

makes a slight ticking sound instead of a continuous tone.

Consider the following revision of Sound List 1:

Sound List 2

(1) 1000.A1.15.C#2.15.E2.25.-5.20.
(2) 1000. . . D2.15.F#2.15.-5.30.
(3) 1000. . . . . . G2.15.
(4) 1000.D2.15.F#2.15.A3.15.

Line 1 is unchanged. If you look at line 2 in both lists, you will see two differences. First of all, in Sound List 2 the noise processor is turned off by the specification of the maximum attenuation (30). Since the noise is never again specified in this list, the *Music Assembler* assumes that it should stay off. Secondly, instead of fields 2 and 3 repeating A1.15., as in line 1, the extra periods tell channel 1 to continue what it was doing. The positions of the periods indicate which channels to continue.

Look now at line 3. Which channels change, and which ones stay the same? Remember, two periods are needed to specify a blank field. At line 4, every note changes, so every field is filled except for the noise fields. Here is another example, this time in two-part counterpoint:

Sound List 3

(1) 2000.C1.8.
(2) 1000. . .A1.8.
(3) 1000.F1.8.
(4) 1000. . .G1.8.
(5) 1000.E1.8.
(6) 2000.F1.8.A1.8.C2.8

Because the top voice (on channel 2) does not enter until line 2, it is not specified. The program ensures that it continues what it was doing before (i.e., nothing). In lines 2 and 4 the lower voice (channel 1) holds through, indicated by the two empty fields. Notice that the upper voice will continue in line 5.

Unlike TI BASIC, Assembly Language does not accept negative frequency values as a means of interrupting a sound list. Instead, you may turn all three channels (plus noise) on and off at will, each independently of the others. For example, Sound List 4 creates a syncopated effect in the middle voice by turning it on and off while the other voices continue.

Sound List 4

(1) 1000.A#3.0.D3.0.F3.0.
(2) 250. . .D3.30.
(3) 250. . .D3.0.
(4) 250. . .D3.30.
(5) 250. . .D#3.0.
(6) 1000.A3.0.C3.0.

The *Music Assembler* can detect most errors, but not all. If a note is specified for a duration of zero, the interrupt routine in the console will take this as the end of your sound list. Any notes currently sounding will continue to sound. Another problem can arise when the attenuation field is blank:

200.C#2. .A#5.4.

This line may assemble, but perhaps not the way you intended.

After a little experimentation with various values in the fields, you will know how to cover the whole range of sound available in TI BASIC. But you will then be able to use the TI-99/4A's sound generator somewhat differently from the way it's used in TI BASIC (see box). This will add yet another dimension to the possibilities your Home Computer offers.

NOTE DESIGNATIONS

Note	Frequency	Note	Frequency	Note	Frequency
A0	110.00	G#2	415.30	G4	1567.98
A#0	116.54	A2	440.00	G#4	1661.22
B0	123.47	A#2	466.16	A4	1760.00
C1	130.81	B2	493.88	A#4	1864.66
C#1	138.59	C3	523.25	B4	1975.53
D1	146.83	C#3	554.37	C5	2093.00
D#1	155.56	D3	587.33	C#5	2217.46
E1	164.81	D3	622.25	D5	2349.32
F1	174.61	E3	659.26	D#5	2489.02
F#1	185.00	F3	698.46	E5	2637.02
G1	196.00	F#3	739.99	F5	2793.83
G#1	207.65	G3	783.99	F#5	2959.96
A1	220.00	G#3	830.61	G5	3135.96
A#1	233.08	A3	880.00	G#5	3322.44
B1	246.94	A#3	932.33	A5	3520.00
C2	261.63	B3	987.77	A#5	3729.31
C#2	277.18	C4	1046.50	B5	3951.07
D2	293.66	C#4	1108.73	C6	4186.01
D#2	311.13	D4	1174.66	C#6	4434.92
E2	329.63	D#4	1244.51	D6	4698.64
F2	349.23	E4	1318.51	D#6	4978.03
F#2	369.99	F4	1396.91	E6	5274.04
G2	392.00	F#4	1479.98	F6	5587.65

Middle C = 262.

Music Assembler
Explanation of the Program

Line nos.	
100-160	Program header.
170-220	Define functions.
230-260	Define arrays.
270-650	Data to define notes.
660-730	Data which defines machine language routine.
740-890	Set up program.
900-1310	Main program loop.
1320-1830	Make sound list.
1840-1970	Reset memory address pointers.
1980-2230	Routine to call machine language routine and play sound list.
2240-3010	Subroutines.
2240-2360	Find note input.
2370-2530	Accept sound list entry from keyboard.
2540-2640	Generate a hexadecimal address.
2650-2840	Check for values out of range.
2850-2890	Out of memory message.
2900-3020	Check machine language routine data.

TI-99/4A

```
100 REM *******************
110 REM * MUSIC ASSEMBLER *
120 REM *******************
130 REM  BY CLEON CHAPEN
140 REM  HOME COMPUTER MAGAZINE
150 REM  VERSION 4.1.1
160 REM  TI BASIC, TI MINI MEMORY
170 REM * MINI MEMORY
180 REM * FUNCTIONS
190 REM *
200 DEF H(N)=INT(N/16)
210 DEF L(N)=N-H(N)*16
220 DEF CODE(N)=INT(111860.8/N+.5)
230 REM *
240 REM * ARRAYS
250 REM *
260 DIM SD(8),NOS(68),FRE(68)
270 REM *
280 REM * DATA BLOCK
290 REM *
300 REM * PRE-DEFINED NOTES
310 REM *
320 DATA A#0,116.54,A0,110
330 DATA B0,123.47,C#1,138.59
340 DATA C1,130.81,D#1,155.56
350 DATA D1,146.83,E1,164.81
360 DATA F#1,185,F1,174.61
370 DATA G#1,207.65,G1,196
380 DATA A#1,233.08,A1,220
390 DATA B1,246.94,C#2,277.18
400 DATA C2,261.63,D#2,311.13
410 DATA D2,293.66,E2,329.63
420 DATA F#2,369.99,F2,349.23
430 DATA G#2,415.30,G2,392
440 DATA A#2,466.16,A2,440
450 DATA B2,493.88,C#3,554.37
460 DATA C3,523.25,D#3,622.25
470 DATA D3,587.33,E3,659.26
480 DATA F#3,739.99,F3,698.46
490 DATA G#3,830.61,G3,783.99
500 DATA A#3,932.33,A3,880
510 DATA B3,987.77,C#4,1108.73
520 DATA C4,1046.5,D#4,1244.51
530 DATA D4,1174.66,E4,1318.51
540 DATA F#4,1479.98,F4,1396.91
550 DATA G#4,1661.22,G4,1567.98
560 DATA A#4,1864.66,A4,1760
570 DATA B4,1975.53,C#5,2217.46
```

TI-99/4A

```
580 DATA C5,2093,D#5,2489.02
590 DATA D5,2349.32,E5,2637.02
600 DATA F#5,2959.96,F5,2793.83
610 DATA G#5,3322.44,G5,3135.96
620 DATA A#5,3729.31,A5,3520
630 DATA B5,3951.07,C#6,4434.92
640 DATA C6,4186.01,D#6,4978.03
650 DATA D6,4698.64,E6,5274.04,F6,5587.
    65
660 REM *
670 REM * MACHINE ROUTINE
680 REM *
690 DATA 1,0,192,96,127,190,192,177
700 DATA 200,2,131,12,4,32,96,24,0,56,1
    92,32,131,28,4,32,96,40,3,0,0,0,200
710 DATA 0,131,204,248,32,127,192,131,2
    53,216,32,127,192,131,206
720 DATA 3,0,0,2,4,224,131,124,4,91
730 DATA 80,76,65,89,32,32,127,194
740 REM *
750 REM * SETUP
760 REM *
770 CALL CLEAR
780 PRINT TAB(7);"MUSIC ASSEMBLER":::::
    ::::
790 FOR I=0 TO 68
800 READ NOS(I),FRE(I)
810 NEXT I
820 GOSUB 2890
830 CALL CLEAR
840 D$="."
850 H$="0123456789ACBDEF"
860 CALL PEEK(28700,H1,L1)
870 FFM=H1*256+L1
880 LC=FFM+2
890 CALL CLEAR
900 REM *
910 REM * MAIN LOOP
920 REM *
930 GOSUB 2400
940 IF FL=99 THEN 1870
950 OPTR=0
960 FOR I=0 TO 8
970 SD(I)=0
980 NEXT I
990 FOR I=0 TO 8
1000 NPTR=POS(IS,DS,OPTR+1)
1010 IF NPTR=0 THEN 1050
```

TI-99/4A

```
1020 IF OPTR+1=NPTR THEN 1290
1030 S$=SEG$(I$,OPTR+1,NPTR-OPTR-1)
1040 GOTO 1070
1050 S$=SEG$(I$,OPTR+1,5)
1060 IF (S$=" ")+(S$="     ")THEN 1370
1070 IF I=7 THEN 1130
1080 IF (I=1)+(I=3)+(I=5)THEN 1090 ELSE
     1190
1090 IF (ASC(S$)>64)*(ASC(S$)<72)THEN 11
     00 ELSE 1120
1100 GOSUB 2290
1110 GOTO 1300
1120 IF (ASC(S$)>47)*(ASC(S$)<58)THEN 11
     90
1130 IF SEG$(S$,1,1)<>"-" THEN 1240
1140 FOR J=2 TO LEN(S$)
1150 IF (ASC(SEG$(S$,J,1))>47)*(ASC(SEG$
     (S$,J,1))<58)THEN 1170
1160 GOTO 1240
1170 NEXT J
1180 IF VAL(S$)>-9 THEN 1260 ELSE 1240
1190 FOR J=1 TO LEN(S$)
1200 IF (ASC(SEG$(S$,J,1))>47)*(ASC(SEG$
     (S$,J,1))<58)THEN 1220
1210 GOTO 1240
1220 NEXT J
1230 GOTO 1260
1240 PRINT :"* BAD VALUE: ";S$:" TRY AG
     AIN...":
1250 GOTO 930
1260 SD(I)=VAL(S$)
1270 GOSUB 2680
1280 GOTO 1300
1290 SD(I)=0
1300 OPTR=NPTR
1310 NEXT I
1320 REM *
1330 REM * MAKE SOUND LIST
1340 REM *
1350 REM * DURATION
1360 REM *
1370 DU=INT(SD(0)*255/4250)
1380 DU$=CHR$(DU-(DU=0))
1390 REM *
1400 REM * SOUND GENERATORS
1410 REM *
1420 G1$=""
1430 G2$=""
1440 G3$=""
1450 NC$=""
1460 NA$=""
1470 IF SD(1)=0 THEN 1510
1480 V=CODE(SD(1))
1490 G1$=CHR$(128+L(V))&CHR$(H(V))
1500 G1$=G1$&CHR$(144+INT(SD(2)/2))
1510 IF SD(3)=0 THEN 1550
1520 V=CODE(SD(3))
1530 G2$=CHR$(160+L(V))&CHR$(H(V))
1540 G2$=G2$&CHR$(176+INT(SD(4)/2))
1550 IF SD(5)=0 THEN 1600
1560 V=CODE(SD(5))
1570 G3$=CHR$(192+L(V))&CHR$(H(V))
1580 G3$=G3$&CHR$(208+INT(SD(6)/2))
1590 REM *
1600 REM * NOISE CONTROL
1610 REM *
1620 IF SD(7)=0 THEN 1730
1630 IF SD(7)<-4 THEN 1660
1640 TY=0
1650 GOTO 1680
1660 TY=4
1670 SD(7)=SD(7)+4
1680 NC$=CHR$(224+TY+ABS(SD(7))-1)
1690 NA$=CHR$(240+INT(SD(8)/2))
1700 REM *
1710 REM * FINISH AND LOAD
1720 REM *
1730 SOUND$=G1$&G2$&G3$&NC$&NA$&DU$
1740 L$=CHR$(LEN(SOUND$)-1)
1750 SOUND$=L$&SOUND$
1760 LF=LF+LEN(SOUND$)
1770 SAVELC=LC
1780 FOR I=1 TO LEN(SOUND$)
1790 CALL LOAD(LC,ASC(SEG$(SOUND$,I,1)))
1800 LC=LC+1
1810 IF LC>32701 THEN 2850
1820 NEXT I
1830 GOTO 930
1840 REM *
1850 REM * RESET POINTERS
1860 REM *
1870 CALL LOAD(LC,4,159,191,223,255,0)
1880 LF=LF+6
1890 LC=LC+6
1900 IF LC/2=INT(LC/2)THEN 1920
1910 LC=LC+1
1920 FH=INT(LF/256)
1930 FL=LF-FH*256
1940 CALL LOAD(FFM,FH,FL)
1950 LCH=INT(LC/256)
1960 LCL=LC-LCH*256
1970 CALL LOAD(28700,LCH,LCL)
1980 REM *
1990 REM * PLAY LIST?
2000 REM *
2010 CALL CLEAR
2020 PRINT "* ASSEMBLY COMPLETED"
2030 INPUT "* PLAY LIST? (Y/N) ":I$
```

Continued on p. 40

Just Assemble Melody

TI-99/4A

```
2040 IF ASC(I$)=89 THEN 2070
2050 IF ASC(I$)=78 THEN 2210
2060 GOTO 2010
2070 RESTORE 690
2080 LOC=32704
2090 CALL LOAD(LOC-2,H1,L1)
2100 PRINT "* LOADING MACHINE ROUTINE"
2110 FOR I=0 TO 63
2120 READ V
2130 CALL LOAD(LOC+I,V)
2140 NEXT I
2150 PRINT "* PLAYING"
2160 CALL LOAD(28702,127,248)
2170 CALL LINK("PLAY")
2180 PRINT :"* PRESS ANY KEY TO PROCEED"
2190 CALL KEY(0,K,S)
2200 IF S=0 THEN 2190
2210 PRINT :"* COMPLETED SOUND TABLE
        LOCATED AT ADDRESS";FFM
2220 GOSUB 2570
2230 STOP
2240 REM *
2250 REM * SUBROUTINES
2260 REM *
2270 REM * SEARCH FOR NOTE
2280 REM *
2290 FOR J=0 TO 68
2300 IF NOS(J)=SS THEN 2350
2310 NEXT J
2320 PRINT :"* UNKNOWN NOTE NAME: ";S$::
2330 INPUT "    RETYPE NOTE: ":S$
2340 GOTO 2290
2350 SD(I)=FRE(J)
2360 RETURN
2370 REM *
2380 REM * KEYBOARD INPUT
2390 REM *
2400 IF OP THEN 2430
2410 PRINT "* ENTER SOUND LISTS":"* ENTE
R NULL STRING TO END"::
2420 OP=-1
2430 INPUT I$
2440 IF I$="" THEN 2450 ELSE 2470
2450 FL=99
2460 RETURN
2470 IF I$="REDO" THEN 2480 ELSE 2500
2480 LC=SAVELC
2490 GOTO 2430
2500 IF SEG$(I$,LEN(I$),1)="." THEN 2530
2510 PRINT :"* FINAL PERIOD MISSING.
        TRY AGAIN...":: 
2520 GOTO 2400
2530 RETURN
2540 REM *
2550 REM * MAKE HEX ADDRESS
2560 REM *
2570 A$=""
2580 FOR D=12 TO 0 STEP -4
2590 N=INT(FFM/(2^D))
2600 FFM=FFM-N*(2^D)
2610 A$=A$&SEG$(H$,N+1,1)
2620 NEXT D
2630 PRINT "   (HEX >";A$;")"
2640 RETURN
2650 REM *
2660 REM * CHECK RANGES
2670 REM *
2680 ON I+1 GOTO 2690,2720,2750,2720,275
0,2720,2750,2780,2750
2690 MIN=1
2700 MAX=4250
2710 GOTO 2800
2720 MIN=110
2730 MAX=44733
2740 GOTO 2800
2750 MIN=0
2760 MAX=30
2770 GOTO 2800
2780 MIN=-8
2790 MAX=-1
2800 IF (SD(I)<=MAX)*(SD(I)>=MIN)THEN 28
10 ELSE 2820
2810 RETURN
2820 PRINT :"* BAD VALUE: ";SD(I):"   ";S
TR$(MIN);" TO ";STR$(MAX):"   IS THE
  VALID RANGE"::
2830 INPUT "    RETYPE BAD NUMBER: ":SD(I)
2840 GOTO 2800
2850 REM *
2860 REM * OUT OF MEMORY
2870 REM *
2880 PRINT ::::"**** MEMORY FULL ****"::
:::
2890 REM *
2890 END
2900 REM *
2910 REM * CHECK MACHINE
2920 REM * DATA - DELETE
2930 REM * WHEN DATA IS
2940 REM * VERIFIED
2950 REM *
2960 FOR I=0 TO 63
2970 READ V
2980 CHECK=CHECK+V
2990 NEXT I
3000 IF CHECK=5790 THEN 3010 ELSE 3020
3010 RETURN
3020 PRINT ::"*** ERROR IN MACHINE ROUTI
NE"::
```

HCM

A Detailed Look Inside the Peanut's Shell

How It Works . . . What It Can Do . . . And Who It's For

By Gary M. Kaplan and William K. Balthrop

When the IBM PCjr was announced this past November, the event drew all of the pomp and attention of a royal birth. Old Big Blue, King of Greater Computerland, finally showed off his youngest progeny, and christened him PCjr. Inhabitants of the kingdom had speculated about a new arrival ever since rumors (and even doubts) of the royal conception started flying nearly one year before. Now a new round of speculation would start up—about a possible rivalry between two famous royal siblings.

PC, the King's first-born, came into the computer world in July of 1981, in a more peaceful and stable era. The King had turned young PC loose in the corporate lands that lay to the west—lands populated by tribes of businessmen newly aware of potential productivity gains with personal computers. Capitalizing on his inherited aggressiveness, swiftness, and muscle, he quickly occupied this fertile frontier. And by the time his younger brother was born, PC had successfully brought a sizable chunk of the New Businessland Territories under his personal flag.

The King's second son, PCjr, was born into a harsher, more cruel world—a world that saw the First Home Computer War take a devastating toll on corporate balance sheets and consumer trust. The wizened King had carefully observed the persistent march of smaller kingdoms as they pushed into the Home Computer Territories—those vast, mostly uncharted lands that lay to the mysterious east. He saw them succumb to repeated sales famines, plagues of inaccurate delivery schedules, and the chaos of faulty quality control. He saw many of their brave crusaders cut down by fierce attacks from marauding bands of retailing and manufacturing price-slashing brigands. It was a land where only the most hardy could survive.

The old King wisely waited until the dust had settled on these bloody battlefields before he sent out the new boy-Prince, PCjr, into the embroiled Territories. The second son's quest: to advance his banner far and wide and bring home the holy grail of dominant market share . . .

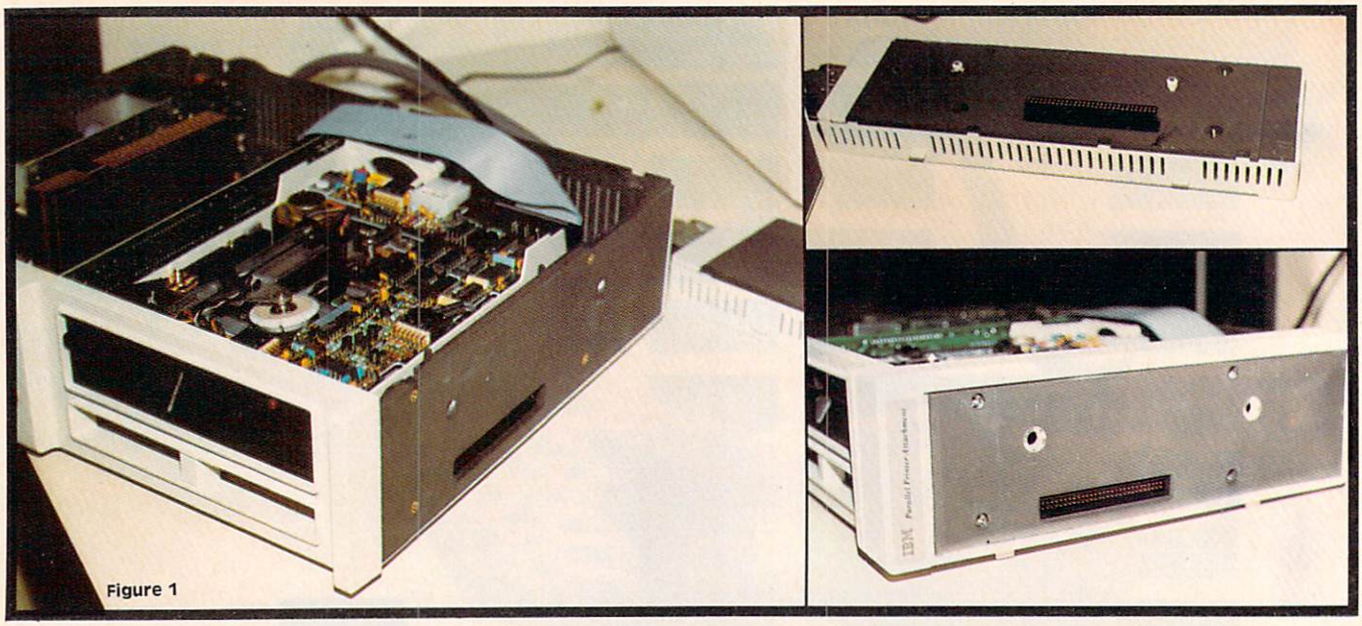


Figure 1

Fig. 1 - A view of the PCjr's right side with the cover panel removed, exposing the I/O expansion bus. Here, the IBM Parallel Printer Interface is shown being attached. A feed-through I/O bus connector on each side-mounted peripheral allows additional peripherals to be snapped together on the I/O bus in freight-train style with an "integrated look". The cover panel is then put back in place.

In a nutshell, the IBM PCjr (code named "Peanut" during its development) is a computing machine with a flexible design architecture. This built-in flexibility allows it to be simultaneously "packaged" as an educational aid, an entertainment device, a home productivity tool, a communications link, and a lower-cost office complement to the IBM PC. It also means that the PCjr can be readily converted into an electronic "black box"—a fully "customizable" machine-servant to humankind.

The architecture of the PCjr is open and straightforward: The machine is constructed of standard, well-proven technology—most of it carried down from its big brother, the PC. And (with the exception of one custom video chip) it is put together with common "off-the-shelf" components and sub-assemblies.

The PCjr is not, however, just a scaled down, smaller version of the PC. In fact, it takes advantage of larger-scale microchip integration that eliminates the need for some adapter cards and improves upon the PC's technical performance and ease of use. For example, the game controller, serial port, light pen interface, enhanced color graphics capabilities, and brand new complex sound generation abilities are located on the computer's main circuit board. In addition, its new (for IBM) capacity for

Continued on p. 44

IN A NUTSHELL

System Features of the PCjr:

The Processor & Memory System:

- Intel 8088 Central Processing Unit (CPU) at 4.77 Mhz clock speed—a 16-bit microprocessor with an 8-bit data bus.
- 64K-bytes Read Only Memory (ROM) containing Cassette BASIC, built-in keyboard tutorial, and power-on self-test diagnostics.
- 64K-bytes Random Access Memory (RAM) expandable to 128K-bytes on the systemboard; same expansion capability as IBM PC to 640K-bytes RAM via input/output (I/O) expansion bus connector. Memory shared between CPU and video display.

The Sound System:

- 3-channel-plus-noise sound generator with external audio output jack; built-in piezioelectric speaker for time-generated tones, alarm, and keyboard "click"; and audio input capability through cassette port and I/O expansion bus connector.

The Video Graphics System:

- Upward compatible with PC Color/Graphics Adapter running on the IBM PC.
- CRT Page Register with any 16K-byte region of main storage capable of being mapped to the display; updates one page while a different page is being displayed.
- Custom video chip used to read the data and format it for display; 16x4 palette allows quick screen color changes—with only the palette changing, not the data in memory. Includes a separate border-color register.
- 3 video output connectors allow for simultaneous cable hook-up.
  - — TV (with RF modulator); color or B/W
  - — RGBI direct drive hi-res color video
  - — Composite video for color, or monochrome with grey shades.

The Keyboard System:

- 62-key keyboard — detached type, cordless with infrared link and battery operation. Functionally equivalent to 83-key PC keyboard. Color-coded and programmable. Optional keyboard cord ($20). Optional keyboard overlays ($10 for set of 5).

The Input/Output (I/O) System:

- 2 64K-byte ROM cartridge slots.
- 60-pin bus expansion, light pen, serial @ 4800 baud max., plus reserved use.
- Game adapter on the system board for connecting either 2 joysticks or paddles. Optional joysticks ($40 each).
- Cassette Interface for cassette player (optional) — Adapter cable for cassette ($30).

Expansion System Options:

- 64K-bytes Memory and Display Expansion card to increase total system memory to 128K-bytes ($140).
- Parallel Printer attachment, hooks onto side panel ($99).
- Internal Modem card ($199).
- IBM PCjr diskette drive and controller card—360K-bytes, double sided, double density ($480). Operates with DOS 2.1.

Physical System:

- External transformer with 10-foot power cord.
- Weight of system box: less than 6 lbs; less than 9 lbs. with disk drive.
- Dimensions: 13.9" x 11.4" x 3.8"
- Entry Model: keyboard, console with 64K ($669).
- Enhanced Model: keyboard, console with 128K and disk drive; 80-column capability ($1269).

Fig. 2A — IBM Compact Printer

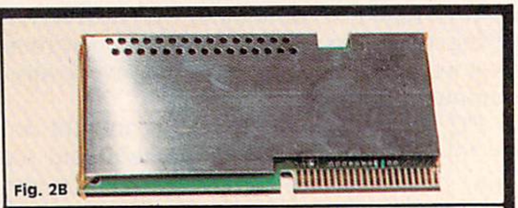


Fig. 2B

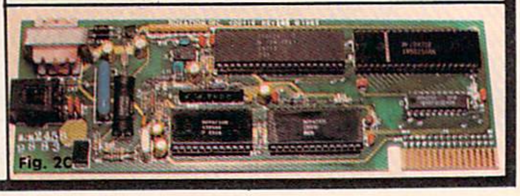


Fig. 2C

Fig. 2B — 64K Memory and Display Expansion card required for video graphic screen modes 5 & 6 (including 80-column text); (See Fig. 7)
Fig. 2C — Internal Modem directly connects to phone line; auto-dial, auto-answer, and auto-file updating capability when used with the Personal Communications Manager software package (see Fig. 16).

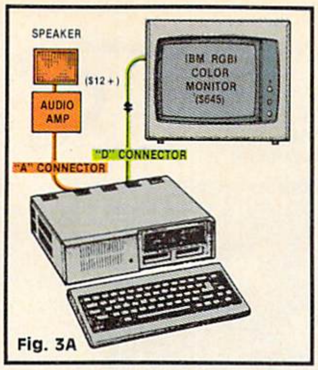


Fig. 3A

Notes: Supports all 40/80-column color/monochrome modes; monochrome text has grey shades; compatible RGBI monitor required for 4-color high resolution; no sound without external amp/speaker; "best" for color graphics; "good" for word processing/business. Extras: IBM cable ($20), standard audio cable with RCA plug ($3+).

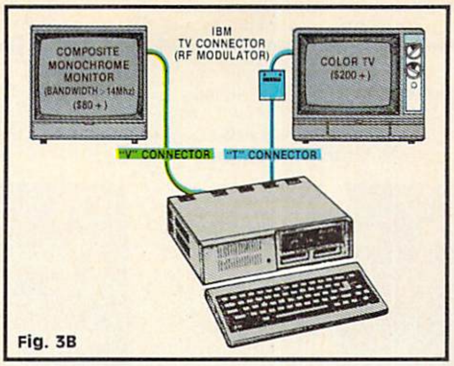


Fig. 3B

Notes: Different inputs allow simultaneous connections; supports all color/monochrome modes except 4-color at high resolution; "adequate" for color graphics; "best" for word processing/business. Extras: IBM TV connector ($30), standard audio/video cable with RCA plugs ($3+).

Fig. 3C

Notes: 40 columns max; low and medium resolution with 16 colors; "adequate" for color graphics. Extras: IBM TV connector ($30)

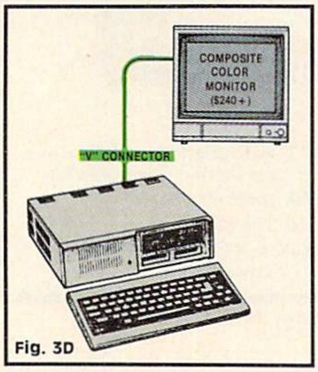
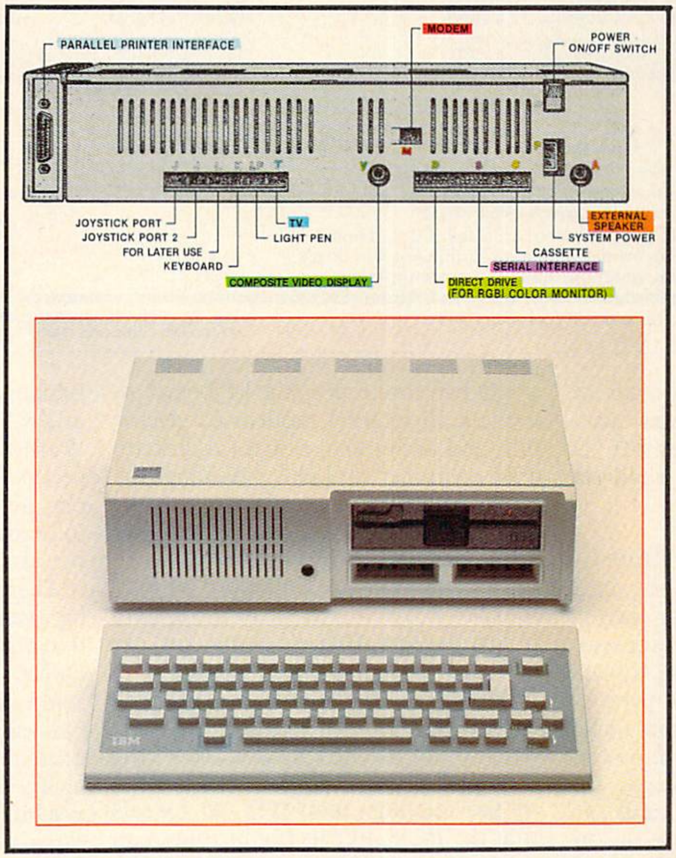


Fig. 3D

Notes: 40 columns max; low and medium resolution with 16 colors; "good" for color graphics; most brands with audio capability — otherwise, separate amp/speaker required (see Fig.3A). Extras: standard audio/video cable with RCA plugs ($3+)



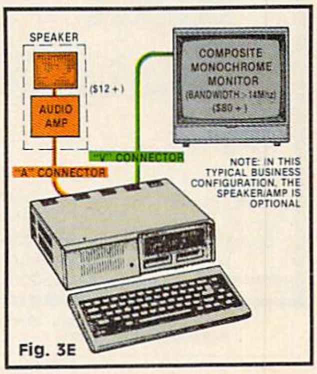


Fig. 3E

Notes: 80-column requires band width >14Mhz; grey shades; "best" for word processing, spread sheets, and business applications; most versatile in combination with color TV (see Fig.3B). Extras: Standard audio/video cable with RCA plugs ($3+).

Fig. 3F

Notes: IBM Compact Printer ($175 includes attached cable) operates at 1200-baud max. RS232 serial interface supports up to 4800 baud with compatible third-party printers. Extras: IBM Adapter Cable for Serial Devices ($25)

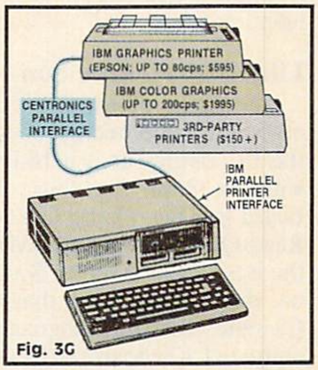


Fig. 3G

Notes: Parallel Printer Interface ($99) supports parallel ("Centronics-compatible") printers and plotters and other output devices. Extras: Common parallel cable with DB25 connectors ($25+), or IBM parallel cable ($55).

Figure 3
Options for PCjr System Configurations

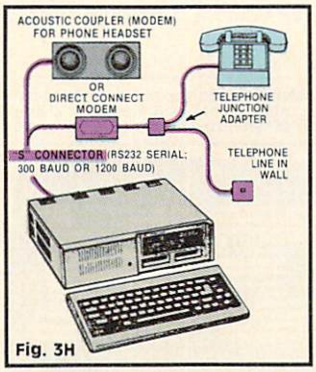


Fig. 3H

Notes: External modem is practical if already have 300-baud modem and don't need the serial port for printer, or if needs require 1200-baud communication with a higher-speed modem. Extras: IBM Adapter Cable for Serial Devices ($25), standard telephone junction adapter ($3+)

How to Buy a PCjr System

As you can see from the diagrams and annotations on this page, it's not just a simple matter of "hooking up" your PCjr right out of the box and starting to compute. Choosing the appropriate monitor(s) and modem/printer combination is a non-trivial matter. Our advice is to read the entire PCjr article, study this page, and give it plenty of thought. Then, you'll be ready to choose wisely.

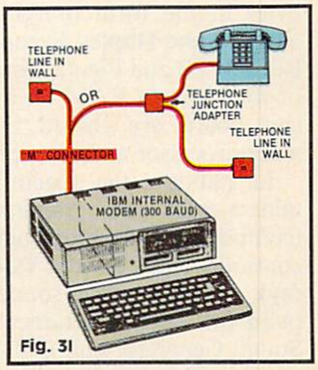


Fig. 3I

Notes: IBM Internal Modem is practical if serial port is needed for a printer or other peripheral, and if 300 baud is a satisfactory rate of communication. Works with rotary and touch-tone telephones. Extras: Standard telephone junction adapter ($3+).

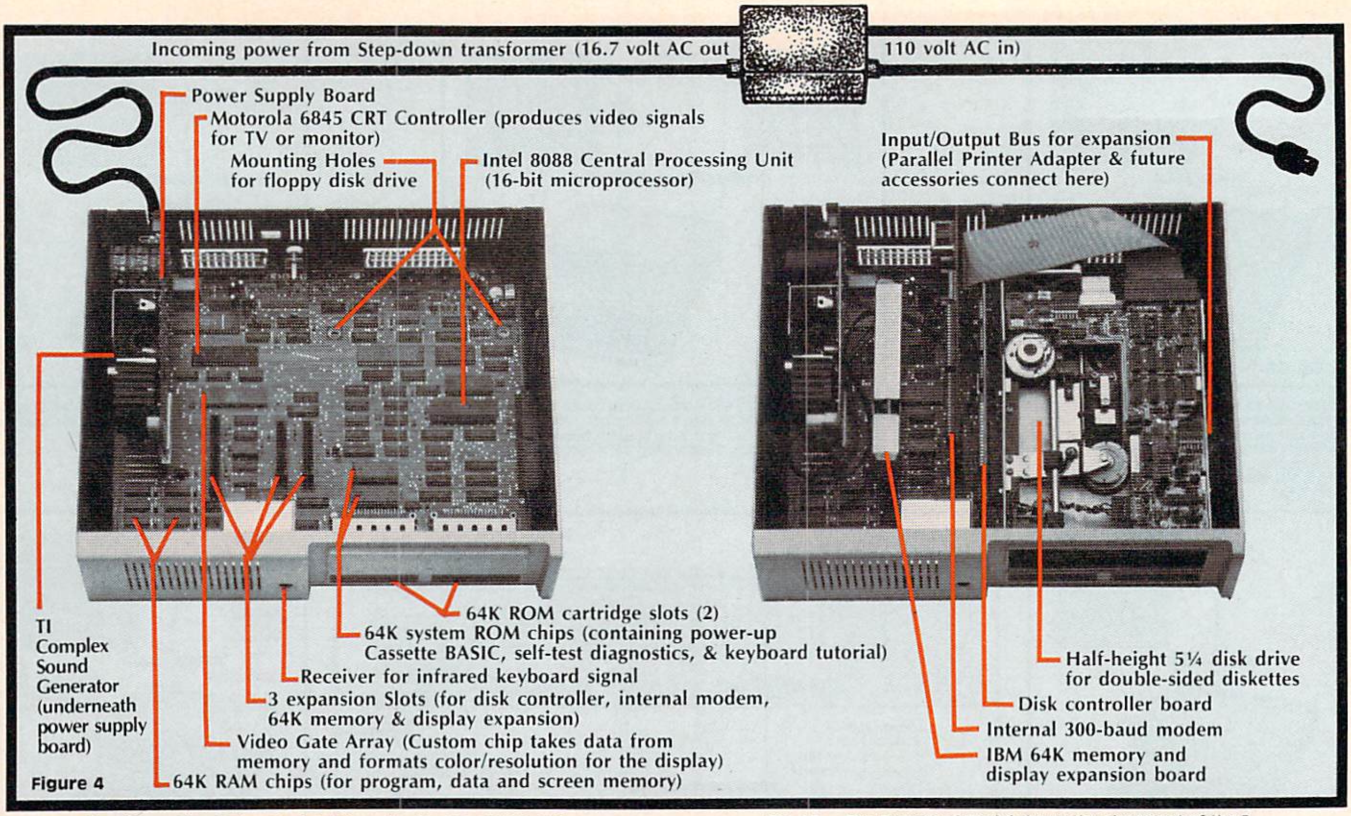


Figure 4

Fig. 4A — The Entry Model PCjr with its connectors and expansion slots clearly visible.

Fig. 4B — The Enhanced Model shows the placement of the 3 expansion cards and disk drive assembly.

accepting and utilizing 128K-bytes of plug-in ROM cartridge software opens the way for an infinite number of machine personalites, as we shall see later.

Like Father, Like Son—Almost

The brain of the PCjr is the Intel 8088, the same microprocessor used in the PC family. It is a 16-bit processor with an 8-bit data bus. The system board [see Fig. 4] also has 64K-bytes of Read-Only Memory (ROM) containing the Basic Input/Output System, power-on diagnostics, user diagnostics, the Cassette BASIC language, and an animated *Keyboard Adventure* on the use of the 62-key keyboard. There are also 64K-bytes of Random Access Memory (RAM) on the system board, plus a connector for an additional 64K-bytes in the form of an IBM 64KB Memory and Display Expansion Option [see Fig. 2B and Fig.4]. The 128K-bytes of total system RAM on the PCjr circuit board are shared by the 8088 microprocessor and the video circuitry.

In addition, the system board contains a color video graphics subsystem (composed of the Motorola 6845 CRT controller and a custom Video Gate Array), a multi-channel sound subsystem (with the Texas Instruments Complex Sound Generator chip), an I/O expansion bus [see also Fig. 1], a 128K-byte ROM cartridge interface, plus six accessory interfaces (a game adapter for joysticks/paddles, an RS232 serial port, a light pen connection, the keyboard interface, an internal modem connector slot, and an internal slot for a diskette drive controller card plus mountings for a half-height 5¼-inch diskette drive).

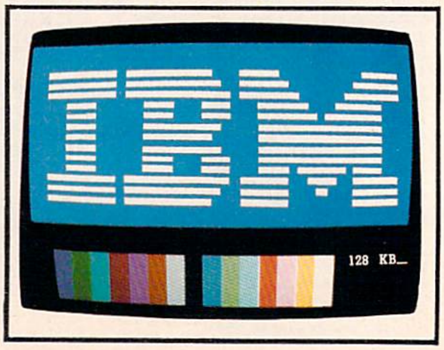
The Memory Is Not Peanuts

The limit (in the Enhanced Model) of 128K-bytes of RAM on the system board is just that—*a limit on the system board.* The design engineers at IBM have actually provided the I/O expansion bus with most of the same lines found on the I/O channel of the models PC and XT. An examination of the memory map [Fig. 6] reveals that the PCjr, like its big brother, can address up to 640K-bytes of RAM. For such an expansion, the memory chips would be situated on a separate board connected to the PCjr's right side—in the same way that the IBM Parallel Printer Interface is connected [Fig. 1].

It is feasible that just about any feature which can be added to the IBM PC and XT can also be added to the PCjr. Of course, there are restrictions on the size, amount of power required, and manner of implementation, but most of these can be worked around. The signals passing through this I/O channel have sufficient drive to support five expansion adapters (in addition to the internal modem and diskette controller), each with one standard TTL load per attachment. This means that users will see *many* expansion peripherals produced for the PCjr. Besides extra RAM boards, diskette drives, bubble memory storage, and Winchester fixed-disk drives, we can expect to see hundreds of exotic new add-ons, as well as separate expansion boxes to house and power many of the larger boards at the same time.

There are two important differences between the PCjr expansion bus and that of the PC: The PCjr's bus lacks Direct Memory Access (DMA) and some interrupts (already used for the built-in interfaces). Although the PCjr does not have DMA, it does have the ability to add it (and allow its diskette system to be run under it) with external expansion. This would then permit applications programs to *overlap* diskette I/O operations as is done on its bigger brother, the PC. Having DMA lets a computer accept typewritten input or communicate *while* it is reading from or

Fig. 5 — This master screen appears when the system is "powered up." After the resident memory is checked, the total number of system bytes appears in the window.



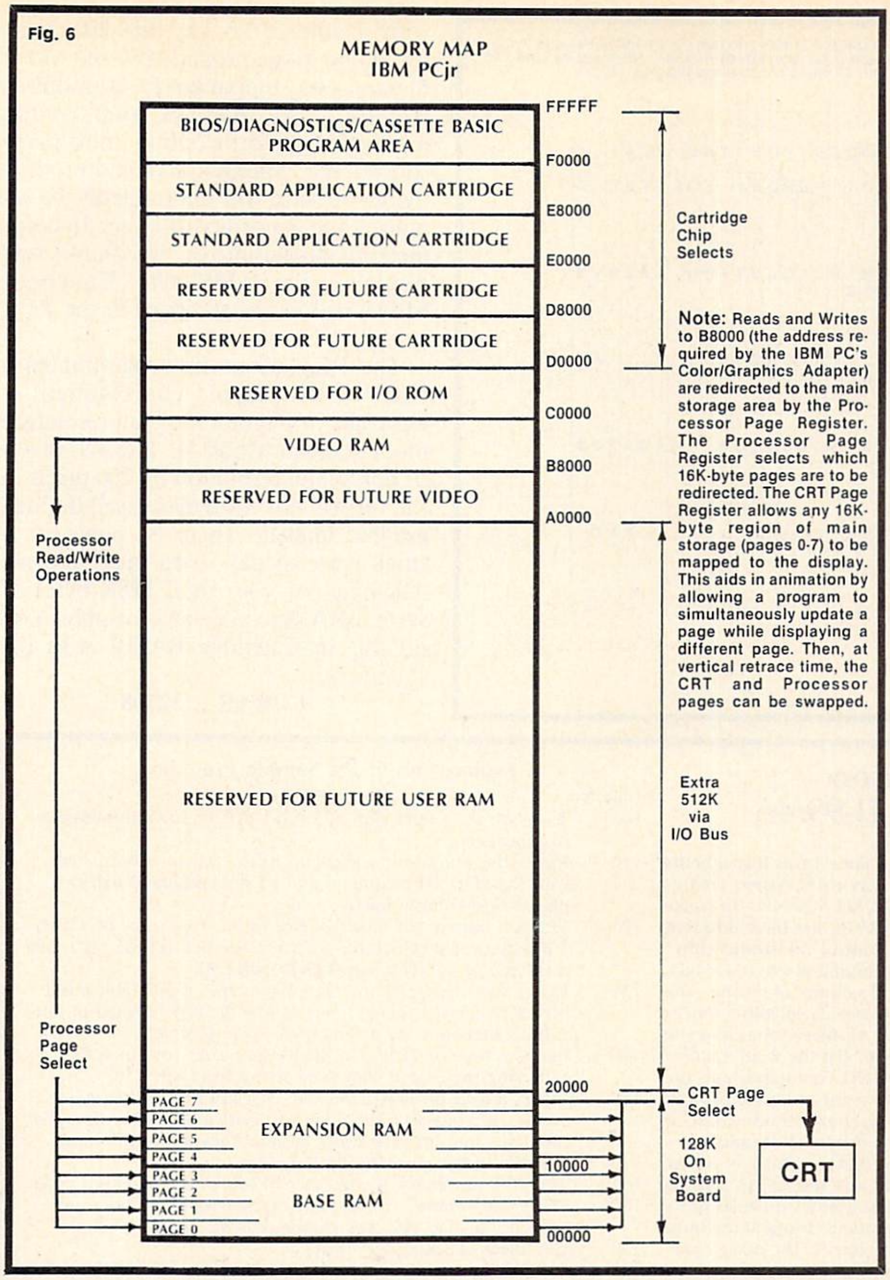


Fig. 6 MEMORY MAP IBM PCjr

writing to the diskette—resulting in faster throughput without noticeable "wait states." Although this lack of DMA is probably not too significant for most home users, accessory devices that add the capability will be in demand for more serious, business-oriented applications.

Saying It With Pictures

The color graphics capabilities of the PCjr are built-in, not added as an accessory as is done with the PC. The graphics implementation is an enhanced version of the PC's Color/Graphics Adapter card. The PCjr differs, however, in that system memory is *shared* between the processor and display. There are two distinct operating modes: (1) alphanumeric and (2) graphic [see Fig. 6]. These are further broken down into seven screen modes (two alphanumeric and five graphic)—each with a unique set of memory requirements, colors, and resolutions. [The resolution of a display is measured in picture elements called "pixels." On the PCjr's screen, each pixel is a small rectangle of light that is either turned on or off, or given one of the 16 unique colors. The resolution of a screen is expressed in dimensions of pixels horizontally by pixels vertically—Ed.]

The microprocessor and video display share the 64K-bytes of RAM on the PCjr system board, accessing the data 8 bits at a time. This supports the 40-column alpha mode and 3 of the graphics modes: 160 × 200 (low resolution) with 16 colors, 320 × 200 (medium resolution) with 4 colors, and 640 × 200 (high resolution) with 2 colors. This means that the Entry Model of the PCjr (for $669) *cannot* display 80 columns of text, medium resolution graphics with more than 4 colors, or high-resolution graphics with more than 2 colors.

If the 64KB Memory and Display Expansion Option (included with the diskette drive and controller board in the Enhanced Model for $1269) is installed on the PCjr, the memory is reconfigured so that all the *even* bytes reside on the system board, and all the *odd* bytes reside on the expansion card. This means that the data is accessed 16 bits at a time instead of just 8. With the PCjr's architecture, this "double fetch" [a discussion of bandwidth constraints is beyond the scope of this article—Ed.] allows it to support 80-column alpha, 320 × 200 (medium resolution) with 16 colors, and 640 × 200 (high resolution)

Fig. 7

Video Graphics Modes Available On The PCjr

Screen Mode	Resolution	Video Memory Required	Available Colors	64K Expansion Required	Cartridge BASIC Required	Compatible With IBM PC Color/Graphics Adapter
0	40/80 Column Alpha	2K/4K	16/16	NO/YES	NO/NO	YES/YES
* 1	320×200	16K	4	NO	NO	YES
2	640×200	16K	2	NO	NO	YES
3	160×200	16K	16	NO	YES	NO
* 4	320×200	16K	4	NO	YES	NO
5	320×200	32K	16	YES	YES	NO
6	640×200	32K	4	YES	YES	NO

*Note: Mode 4 (only available on PCjr) has a variable palette whereas Mode 1 (compatible with PC) does not.

Screen Mode: the video graphics mode set with the SCREEN command.

Resolution: the screen resolution. The first value on the chart is the number of horizontal pixels, and the second number is the number of vertical pixels.

Video Memory Required: buffer size specifies the amount of memory needed to use the mode specified. Mode 0 is a special case in which two values are given. The first value refers to a 40-column screen and the second to an 80-column screen.

Available Colors: the number in this column indicates the maximum number of colors which can be displayed on the screen at any one time.

64K Expansion Required: some modes require more memory to store the screen contents than others. In these cases (indicated by YES on the chart) the 64K Memory Expansion must be installed.

Cartridge BASIC Required: a YES indicates that this mode is unavailable through Cassette BASIC.

Compatible with IBM Color/Graphics Adapter: YES indicates compatibility with PC equipped with this board.

Fig. 8 — This listing demonstrates some of the new or modified graphic screen commands available in Cartridge BASIC using the 64KB Memory and Display Expansion card. The 32K of memory reserved by Line 140 is necessary for the high-resolution modes (SCREENs 5 and 6) called for in lines 180 and 260.

```
100 REM NEW SCREEN MODES FOR THE PCjr
110 REM
120 REM RESERVE 32K OF MEMORY FOR THE D
ISPLAY
130 REM
140 CLEAR ,,,32768!
150 REM
160 REM SELECT MEDIUM RESOLUTION (320x2
00) WITH 16 COLORS
170 REM
180 SCREEN 5
190 REM
200 REM SELECT A RED FOREGROUND (4), AN
D WHITE BACKGROUND (15)
210 REM
220 COLOR 4,15
230 REM
240 REM SELECT HIGH RESOLUTION (640x200
) WITH 4 COLORS
250 REM
260 SCREEN 6
270 REM
280 REM CHANGE PALETTE COLOR #2 TO RED
(4)
290 REM
300 PALETTE 2,4
310 END
```

with 4 colors. To see this last high-resolution mode properly (Screen Mode 6) requires a compatible RGBI monitor [see Fig. 3A]. Because most of the mountain of forthcoming third-party educational, entertainment and productivity software will undoubtedly be requiring the versatility of either 16-color medium resolution or 80-column text displays, the 128K-byte Enhanced Model will be the standard in the PCjr world.

The PCjr offers its programmers a rich development environment—especially in 16-color medium-resolution mode (Screen Mode 5). The 320 pixels of horizontal resolution by 200 pixels in the vertical direction allow well-defined graphic images. There is, however, a small price to pay—you must allocate 32K-bytes of your total 128K-bytes of system RAM to the video display. You do this in Cartridge BASIC with the statement:

CLEAR ,,,32768

A Sound Comparison Between the PCjr and TI-99/4A

Although the TI-99/4A and the IBM PCjr use the same Texas Instruments sound chip, there are major differences in the way each system creates sound or music. TI BASIC uses a single command—CALL SOUND—to create both sound effects and music. IBM PCjr Cartridge BASIC has three different commands—SOUND, NOISE, and PLAY—that control the sound chip.

The PCjr's SOUND command is similar to the TI command CALL SOUND. Both let you specify the frequency, duration, and volume of a tone. The difference is that with the TI command you can specify all three voices plus the noise channel within a single command. All three tones and the noise that are created will have the *same* duration. On the PCjr, *each* of the three voices is activated with a separate SOUND command, and the noise channel is activated with NOISE. This may seem awkward at first, but it has its advantages: Because the individual voices are set up with their own SOUND commands, each voice can have a *different* duration.

PCjr Cartridge BASIC has a third command—PLAY—which is unlike anything found in TI BASIC. The PLAY command lets you set up a string of sub-commands to compose music with the actual notes (instead of by frequency). A program can play three of these command strings at the same time, with each string controlling one of the three voices. The noise channel can't be directly inserted into the PLAY command, but if the music is playing in the background mode (e.g., the program continues while the music buffer plays), then the NOISE command can be used to mix noise with the three voices. The placement of the noise within the music chain of commands can be calculated by testing one of the three voices to determine the number of notes left in the buffer. When the program is creating sounds or playing music in the foreground mode, the statement that is creating the sound must finish executing before the program proceeds to the next statement.

Explanation of the Sample Program

Line No.	Purpose
100	Turns on the sound chip and turns off the internal piezoelectric speaker.
110	Places the sound in background mode. All sounds created with the SOUND command and PLAY command will be placed in the music buffer.
120	Places a tone in the music buffer with a frequency of 110Hz, a duration of 20 clock ticks (each clock tick is 18.5 ms.), and a volume of 15. The output is to voice #0.
130	Places the string of music commands into the default music buffer (voice #0). These notes follow the tone created in line 120 because they are on the same voice channel.
140	Places a tone of 220Hz in the music buffer for voice #0 following the PLAY command string from line 130.
150	Places a tone of 440Hz into the first position of the music buffer for voice #1. This tone will start at the same time that the tone from line 120 starts because they are in the same position in the music buffer.
160	Tests the buffer for the number of remaining unplayed notes.
170	When the number of notes left reaches four, the program branches to line 190. Subroutine returns; program halts.
180	Continues checking the buffer.
190	Places a tone of 440Hz in the music buffer for voice #1. This note will start to play when there are four notes left in the buffer for voice #0.
200	Places a tone of 880Hz into the buffer for voice #2. This tone will play at the same time as the tone from line 190 and the note G from voice #0.
210	Turns on the noise generator at the same time as the tones in lines 190, 200, and the note G in voice #0.
220	Exits the subroutine.

```
100 BEEP OFF: SOUND ON
110 PLAY "MB"
120 SOUND 110, 20, 15, 0
130 PLAY "C#DEF#GAB"
140 SOUND 220, 20, 15, 0
150 SOUND 440, 20, 15, 1
160 A=PLAY (3)
170 IF A=4 THEN GOSUB 190: END
180 GOTO 160
190 SOUND 440, 20, 15, 1
200 SOUND 880, 20, 15, 2
210 NOISE 1, 15, 20
220 RETURN
```

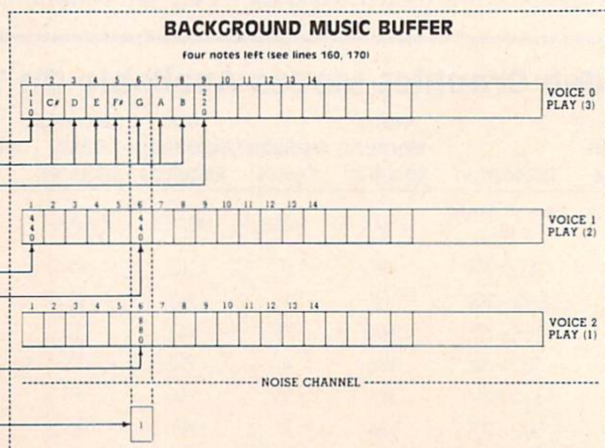


Fig. 10A — Cassette BASIC initial screen

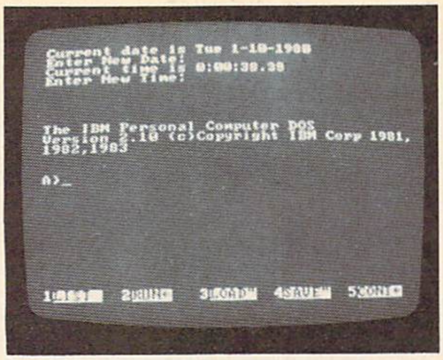


Fig. 10B — DOS 2.1 initial screen

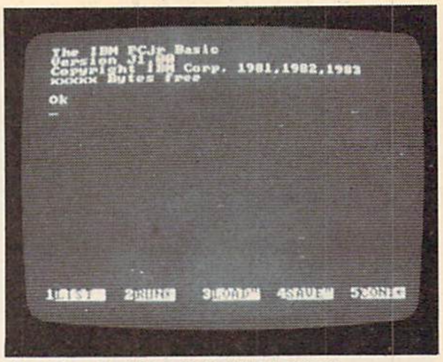


Fig. 10C — Cartridge BASIC initial screen

The labels across the bottom of each screen correspond to the first five function (FN) key-press sequences—e.g., pressing the FN key together with the number 2 key is the equivalent of typing in the letters R, U, and N, and then pressing the ENTER key. Up to 10 (in 80 column mode) user-defined keystroke sequences may be programmed in to appear in the bottom windows with the KEY command. The number of bytes appearing in screens 10A and 10C depend on system configuration (e.g., with Cartridge BASIC on the standard 64K-byte system, there are 42,210 bytes free and on the 128K-byte system there are 60,130 bytes free).

On the IBM PC, a *separate* bank of memory (16K-bytes) situated on the Color/Graphics Adapter card is used to map the color display. The PC is, however, limited to only 4 colors in medium resolution mode.

Because there is no built-in sprite capability [for smoothly moving graphic shapes that have a color and location anywhere on the screen, and can be set in motion at a variety of speeds—Ed.] as on the TI-99/4A or Commodore 64, programmers must approach the PCjr differently to achieve "spritely" effects. Fortunately, the 16-bit microprocessor offers great speed. This—coupled with the PCjr's CRT Page Register, custom Video Gate Array and an architecture that allows virtually *all* of the system's 128K-bytes of RAM (in 16K byte "pages") to be available for display purposes [see Fig. 6]—permits very impressive color animations to be put on the screen.

IBM's custom microchip, the Video Gate Array, takes the data from memory and formats it for display. It contains a "palette" (16 word × 4 bit) which assigns color to the data. For example, in the 4-color medium-resolution mode, the palette allows the programmer to select any 4 colors out of the 16 possible. It also allows the program to quickly change the colors of the screen image by modifying the palette rather than memory data. [To learn how you can use this technique to create exciting animation, see our next issue—Ed.]

And finally, this versatile Video Gate Array also controls the screen's border color. Its border-color register is independent of the palette, and allows the program to select 1 out of 16 possible colors.

There are 3 types of output on the PCjr to get the video images to a screen: (1) RGBI direct drive video for IBM's Color Monitor or other compatible RGB monitor, (2) composite video for black/white or color composite monitors, and (3) hook-up to a standard TV via IBM's RF modulator [see Fig. 3].

Big Sound from Small Peanuts

The sound system in the PCjr is built around an "analog multiplexer" which allows one of four different sound sources to be selected, amplified, and sent to the audio outputs. Hiding under the power supply card on the left side of the system board [see Fig. 4A] is the first of the sources, the Complex Sound Generator. This is the small Texas Instruments chip (also found in the TI-99/4A) with the really "big" sound—three programmable frequencies which can be mixed to form chords, plus a white noise generator for special effects. Each of the channel's volume levels may be individually controlled, and each may be given separate durations. There is, however, no Attack, Decay, Sustain, and Release (ADSR)

Fig. 11 — Commands included in the two BASIC dialects (inner rectangle for Cassette BASIC and the *sum* of the three rectangles for Cartridge BASIC) available for the PCjr. The commands found within the two inner "shells" are "sub-sets" of Cartridge BASIC (it being the "super-set"). BASICA is the Advanced BASIC on disk for the IBM PC.

Commands Added to BASICA with IBM Cartridge BASIC

NOISE, PALETTE, PALETTE USING, PCOPY, PLAY OFF, PLAY ON, TERM

Commands Added to Cassette BASIC Through BASICA on the IBM PC

CHDIR	ON KEY(n)
CIRCLE	ON PEN
COM	ON PLAY(n)
COMMON	ON STRIG
CVI	ON TIMER
CVS	OPEN "COM
CVD	PAINT
DATE$	PLAY
DRAW	PLAY(n)
FIELD	PMAP
FILES	PUT files
GET files	PUT graphics
GET graphics	RESET
KEY(n)	RMDIR
KILL	STRIG(n)
LOC	TIME$
LOF	TIMER
LSET	VARPTR$
RSET	VIEW
MKDIR	WINDOW
MKI$	
MKS$	
MKD$	
NAME	
ON COM(n)	

IBM Cassette BASIC Commands

ABS	DELETE	LEFT$	PEEK	SIN
ASC	DIM	LEN	PEN	SOUND
ATN	EDIT	LET	POINT	SPACE
AUTO	END	LINE	POKE	SPC
BEEP	EOF	LINE INPUT	POS	SQR
BLOAD	ERASE	LINE INPUT#	PRINT	STICK
BSAVE	ERR	LIST	PRINT USING	STOP
CALL	ERL	LLIST	PRINT#	STR$
CDBL	ERROR	LOAD	PRINT#USING	STRIG
CHAIN	EXP	LOCATE	PSET	STRING$
CHR$	FIX	LOG	PRESET	SWAP
CINT	FOR-NEXT	LPOS	RANDOMIZE	SYSTEM
CLEAR	FRE(0)	LPRINT	READ	TAB
CLOSE	GOSUB	LPRINT USING	REM	TAN
CLS	GOTO	MERGE	RENUM	TRON
COLOR	HEX$	MID$	RESTORE	TROFF
CONT	IF-THEN	MOTOR	RESUME	USR
COS	INKEY$	NEW	RETURN	VAL
CSNG	INP	OCT$	RIGHT$	VARPTR
CSRLIN	INPUT	ON ERROR	RND	WAIT
DATA	INPUT#	ON GOSUB	RUN	WHILE
DEF FN	INPUT$	ON GOTO	SAVE	WEND
DEF SEG	INSTR	OPEN	SCREEN function	WIDTH
DEF type statement	INT	OPTION BASE	SCREEN statement	WRITE
DEF USR	KEY	OUT	SGN	WRITE#

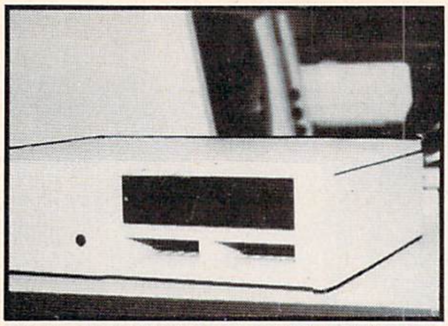


Fig. 12 — View of the two empty 64K ROM cartridge slots and slim-line disk drive. Cartridges may contain applications programs, data files, languages, and even completely new operating systems to "customize" the machine.

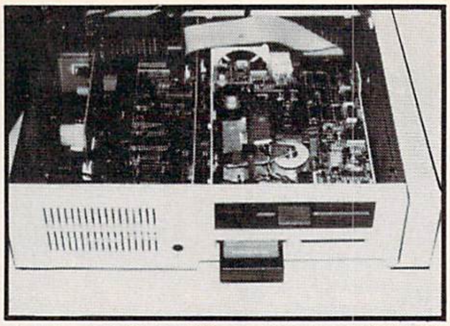


Fig. 13 — **Cartridge BASIC** in one of the two available 64K slots. With the language residing on a ROM cartridge rather than disk, a much larger amount in the memory of a 64K Entry Level System is available to the user.

AN IBM PC/PCjr COMPATIBILITY SAMPLER

A LOOK AT WHICH PC SOFTWARE RUNS ON THE PCjr

OPERATES ON PCjr

These disk-based programs distributed by IBM for the PC will run on a PCjr with 128K-bytes of memory and a single disk drive.

Program Name/Version	Cartridge BASIC Required
EDUCATION	
Adventures In Math/1.00	YES
Arithmetic Games (Set 1 & 2)/1.00 Note: No Color on composite monitor	YES
Bumble Games/1.00	YES
Bumble Plot/1.00	YES
Juggles' Butterfly/1.00	YES
LOGO/1.00	NO
Monster Math/1.00	YES
Turtle Power/1.00	NO
ENTERTAINMENT	
Adventure/1.00 Note: Can't set foreground/background colors	NO
Adventure in Serenia/1.00 Note: Colors may vary from IBM PC	NO
Casino Games	YES
Strategy Games	YES
PRODUCTIVITY	
Dow Jones Reporter/1.00	YES
Easy Writer/1.15 Note: Requires compatible 80-column display	NO
Home Budget, jr/1.00	YES
HomeWord/1.00	NO
Multiplan/1.00	NO
PeachText/1.00 Note: Requires compatible 80-column display; not recommended for PCjr	NO
Personal Communications Manager/1.00	NO
Personal Editor	NO
pfs: FILE/1.05 Note: Requires compatible 80-column display	NO
pfs: REPORT/1.05 Note: Requires compatible 80-column display	NO
Professional Editor/1.00 Note: Requires compatible 80-column display	NO
Time Manager/1.05	NO
VisiCalc/1.20	NO
Word Proof/1.00	NO
UTILITY & LANGUAGE	
Animation Creation/1.00 Note: Runs on 64K-byte PCjr	NO
Basic Compiler/1.00 Note: Requires sufficient storage for Compile & Link	NO
BASIC Programming Development System/1.05 Not recommended for PCjr	NO
Diskette Librarian/1.00	YES
File Command/1.00 Requires compatible 80-column display	NO
Macro Assembler/1.00	NO

DOES NOT OPERATE ON PCjr

EDUCATION	*UTILITY & LANGUAGE
Multiplication Tables/1.00 Private Tutor/1.00 Typing Tutor/1.00	Asynchronus Communications/2.00 BASIC Primer/1.00 Learning DOS 2.00/1.00 Learning to Program in BASIC/1.00
PRODUCTIVITY	**ENTERTAINMENT**
BPI Accounting Software Fact Track/1.00 Mail List Manager/1.00 Peachtree Accounting Software	Decathlon /1.00 101 Monochrome Mazes/1.00

* Note: Compiler output from COBOL, FORTRAN, and Pascal Compilers (1.00) will run on PCjr if sufficient storage; p-system products will not operate on PCjr.

envelope generator for "fine tuning" the quality of the sound as is found in the Commodore 64.

The second sound source is a "timer channel." It counts the internal clock "ticks" and can provide single tones to the small piezoelectric speaker found in both the PCjr and PC.

The third sound source is the audio from a cassette player. This allows the audio information from cassette tapes to be routed through to the audio outputs.

The last source is audio input over the I/O channel through a pin in the bus expansion connector (on the right side of the system box). This was put in to accommodate the addition of speech and other audio features.

Plug-In Peanut Power

The use of ROM cartridges as program media for home computers is nothing new. In fact, the practice goes back several years. Texas Instruments was the first to offer them; Commodore and most of the others soon followed suit. The PCjr's cartridge interface is, however, unique in three important aspects. First, it is the *program* in the cartridge that decides where in the memory map [see Fig. 6] the program will place itself. [This is accomplished by wiring the desired "chip select" to the ROM modules inside—Ed.]

Second, the PCjr provides *two* cartridge slots that are able to handle a total of up to 128K-bytes of ROM—enough memory to contain very sophisticated, complex application programs. With two cartridges inserted simultaneously, it becomes feasible to set up large data bases on the PCjr. One cartridge slot can accommodate the program, while the second accepts data from a series of several cartridges. A one-disk system configured this way would be very powerful, indeed—and in many cases would out-perform a conventional twin-disk system. It also should be said that putting programs in ROM cartridge form may be the *only* way some software developers will ever truly be satisfied with the performance of their PCjr implementations. This is because many programs executing out of the 128K-bytes of RAM pay a speed penalty due to the sharing of memory with the display.

The third unique aspect of this cartridge interface is the ability to disable the systems board's own 64K-bytes of ROM and replace it with ROM on a cartridge. This allows the cartridge-based program to gain control of the system at power-on time, and to replace the entire Basic Input/Output System (BIOS), Cassette BASIC, *Keyboard Adventure*, and everything else residing in the pair of 32K-byte system ROMs.

This last feature of the cartridge interface has significant implications. It makes it possible to turn the PCjr into an entirely different computer—from a general-purpose machine into a dedicated device. For example, the right programming (and peripherals) could have the machine taking telephone orders, controlling laboratory instrumentation, or keeping tabs on your home's security and energy efficiency. [This customization ability, used in conjunction with the machine's excellent "interrupt handling" facility, provides this home computer with the power and versatility to finally tackle some non-trivial applications in a home environment. In this regard, the major difference here between the PCjr and its predecessors is that the PCjr supports "event trapping." Its interrupt timer can be programmed (through Cartridge BASIC) to stop a running program at a specific time, prompt for something, branch to a specific routine, then continue. If external devices are connected, they can be ignored until this programmable interrupt occurs. Without event trapping, a program must continuously interrogate the system clock to get an accurate time period—Ed.]

Finally, it should be noted that whenever a cartridge is inserted or removed from a system slot, a momentary grounding of one pin causes the system to reset and go through a warm power-up. Any data in the system RAM will be lost.

Split-Level BASIC

The PCjr BASIC interpreter is structured in two functional levels: (1) Cassette BASIC, and (2) Cartridge BASIC. Cassette BASIC is built into the ROMs on the system board, and is operational when the power is first turned on. It not only handles the tape recorder operation, but also supports the keyboard, display, printer, joysticks and light pen, plus a full load of logic, math, string, and editing functions. [See Fig. 11.]

Cartridge BASIC is optional ($75), and plugs into one of the two cartridge slots on the front of the system unit [see Fig. 12 and Fig. 13]. It is a more powerful version of the disk-based Advanced BASIC (known as BASICA) for the PC. As shown in Figure 11, Cartridge BASIC is the super-set of both Cassette BASIC and BASICA. Most of its language extensions can be used without a diskette system but it also supports the diskette, and its structured directories, date, and time.

As you can see in Figure 7, Cartridge BASIC supports the PCjr's new enhanced screen modes 3 through 6. It also supports: the new statements PALETTE and PALETTE USING; an enhanced CLEAR statement that allows you to increase/decrease your video memory page size (with default at 16K-bytes); the enhanced sound capabilities [see Fig. 9] with SOUND, PLAY, and NOISE commands; and a terminal emulation mode (with the TERM subprogram) for simple RS232 asynchronous communications as well as both internal and external modem support.

In our view, Cartridge BASIC is a *MUST* for all PCjr users. Without it, the only way to get at the machine's rich graphics, sound, communications, and event-trapping capabilities is through assembly language programming—certainly possible, but definitely *not* for the general audience for which this machine is intended. Furthermore, most—if not all— of the commercial software written in BASIC for this machine will require the cartridge version to run.

Examine some of the program listings for the PCjr in this issue of *Home Computer Magazine*, and you will start to see the power and versatility of this new dialect of BASIC.

At Home With DOS

The Disk Operating System (DOS) of the PCjr is IBM's new DOS 2.1, an enhanced release of 2.0 for the PC and XT models. DOS 2.1 is the link between the systems hardware and the disk-based software that you choose to feed in. All DOS commands are supported in the 64K-byte Entry Model environment of PCjr, so the expansion memory is not

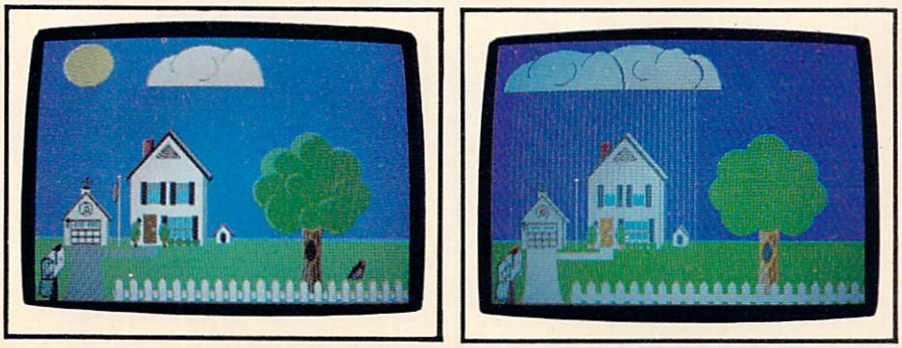


Fig. 15 — Two IBM sales demonstration screens show some of the graphic special effects possible on the PCjr. These were programmed on a 128K system in screen mode 5.

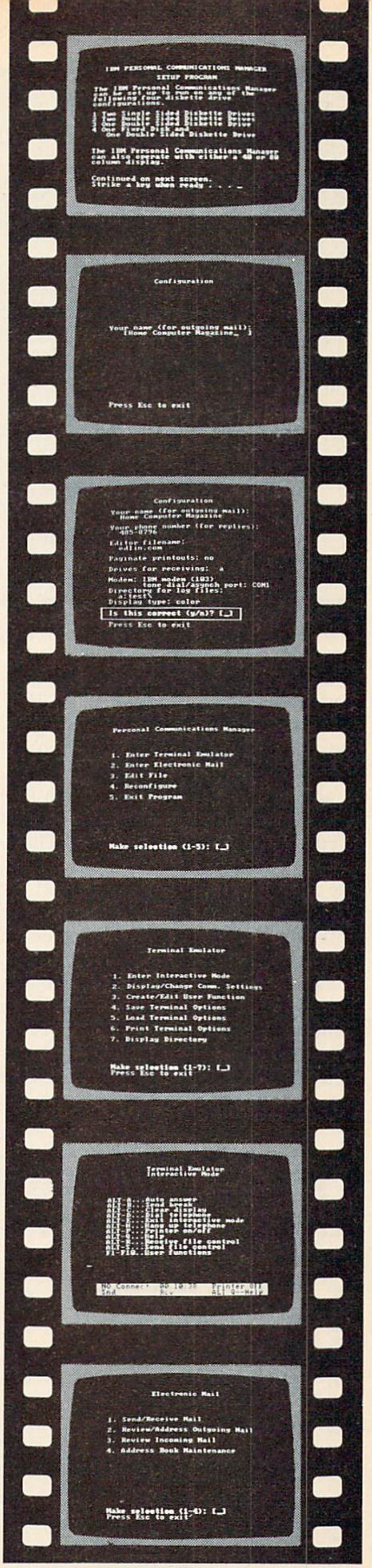


Fig. 16 — These screens from the IBM Personal Communications Manager illustrate three functions of the software: Terminal Emulation, Electronic Mail and Word Processor/Editor access. This package enables users to send and receive information via the telephone line. The menu-driven software provides many functions easily modified for use with different modems.

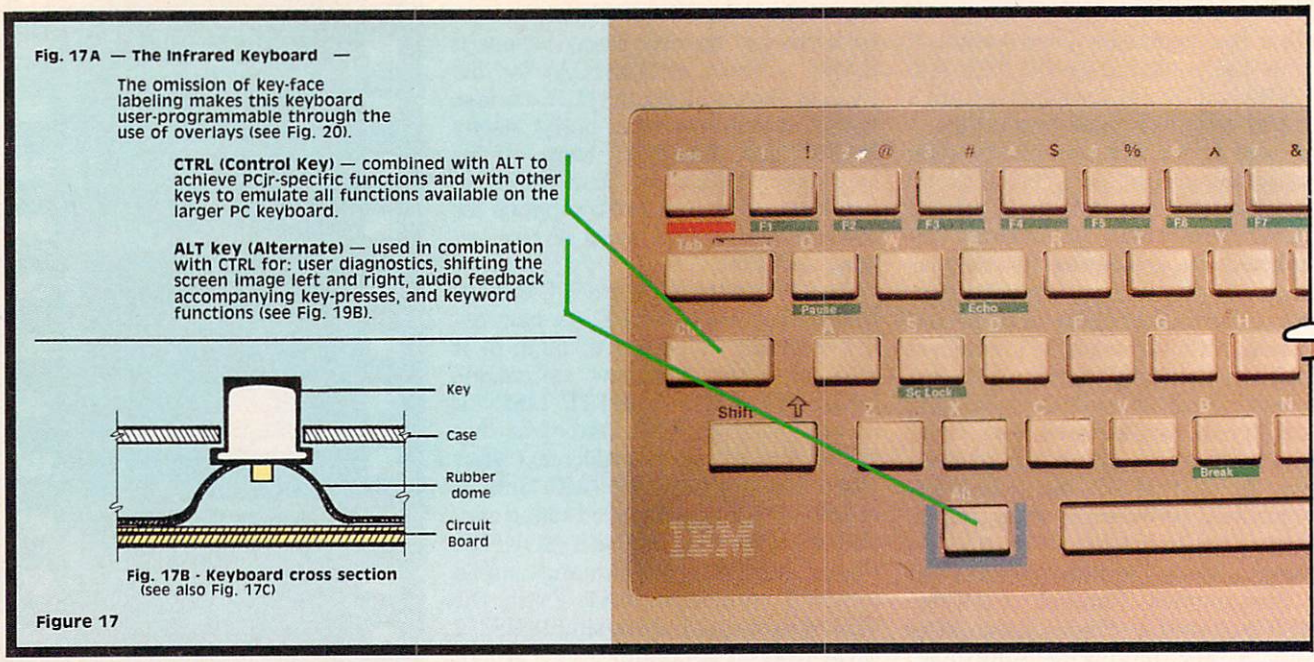


Fig. 17A — The Infrared Keyboard —

Fig. 17B - Keyboard cross section (see also Fig. 17C)

Figure 17

required to use a diskette with this system.

DOS 2.1 contains extensions for supporting the new Cartridge BASIC, IBM PC Compact Printer (See Fig. 3F), and the different diskette drive characteristics of the PCjr 360K-byte slim-line drive. It also comes with documentation that is easier to use than that of its predecessor—a separate User's Guide (for first-time users) that accompanies the standard Reference Manual.

A powerful DOS is a necessity for software developers—and a common one shared with the ubiquitous IBM PC is a boon to rapid software availability. But we can't help but wonder if a somewhat more "transparent" operating system (one that is more "hidden" and less intrusive until requested) wouldn't be better and less intimidating for a typical first-time home user. Nevertheless, all the instructions are there—they will just take a little getting used to for the neophytes. In subsequent issues of this magazine, we'll be showing you "all the ropes."

Peanuts Can Talk and Write Too

The RS232 serial port on the PCjr system board will support asynchronous (1-way at a time) communications at 4,800 bits per second (baud) for transmitting, 4,800 baud for receiving with no keyboard overlap, and 1,200 baud for receiving with overlapped keyboard operation. This is half the speed of serial ports on other popular home computers, but should be adequate for most uses. If faster printer operation is required, IBM's Parallel Printer Attachment ($99) should be purchased [see Fig. 3G]. The serial port works well with the IBM PC Compact Printer ($175). This thermal 80-column unit has a top speed of 1,200 baud, so it is not "held back" by the limit of the serial interface [see Fig. 3F].

Of course, serial ports aren't only for printers. Many home users who already own modems will want to hook them up to this interface for telephone communication capabilities [see Fig. 3H and Fig. 3I]. IBM does provide an optional Internal Modem card ($199). Using this auto-dial, auto-answer 300-baud internal modem rather than an external one provides the advantage of leaving the one serial port free for other uses. It also has been designed to perform its wonders with the separate Personal Communications Manager software package ($100). Business users who need the 1,200 baud over-the-telephone communications capability can buy the faster external modems or wait for a third-party hardware firm to offer a PCjr-compatible, 1,200-baud internal modem.

The Personal Communications Manager software [see Fig. 16] has been designed for first-time users. It provides a wide range of functions in terminal emulation and electronic mail, plus allows easy editing (with all IBM PC editors). The modems supported right

Fig. 18 — Character and graphics symbols available on the PCjr. There are 16 special characters for games support; 15 for word processing/editing; 96 for standard ASCII/graphics set; 48 for foreign languages; 48 for business block graphics; 16 Greek symbols; and 15 scientific notation characters.

ASCII value	Control character	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character	ASCII value	Character
000	NUL	(null)	032	(space)	064	@	096	`	128	Ç	160	á	192	└	224	α
001	SOH	☺	033	!	065	A	097	a	129	ü	161	í	193	┴	225	β
002	STX	●	034	"	066	B	098	b	130	é	162	ó	194	┬	226	Γ
003	ETX	♥	035	#	067	C	099	c	131	â	163	ú	195	├	227	π
004	EOT	♦	036	$	068	D	100	d	132	ä	164	ñ	196	─	228	Σ
005	ENQ	♣	037	%	069	E	101	e	133	à	165	Ñ	197	┼	229	σ
006	ACK	♠	038	&	070	F	102	f	134	å	166	ª	198	╞	230	µ
007	BEL	(beep)	039	'	071	G	103	g	135	ç	167	º	199	╟	231	τ
008	BS	◘	040	(	072	H	104	h	136	ê	168	¿	200	╚	232	Φ
009	HT	(tab)	041	)	073	I	105	i	137	ë	169	⌐	201	╔	233	Θ
010	LF	(line feed)	042	*	074	J	106	j	138	è	170	¬	202	╩	234	Ω
011	VT	(home)	043	+	075	K	107	k	139	ï	171	½	203	╦	235	δ
012	FF	(form feed)	044	,	076	L	108	l	140	î	172	¼	204	╠	236	∞
013	CR	(carriage return)	045	-	077	M	109	m	141	ì	173	¡	205	═	237	Ø
014	SO	♫	046	.	078	N	110	n	142	Ä	174	«	206	╬	238	ε
015	SI	☼	047	/	079	O	111	o	143	Å	175	»	207	╧	239	∩
016	DLE	►	048	0	080	P	112	p	144	É	176	░	208	╨	240	≡
017	DC1	◄	049	1	081	Q	113	q	145	æ	177	▒	209	╤	241	±
018	DC2	↕	050	2	082	R	114	r	146	Æ	178	▓	210	╥	242	≥
019	DC3	‼	051	3	083	S	115	s	147	ô	179	│	211	╙	243	≤
020	DC4	¶	052	4	084	T	116	t	148	ö	180	┤	212	╘	244	⌠
021	NAK	§	053	5	085	U	117	u	149	ò	181	╡	213	╒	245	⌡
022	SYN	▬	054	6	086	V	118	v	150	û	182	╢	214	╓	246	÷
023	ETB	↨	055	7	087	W	119	w	151	ù	183	╖	215	╫	247	≈
024	CAN	↑	056	8	088	X	120	x	152	ÿ	184	╕	216	╪	248	°
025	EM	↓	057	9	089	Y	121	y	153	Ö	185	╣	217	┘	249	•
026	SUB	→	058	:	090	Z	122	z	154	Ü	186	║	218	┌	250	·
027	ESC	←	059	;	091	[	123	{	155	¢	187	╗	219	█	251	√
028	FS	(cursor right)	060	<	092	\	124	¦	156	£	188	╝	220	▄	252	ⁿ
029	GS	(cursor left)	061	=	093	]	125	}	157	¥	189	╜	221	▌	253	²
030	RS	(cursor up)	062	>	094	^	126	~	158	Pt	190	╛	222	▐	254	■
031	US	(cursor down)	063	?	095	_	127	⌂	159	ƒ	191	┐	223	▀	255	(blank 'FF')

Fig. 19A

ALT + CTRL Functions

Available Only on the PCjr (Not on the IBM PC)

Press ALT + CTRL	PCjr Special Functions
+ cursor left	Adjust entire screen image to the left
+ cursor right	Adjust entire screen image to the right
+ Caps Lock	Keys "click" when pressed
+ Ins	User diagnostics

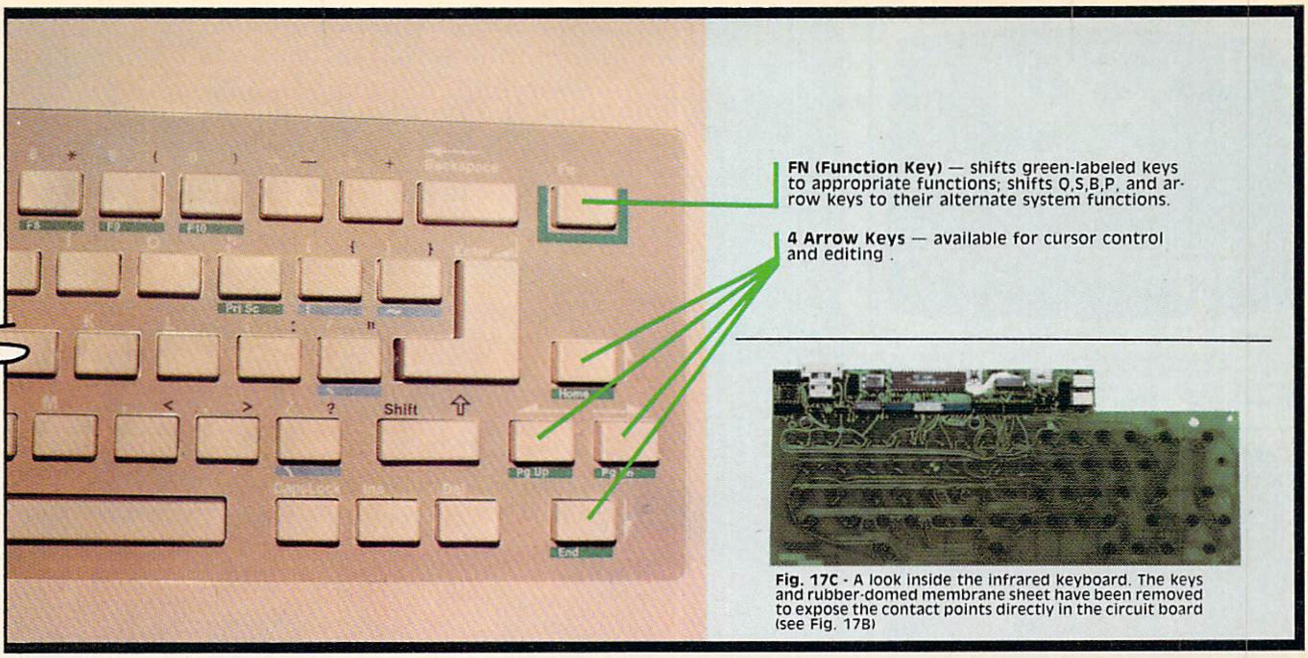


Fig. 17C - A look inside the infrared keyboard. The keys and rubber-domed membrane sheet have been removed to expose the contact points directly in the circuit board (see Fig. 17B)

off the shelf are the IBM PCjr Internal Modem, the Microcom PCS or RX modems, and the Hayes Smartmodems. Use with other modems requires writing an assembly language program for a "modem driver"—a task best left to the various modem manufacturers. Make sure that one is supplied with any modem you purchase.

A Cordless Peanut in the Home

Probably more has been said about the PCjr keyboard than any of its other features—or in this particular case, liabilities. Before examining the merits of all the criticism, the interface itself should be understood.

The keyboard interface on the system board is a small infrared receiver card mounted at the front of the board behind the circular opening on the front panel [see Fig. 4A]. There's also a small infrared transmitter mounted on this receiver card for diagnostic purposes.

The way it works is simple. When a key is pressed on the detached keyboard, a microprocessor in the keyboard sends the serial data stream to the infrared-emitting diodes. These flash on and off, sending the pulses across the room in the same way as a TV remote control. The receiver card in the PCjr then amplifies and processes the signal ("demodulates" it into an electrical signal) before sending it on to the system board. The 8088 microprocessor then converts the data to obtain the correct keyboard scan codes.

The keyboard itself is a low profile, 62-key detached unit with full-travel keys arranged in standard QWERTY typewriter layout. It has an additional function key and cursor control keys [see Fig. 17]. There are no markings on the keytops—rather, the usual key designations are found printed on the flat panel above the keys. This was done to allow the use of overlays with different markings [see Fig. 20], making the keyboard easy to use for dedicated tasks, without having to commit a lot of special key-press functions to memory.

The PCjr's cordless keyboard—or "free board," as some Madison Avenue types will undoubtedly be calling it—is battery-powered by four AA 1.5 volt cells. There is no on-off switch required because of the low power integrated circuitry (CMOS), and because the unit is normally in the standby, power-down mode until a key is pressed. When an optional cord ($20) is attached, the cord's keyboard connector removes the battery power, and a cable-connect signal disables the infrared receiver circuit on the PCjr system board. This allows other infrared devices to be used without interfering with the computer—a necessity in a school room or office environment with multiple PCjrs.

Unlike other keyboards used by IBM, this one has "phantom-key" detection—occurring when three or more keys are pressed simultaneously. The phantom-key code that is generated instructs the keyboard's microprocessor to ignore all of the keys that were pressed at that time. This feature will undoubtedly be responsible for many complaints of the type: "The keyboard didn't pick up some of the characters I typed."

Figures 17B and 17C will show you how the keyboard is constructed. It's

Fig. 19B **Keyword Functions**

Press ALT + keys listed to call up the appropriate keywords.

Key	Keywords	Key	Keywords
A	AUTO	O	OPEN
B	BSAVE	P	PRINT
C	COLOR	Q	(not used)
D	DELETE	R	RUN
E	ELSE	S	SCREEN
F	FOR	T	THEN
G	GOTO	U	USING
H	HEX$	V	VAL
I	INPUT	W	WIDTH
J	(not used)	X	XOR
K	KEY	Y	(not used)
L	LOCATE	Z	(not used)
M	MOTOR		
N	NEXT		

Fig. 20 - Overlays, like this one from Sierra On-Line's HomeWord word processing software allow the keyboard to be programmed to facilitate special applications.

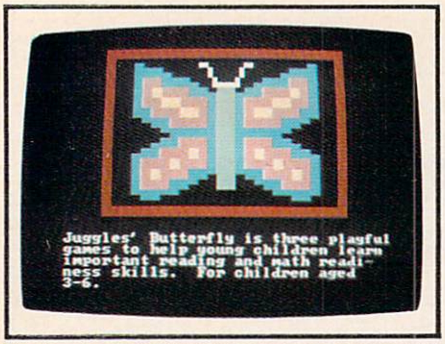


Fig. 21 — **Juggles' Butterfly:** three game programs to introduce young children (3-6 yrs.) to computers before they can read. Games feature colorful clowns etc. to teach children spatial concepts and math skills.

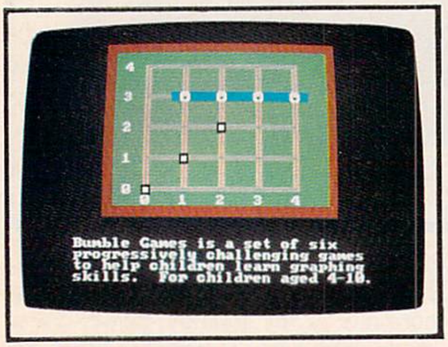


Fig. 22 - **Bumble Games:** is a set of six animated programs of varying difficulty (ages 4-10 yrs.). Teaches concepts such as "greater than", "less than", and drawing computer pictures and graphs.

Fig. 23 - The initial batch of software for the PCjr includes: *Monster Math, Juggles' Butterfly, Bumble Games, Mine Shaft, Crossfire, Scuba Adventure, Mouser,* and *Animation Creation.* These offerings in their "soap-dish" style packaging (accomodating either cartridge or disk software) are accompanied by colorful booklets. Most initial releases have been disk-based.

really an elegantly simple, peanut-butter-proof design. Protruding through the top cover is a set of 62 "Chiclet-style" keys that sit on a bumpy rubber membrane—each key atop a dome-shaped bump. The membrane provides an effective shield against spilled orange juice and cookie crumbs. The rubber domes provide the pressure and spring-back when keys are pressed. A tiny carbon disk makes connection with the circuit board on each key-press, and sends its corresponding signal code to the keyboard's microprocessor.

The keyboard design offers a new element into the home computer equation—that of being able to use the computer in some capacities while sitting across the room (up to 20 feet away). This feature is obviously not for word processing—you'd need a huge monitor to be able to work from 20 feet away! But it is useful for many educational and entertainment applications. Think about using a large TV in a classroom as a "blackboard" and passing the keyboard around instead of asking students to "come up and write on the board." This remote control ability also lends itself to commanding the PCjr to control household functions: with each task appropriately marked on your custom overlay, you would just point and press. . . The infrared keyboard decoder design also allows for IBM or others to later offer more exotic input devices—wireless joysticks, mice, touch pads, and the like.

For extended bouts of word processing, touch-typists will want a more standard keyboard with a better feel—larger key-tops with key designations *on* the keys (in a lower/upper format instead of the PCjr's slightly disorienting left/right, un-shifted/shifted design), and tactile feedback. Fortunately, third-party keyboard manufacturers will be offering excellent accessory keyboards in the $200 price range, so all is not lost for more professional use of this otherwise versatile machine. It's also possible that IBM might "unbundle" this keyboard from the system price and/or offer its own 83-key PC keyboard separately.

Figures 17, 19A, and 19B reveal some similarities and differences between the keyboard functions of the PCjr and the larger PC. With the right combination/sequences of the PCjr's CTRL, ALT, and FN keys, all the functions of the PC's larger 83-key keyboard can be produced—even the BASIC keyboard functions (so useful in programming) are produced the same easy way. The PCjr also offers four new functions not found on the PC: the ability to adjust the screen image either right or left (useful for centering on different monitors), access to user diagnostics, and the ability to turn on an audible "click" (somewhat muffled) when keys are pressed.

The Shell, Only Just Cracked. . .

According to Philip D. Estridge, president of IBM's Entry Systems Division, "The PCjr's uses are limited only by the imagination of the person using it." In working with the machine, you get the feeling that its wide-open architecture and dormant versatility are begging you to apply your creative potential. It won't be long before users will be able to purchase off-the-shelf hardware and software enhancements for every conceivable application—much as Apple users can choose from the large outpouring of Apple II add-on products.

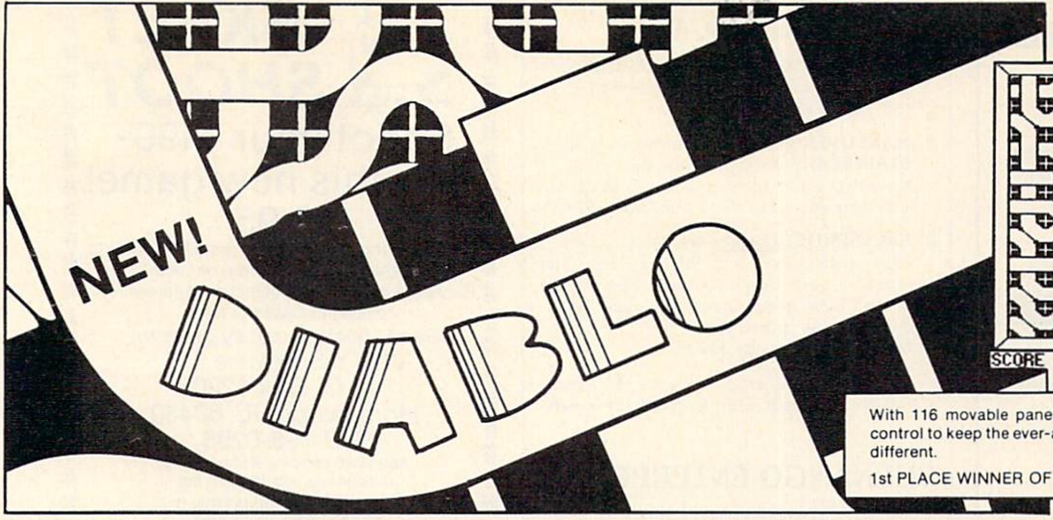
It's obvious that IBM is focusing primarily on providing "user-friendly" software for home management, family entertainment, and child education. They've made the PCjr suitable for preschoolers as well as adults. In fact, the PCjr will probably fuel an explosion in sophisticated disk-based educational games. We expect to see a prodigious offering of adult self-improvement, strategy, and simulation programs from the third-party software industry—categories heretofore virtually ignored because of memory limitations on other popular home computers.

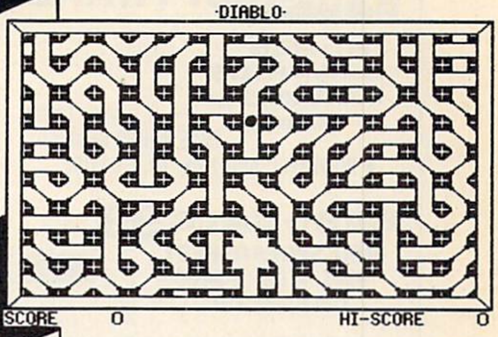
The business world too, might be the scene of a Peanut invasion. We expect to see the PCjr take on the role of a low-end communications terminal. It would complement its big brother, PC, in the office, and tie into the big IBM mainframes of corporate America.

Is there a PCjr in your future? If IBM has its way, finding one at a local dealer will soon be as easy as Peanut pie. . . HCM

SOFTWARE FOR THE 99/4(A) FROM ESC

XTENDED OFTWARE OMPANY

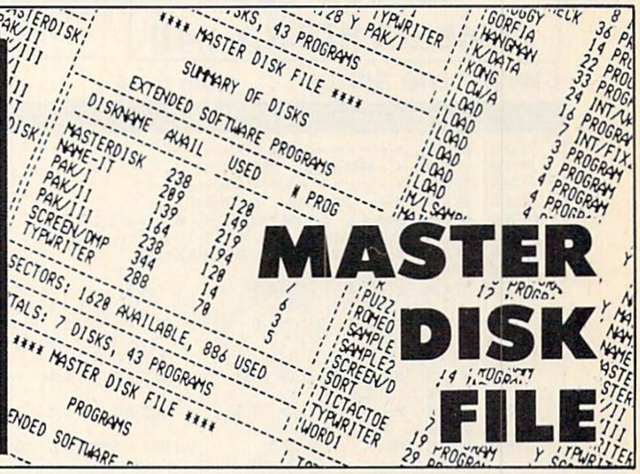




With 116 movable panels, this game demands strategy, planning & decisive control to keep the ever-advancing ball from rolling off the board. Every board is different.

1st PLACE WINNER OF THE T.I.S.H.U.G. AUSTRALIA CONTEST

MASTER DISK FILE — *A master index of your disks.*
MASTER DISK FILE uses a large portion of the 90K bytes available on a single sided disk as virtual memory to create and store a perpetual file of disks, programs and applicable data. Data can be added, removed or updated. The file may then be accessed for viewing on the screen or for printing several types of lists on a printer.
Insert disk into drive, it is catalogued then can be filed.
Reads up to 95 programs and files per disk.
Maximum of 120 disks or 1100 programs may be filed.
Does not require memory expansion.
Supports single or double sided disks (or mixed).
Supports single or multiple drives.
Programs are catalogued from disk in order and merged into the file; no sort is required after filing.
SEARCH for disks or programs by name.
Look-up time from a cold start: under one minute; from a running program: 15 to 25 seconds!
List on screen or a printer in alphabetical order by program name or disk name.
Up-date disks that have been changed by inserting into drive. Automatically replaces and up-dates old information.
May be used without a printer (on screen); file is portable and can be taken to a friend's for printing.
Supports any printer: serial or parallel.
Diskette (only) $15.00
GUARANTEED TO BE THE BEST DISK CATALOG UTILITY FOR THE 99/4A



GAMES PAK/III

KONG - must fight his way past barrels, ladders & trap doors to save Roxanne from Igor's bomb. Six different screens. Joystick req. $15.00

BOUNCER — bounds from one trampoline to another, scoring points while avoiding the arrows which will burst him. 6 screens. Joysticks req. $15.00

ROMEO — must traverse a desert, stream (alligators & sharks) & cavern to obtain a fitting reward. Joysticks req. $15.00

GAMES PAK/II

ARTILLERY — The opposing force must be destroyed by determining angle and force of each shot. An ever-changing wind complicates matters. 1 or 2 players $9.95

DE-CYPHER — An encrypted message is displayed and guesses change all corresponding letters to the guess. Comes with 50 messages which can be changed $9.95

PUZZLE 15 — Slide alphabetic squares into order. Multiple squares may be moved. Computer keeps track of moves & has replay feature $9.95

FLIP CHECKERS — Sandwich opponent's piece to flip it to your color. 1 or 2 players. Real-time clock. Fast computer decisons. Joysticks req. $9.95

GAMES PAK/I

FROGGY — Jump FROGGY across 10 lanes of traffic, then across 6 logs. Keyboard or joysticks $9.95

EXTENDED BASEBALL - Joystick control of the pitcher & batter, and individual batting averages. Multi-base & multi-runner plays. Joysticks req. $9.95

GORFIA PESTULITIS - Joystick control of a laser sight or inertia influenced space mines to shoot down the invading Gorfians. Joysticks req. $9.95

EXTENDED HANGMAN - Quick graphics, music, color, speech (optional) and sound are added to keep the players entertained. Includes 580 words of 4 to 9 letters in length in easy, medium and difficult groups $9.95

TIC-TAC-TOE — Quick set-up and quick decision making at four levels of difficulty. The levels avoid the frustration of the novice never having a chance to win $9.95

TYPWRITER
A 16K WORD PROCESSOR on Cassette or Disk
Any Input/Output storage of text — disk, cassette, cassette input/disk output, or vice versa.
Selectable *right justify.*
Complete text *Editing* — by cursor control; including insert & delete lines, partial print, printer halt or abort without text loss, page FWD & BKWD, and more.
Complete *Software Control* of Printer (depending upon its capabilities) — for enhanced print, underlining, formatting, 28 to 254 characters per print line, etc.
No Special Equipment — monitor, console, Extended Basic module, C or D, printer.
Comes with a 20 page instruction booklet.
Cassette $32.00 Diskette $35.00

SCREEN/DUMP — Print the screen to a dot-addressable, 8-bit printer. Does not require extra memory! Disk version is simple to use. Cassette version requires mild programming knowledge $12.00

NAME-IT
DATA BASE for: *Mail Lists, Labels, Files*
Records: 250 records per diskette consisting of up to nine 28-character items per record.
Prompts: user designated prompts.
Complete *File Sort:* 250 records in *100 Seconds.*
Search; Pre-set; print labels & lists.
Includes a FORM LETTER program that uses NAME-IT data in TYPWRITER generated form letters
Cassette version differs from disk version.
Cassette $32.00 Diskette $35.00

NOTE: Should you decide to up-grade to the TI-WRITER module, TYPWRITER and NAME-IT data can be converted for use by that module. NAME-IT alone, will generate 250 TI-WRITER form letter records.
TI-WRITER is copyrighted software of Texas Instr.

A detailed catalog is available free. Circle "FREE" on the order form or send a letter or postcard.

ORDER FORM

EXTENDED BASIC MODULE REQUIRED FOR ALL PROGRAMS

CATALOG	FREE		$ FREE
GAMES PAK/I (Froggy, Extended Baseball, Gorfia Pestulitis, Extended Hangman, Tic-Tac-Toe)	$26.95	(C or D)	$
GAMES PAK/II (Artillery, De-Cypher, Puzzle 15, Flip Checkers)	$26.95	(C or D)	$
GAMES PAK/III (Kong, Bouncer, Romeo)	$26.95	(C or D)	$
TYPWRITER (word processor)	$32.00	(C price)	$
	$35.00	(D price)	$
NAME-IT (data base/mail list)	$32.00	(C price)	$
	$35.00	(D price)	$
SCREEN/DUMP (printer required)	$12.00	(C or D)	$
MASTER DISK FILE	$15.00	(D only)	$
INDIVIDUAL GAMES: DIABLO	$19.95	(C or D)	$

☐ Froggy ☐ Extended Baseball ☐ Gorfia Pestulitis ☐ Extended Hangman ☐ Bouncer ☐ Kong
☐ Tic-Tac-Toe ☐ Artillery ☐ De-Cypher ☐ Puzzle 15 ☐ Romeo ☐ Flip Checkers

Total Individual Games: (C or D) $

Shipping & Handling (1st Class Mail): Included
Sales Tax: Included
Add $2.00 if C.O.D. (U.S. Mail Only): $
Check or money order or C.O.D. Total: $
PLEASE SPECIFY CASSETTE OR DISK

Send this form or a substitute with check or money order to:
Extended Software Company
11987 Cedarcreek Drive
Cincinnati, Ohio 45240
(513) 825-6645

SATISFACTION GUARANTEED
IF YOU ARE NOT COMPLETELY SATISFIELD, YOU MAY RETURN THE PROGRAMS (and instructions) WITHIN 15 DAYS FOR A FULL REFUND OF YOUR PURCHASE PRICE.

Dealer and programmer inquiries welcome.

Cut Here (may be copied or substituted)

I Write the Songs . . . from p. 36

TI-99/4A

```
100 REM ***************
110 REM * MUSIC MAGIC *
120 REM ***************
130 REM BY CAROL BURRIS
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.1.1
160 REM TI EXTENDED BASIC
170 REM
180 DIM S(24),K(43,3),U(43),SLN(14),LS$(14),X(3),NT(14),FS(5),START(3)
190 DATA 8,8,7,7,6,6,5,5,4,4,3,3,2,2
200 DATA "L","S","L","S","L","S","L","S","L","S","L","S","L","S"
210 FOR RLN=1 TO 14 :: READ SLN(RLN):: NEXT RLN
220 FOR RLN=1 TO 14 :: READ LS$(RLN):: NEXT RLN
230 FOR N=1 TO 24 :: S(N)=208*1.059463094^N :: NEXT N
240 DATA 1,3,4,6,8,9,11,13,15,16,18,20,21,23
250 FOR I=1 TO 14 :: READ NT(I):: NEXT I
260 CALL CHAR(96,"000000000FF")
270 HL$="0000003E41FF413E" :: CALL CHAR(108,HL$):: HOL$="0000007C82FF827C" :: CALL CHAR(99,HOL$):: CALL CHAR(95,HOL$)
280 CALL CHAR(94,"0101010101FF0101")
290 CALL CHAR(127,"0000007E8181817E")
300 HS$="3E4141413EFF" :: CALL CHAR(118,HS$):: HOS$="7C8282827CFF" :: CALL CHAR(127,HOS$):: CALL CHAR(123,HOS$)
310 CALL CHAR(104,"0101017D83FF837D")
320 CALL CHAR(126,"0101017D8381837D")
330 CALL CHAR(114,"7D8383837DFF0101")
340 QNL$="0101017DFFFFFF7D" :: CALL CHAR(102,QNL$):: CALL CHAR(101,QNL$)::QOL$="0000007CFEFFFE7C" :: CALL CHAR(92,QOL$)
350 CALL CHAR(93,QOL$)
360 QNS$="7DFFFFFF7DFF0101" :: CALL CHAR(112,QNS$):: CALL CHAR(111,QNS$)::QOS$="7CFEFEFE7CFF" :: CALL CHAR(120,QOS$):: CALL CHAR(121,QOS$)
370 CALL CHAR(103,"9060000000FF")
380 CALL CHAR(128,"7E24247E24FF"):: CALL CHAR(129,"0000000000FF0024"):: CALL CHAR(130,"0000247E24FF247E"):: CALL CHAR(131,"2400000000FF")
390 CALL CHAR(132,"3C2424243CFF"):: CALL CHAR(133,"0000000000FF2020"):: CALL CHAR(134,"0020203C24FF243C"):: CALL CHAR(135,"0000000000FF")
400 A$=RPT$("F",16):: CALL CHAR(137,A$)
410 GOSUB 1350 :: DISPLAY AT(12,7)SIZE(15):"1.RECORDED";CHR$(137);"SONG"
420 DISPLAY AT(14,7)SIZE(10):"2.NEW";CHR$(137);"SONG"
430 CALL KEY(0,Z,W):: IF W=0 OR Z<49 OR Z>50 THEN 430 ELSE IF Z=50 THEN GOTO 470
440 GOSUB 1350 :: DISPLAY AT(12,7)SIZE(10):"1.CASSETTE" :: DISPLAY AT(14,7)SIZE(10):"2.DISKETTE" :: KEY$=""
450 CALL KEY(0,Z,W):: IF Z=82 THEN 410 ELSE IF Z<49 OR Z>50 THEN 450
460 GOSUB 1230 :: IF Z=82 THEN 440 ELSE 790
```

TI-99/4A

```
470 GOSUB 1350 :: DISPLAY AT(12,5)SIZE(14):"1.SINGLE";CHR$(137);"NOTE";CHR$(137);"ENTRY"
480 DISPLAY AT(14,5)SIZE(16):"2.2";CHR$(137);"PART";CHR$(137);"HARMONY"
490 DISPLAY AT(16,5)SIZE(16):"3.3";CHR$(137);"PART";CHR$(137);"HARMONY"
500 CALL KEY(0,Z,W):: C=(-1)*(Z=49)-2*(Z=50)-3*(Z=51)
510 IF Z=82 THEN 410 ELSE IF W=0 THEN 500
520 CL=2 :: CTN=1 :: P=1 :: START(1)=1 :: W=C :: GOSUB 1420 :: IF S$="R" THEN 470
530 GOSUB 1070 :: IF D$<>"" THEN GOSUB 1530
540 FOR R=P TO 45
550 FOR M=1 TO C :: DISPLAY AT(21,11):"NOTE";M :: DISPLAY AT(23,1):R
560 DISPLAY AT(23,22):"END=Q"
570 ACCEPT AT(22,14)VALIDATE("ABCDEFG'QR"):K$ :: DISPLAY AT(23,22):""
580 IF K$="R" AND R=1 THEN 520
590 IF K$="R" THEN R=R-1 :: CTN=CTN+(R+1=START(CTN)):: CL=2 :: GOSUB 1070 :: IF D$<>"" AND CTN=1 THEN GOSUB 1530
600 IF K$="R" THEN GOSUB 1860 :: P=P+1 :: GOTO 540
610 IF (LEN(K$)<>1 AND SEG$(K$,2,1)<>"'")OR SEG$(K$,1,1)="'" THEN 570
620 IF LEN(K$)=1 AND K$="A" AND F=1 THEN 570
630 IF LEN(K$)=1 THEN K(R,M)=ASC(K$)-64 ELSE K(R,M)=(ASC(SEG$(K$,1,1))-64)+(ASC(SEG$(K$,2,1))-32)
640 IF K(R,M)=17 THEN 790
650 NEXT M
660 STF=0 :: IF W=1 THEN K(R,2),K(R,3)=K(R,1)
670 IF W=2 THEN K(R,3)=K(R,2)
680 DISPLAY AT(21,11):"DURATION"
690 DISPLAY AT(22,15):" "
700 ACCEPT AT(22,14)VALIDATE("1248")SIZE(1):U$
710 IF U$="" THEN 700 ELSE U(R)=VAL(U$)
720 X(1)=K(R,1):: X(2)=K(R,2):: X(3)=K(R,3)
730 IF CL>26 AND CTN=3 THEN DISPLAY AT(10,4)SIZE(23):"YOU FILLED THREE STAFFS" :: GOTO 740 ELSE 750
740 FOR ZZ=1 TO 500 :: NEXT ZZ :: GOTO 790
750 IF CL>26 THEN GOSUB 1070 :: CL=2 :: CTN=CTN+1 :: START(CTN)=R
760 GOSUB 1580 :: GOSUB 1810
770 CALL SOUND(250*U(R),S(X(1)),5,S(X(2)),5,S(X(3)),5)
780 NEXT R
790 CL=0
800 CALL CLEAR
810 FOR I=1 TO 14 :: CALL COLOR(I,2,1):: NEXT I
820 FOR OLN=1 TO 24 STEP 8 :: FOR LN=1 TO 5
830 CALL HCHAR(LN+OLN,2,96,29)
840 NEXT LN :: NEXT OLN :: P=R-1 :: GOSUB 1530
850 FOR R=1 TO P
```

TI-99/4A

```
860 X(1)=K(R,1):: X(2)=K(R,2):: X(3)=K(
R,3):: GOSUB 1580
870 IF CL>26 AND STF=0 THEN STF=8 :: CL
=0
880 IF CL>26 AND STF=8 THEN STF=16 :: C
L=0
890 NEXT R :: P=R-1 :: FOR R=1 TO P ::
X(1)=K(R,1):: X(2)=K(R,2):: X(3)=K(
R,3):: GOSUB 1810
900 CALL SOUND(250*U(R),S(X(1)),5,S(X(2
)),5,S(X(3)),5)
910 NEXT R
920 DISPLAY AT(23,8):"PRESS ANY KEY" ::
 STF=0
930 CALL KEY(0,Z,W):: IF W=0 THEN 930
940 GOSUB 1350
950 DISPLAY AT(12,7)SIZE(6):"1.PLAY" ::
 DISPLAY AT(14,7)SIZE(10):"2.NEW";C
HR$(137);"SONG"
960 DISPLAY AT(16,7)SIZE(13):"3.RECORD"
;CHR$(136);"SONG" :: DISPLAY AT(18,
7)SIZE(6):"4.EXIT"
970 CALL KEY(0,Z,W):: IF W=0 THEN 970
980 IF Z<49 OR Z>53 THEN 970
990 IF Z=49 THEN 790
1000 IF Z=50 THEN CL=0 :: GOTO 410
1010 IF Z=52 THEN 1870
1020 GOSUB 1350
1030 DISPLAY AT(12,7)SIZE(10):"1.CASSETT
E" :: DISPLAY AT(14,7)SIZE(10):"2.D
ISKETTE"
1040 CALL KEY(0,Z,W):: IF W=0 THEN 1040
1050 IF Z<49 OR Z>50 THEN 1040
1060 GOSUB 1110 :: GOTO 940
1070 CALL CLEAR :: FOR I=1 TO 12 :: CALL
 COLOR(I,8,8):: NEXT I
1080 FOR LN=2 TO 6 :: CALL HCHAR(LN,2,96
,29):: NEXT LN
1090 FOR I=1 TO 13 :: CALL COLOR(I,2,8):
: NEXT I
1100 RETURN
1110 ON ERROR 1340
1120 K(0,0)=(R-1):: IF Z=49 THEN 1170
1130 SONG$="" :: GOSUB 1350 :: DISPLAY A
T(12,5)SIZE(10):"SONG";CHR$(137);"T
ITLE"
1140 CALL KEY(0,Z,W):: IF W=0 THEN 1140
1150 L$=CHR$(Z):: IF Z=13 THEN 1160 ELSE
 SONG$=SONG$&L$ :: DISPLAY AT(14,5)
SIZE(10):SONG$ :: DISK$="DSK1."&SON
G$ :: GOTO 1140
1160 OPEN #1:DISK$ :: GOTO 1180
1170 OPEN #1:"CS1",OUTPUT,FIXED 64
1180 PRINT #1:K(0,0):: PRINT #1:C :: PRI
NT #1:N :: PRINT #1:SFLG :: FOR J=1
 TO N :: PRINT #1:FS(J):: NEXT J
1190 FOR I=1 TO R
1200 FOR Y=1 TO 3 :: PRINT #1:K(I,Y):: N
EXT Y :: PRINT #1:U(I):: NEXT I
1210 CLOSE #1
1220 RETURN
1230 ON ERROR 1330 :: IF Z=49 THEN 1280
1240 SONG$="" :: GOSUB 1350 :: DISPLAY A
T(12,5)SIZE(10):"SONG";CHR$(137);"T
ITLE"
1250 CALL KEY(0,Z,W):: IF W=0 THEN 1250
1260 L$=CHR$(Z):: IF Z=13 THEN 1270 ELSE
 SONG$=SONG$&L$ :: DISPLAY AT(14,5)
SIZE(10):SONG$ :: DISK$="DSK1."&SON
G$ :: GOTO 1250
1270 OPEN #1:DISK$,INPUT :: GOTO 1290
```

TI-99/4A

```
1280 OPEN #1:"CS1",INPUT ,FIXED 64
1290 INPUT #1:K(0,0):: INPUT #1:C :: INP
UT #1:N :: INPUT #1:SFLG :: FOR J=1
 TO N :: INPUT #1:FS(J):: NEXT J
1300 QR=K(0,0):: FOR R=1 TO QR :: FOR Y=
1 TO 3 :: INPUT #1:K(R,Y):: NEXT Y
:: INPUT #1:U(R):: NEXT R
1310 CLOSE #1
1320 RETURN
1330 Z=82 :: RETURN 1320
1340 RETURN 1210
1350 CALL CLEAR :: FOR I=2 TO 14 :: CALL
 COLOR(I,8,1):: NEXT I
1360 FOR I=2 TO 22 :: CALL HCHAR(I,6,137
,18):: NEXT I
1370 CALL HCHAR(23,7,128,17):: CALL VCHA
R(3,24,128,21)
1380 CALL COLOR(14,16,16,13,2,2):: FOR I
=2 TO 8 :: CALL COLOR(I,2,16):: NEX
T I
1390 DISPLAY AT(9,5)SIZE(17):"YOUR";CHR$
(137);"CHOICES";CHR$(137);"ARE:"
1400 FOR I=1 TO 2 :: CALL SOUND(250,330,
5):: NEXT I
1410 RETURN
1420 F=0 :: SFLG=0 :: N=0 :: GOSUB 1350
:: DISPLAY AT(12,5)SIZE(16):"ANY";C
HR$(137);"SHARPS?(Y/N)" :: ACCEPT A
T(13,9)VALIDATE("YNR")SIZE(1):S$
1430 IF S$="R" THEN RETURN
1440 IF S$="" THEN 1420
1450 IF S$="Y" THEN KEY$="SHARPS" :: GOT
O 1480
1460 GOSUB 1350 :: DISPLAY AT(12,5)SIZE(
15):"ANY";CHR$(137);"FLATS?(Y/N)" :
: ACCEPT AT(13,9)SIZE(1)VALIDATE("Y
N"):K$ :: IF K$="" THEN 1460
1470 IF K$="N" THEN RETURN ELSE KEY$="FL
ATS"&CHR$(137)
1480 DISPLAY AT(14,5)SIZE(15):"HOW";CHR$
(137);"MANY";CHR$(137);KEY$ :: ACCE
PT AT(15,9)VALIDATE(DIGIT)SIZE(1):N
 :: IF N>5 THEN 1480
1490 IF S$="Y" THEN SFLG=1 ELSE SFLG=-1
1500 GOSUB 1350 :: FOR G=1 TO N :: DISPL
AY AT(12,5)SIZE(8):KEY$;CHR$(137);"
=" :: ACCEPT AT(13,9)SIZE(1)VALIDAT
E("ABCDEFG"):SF$(G)
1510 IF SF$(G)="A" THEN F=1
1520 D$=D$&SF$(G):: DISPLAY AT(14,5)SIZE
(G):D$ :: FS(G)=ASC(SF$(G))-57 :: N
EXT G :: FOR A=1 TO 500 :: NEXT A :
: RETURN
1530 CL=2 :: IF KEY$="SHARPS" OR SFLG=1
THEN SIG=128 ELSE SIG=132
1540 FOR G=1 TO N :: IF LS$(FS(G))="L" T
HEN 1560
1550 CALL HCHAR(SLN(FS(G)),CL,SIG):: CAL
L HCHAR(SLN(FS(G))-1,CL,SIG+1):: GO
TO 1570
1560 CALL HCHAR(SLN(FS(G)),CL,SIG+2):: C
ALL HCHAR(SLN(FS(G))+1,CL,SIG+3)
1570 CL=CL+1 :: NEXT G :: RETURN
1580 CL=CL+2 :: CLF=0 :: Z=C :: F=C :: D
=C :: W=C :: SIM=0
1590 FOR J=1 TO F :: NS(J)="" :: NEXT J
1600 FOR J=1 TO Z-1 :: IF CLF=1 THEN 162
0 :: NS(J+1)="" :: IF K(R,J+1)-K(R,
J)=1 THEN NS(J+1)="O" :: CLF=1
1610 NEXT J
```

Continued on p. 56

I Write the Songs . . . from p. 55

TI-99/4A

```
1620 IF C=3 AND K(R,2)=K(R,3)THEN SIM=1
1630 FOR Y=1 TO W-SIM
1640 IF LS$(K(R,Y))="S" AND N$(Y)<>"O" T
     HEN CALL HCHAR(SLN(K(R,Y))+STF,CL,U
     (R)+110)
1650 IF LS$(K(R,Y))="L" AND N$(Y)<>"O" T
     HEN CALL HCHAR(SLN(K(R,Y))+STF,CL,U
     (R)+100)
1660 IF LS$(K(R,Y))="S" AND N$(Y)="O" TH
     EN CALL HCHAR(SLN(K(R,Y))+STF,CL+1,
     U(R)+119)
1670 IF LS$(K(R,Y))="L" AND N$(Y)="O" TH
     EN CALL HCHAR(SLN(K(R,Y))+STF,CL+1,
     U(R)+91)
1680 NEXT Y
1690 IF U(R)=8 THEN 1790
1700 FOR Q=1 TO D-1 :: BA(Q)=SLN(K(R,Q))
     -SLN(K(R,Q+1)):: IF BA(Q)<=1 THEN 1
     720 ELSE 1710
1710 FOR T=1 TO BA(Q)-1 :: CALL HCHAR(SL
     N(K(R,Q))-T+STF,CL,94):: NEXT T
1720 NEXT Q :: CALL HCHAR(SLN(K(R,D))-1+
     STF,CL,94):: IF C=1 THEN 1770
1730 IF C=3 AND LS$(K(R,1))="S" AND LS$(
     K(R,2))="L" AND BA(1)=1 AND SLN(K(R
     ,2))-SLN(K(R,3))<>0 THEN CALL HCHAR
     (SLN(K(R,2))+STF,CL,94)
1740 IF C=3 AND LS$(K(R,1))="S" AND LS$(
     K(R,2))="L" AND BA(1)=1 AND SIM=1 T
     HEN CALL HCHAR(SLN(K(R,2))+STF,CL,9
     4)
1750 IF C=3 AND LS$(K(R,2))="S" AND LS$(
     K(R,3))="L" AND SLN(K(R,2))-SLN(K(R
     ,3))=1 AND BA(1)<>0 THEN CALL HCHAR
     (SLN(K(R,2))-1+STF,CL,94)
1760 IF C=2 AND LS$(K(R,1))="S" AND LS$(
     K(R,2))="L" AND SLN(K(R,1))-SLN(K(R
     ,2))=1 THEN CALL HCHAR(SLN(K(R,D))+
     STF,CL,94)ELSE 1780
1770 CALL HCHAR(SLN(K(R,D))-1+STF,CL,94)
1780 IF U(R)=1 THEN CALL HCHAR(SLN(K(R,D
     ))-1+STF,CL+1,103)
1790 CL=CL+CLF
1800 RETURN
1810 FOR E=1 TO 3 :: FOR V=1 TO N :: IF
     (FS(V)=X(E))OR(FS(V)=X(E)+7)THEN X(
     E)=NT(K(R,E))+SFLG :: GOTO 1840
1820 NEXT V
1830 X(E)=NT(K(R,E))
1840 NEXT E
1850 RETURN
1860 P=R-1 :: FOR R=START(CTN)TO P :: GO
     SUB 1580 :: NEXT R :: RETURN
1870 END
```

HCM

Want to Get Published?

Home Computer Magazine is looking for articles in all areas of interest that concern Apple, Commodore, IBM, and Texas Instruments home computers. Here are the kinds of articles that we want you to write for us:

- Are you a businessman, professional, hobbyist, scientist, or engineer with an interesting microcomputer application? Tell us how it works, what problems you've had to overcome, and what recommendations you have for others. We're especially interested in sharing user-written software with our readers.
- Have you recently purchased a piece of hardware or software that hasn't quite come up to your expectations, or has, on the other hand, impressed you with its performance? We're looking for comprehensive product and book reviews from different perspectives.
- Are you an educator or parent with something to contribute to computer-assisted instruction (CAI)? We're always looking for new ideas and fresh approaches to educational problems.
- Have you created any unusual computer games or simulations? Let our readers experience your excitement and pleasure.
- Perhaps you've modified your microcomputer or have interfaced it with some unique or useful hardware. Send us your how-to-do-it story.

These are just some ideas. Perhaps you have others. Don't worry if you're not a professional writer. Our editorial staff stands ready to help polish up your manuscripts. And we'll be more than happy to send you a copy of our author guidelines.

Please send your double-spaced typed manuscripts, plus disks or cassettes (recorded on both sides) if the article includes program material, to:

Home Computer Magazine/Editorial Dept.
1500 Valley River Drive, Suite 250
Eugene, OR 97401

HEN PECKED

A Review by Greg Roberts
99'er HCM Staff

Name:	Henpecked
Program Type:	Barnyard Affair
Machine:	TI-99/4A
Distributor:	Navarone Industries 501 Vandell Way Campbell, CA. 95008
Price:	$37.50, cartridge

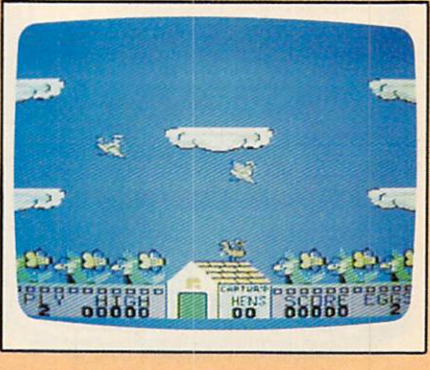
System requirements:
Assembly Language, Joysticks optional

Poor Fair Good Excellent
Performance:
Engrossment:
Documentation:

A computer chicken's life is not easy. Crossing the road, of course, is a day-to-day struggle for some chickens, while others are stuck in computer games with wolves and other predators. In *Henpecked* the problems are domestic: The birds are caught up in a courtship ritual that is not all love and kisses; in fact, it's closer to the domestic turmoil of a nest of black widows.

The prime mover in this drama is a red rooster whose only aim in life is to harass the hens that flutter and fly about the chicken house in the center foreground of your screen. To play this game, use a joystick or arrow keys to keep the rooster hovering, and to help him pull off a successful "landing," at which point the hen lays an egg and disappears. You may then chase the rolling egg and descend on it to rack up even more points—a move you can repeat time and again over the same egg. Even the flying hens can lay, and their eggs always fall gently to the ground where they hatch into more hens.

But these hens are definitely not the passive kind. If a she-bird lands on the



rooster, he's dead. And there is just one rooster against a whole flock of hens, a flock which gets bigger and more aggressive at the higher levels of play. There are ways, however, to protect your rooster: station him just under a cloud or at the top of the screen—either of which will act as a shield from the Amazon chickens attacking from above. Thus stationed, you can make short, not very daring attacks on the hens, and build up a score that would even make Casanova grunt his approval. This way, however, the game gets too easy, unless you establish a rule against cloud-hovering and hire a referee to enforce it—someone who understands rabbit punches, steroids, and sticky bats.

The game's scenario may not be an absolute delight to everyone—in fact, it is easy to see why half the population might be rubbed the wrong way by these birds' mating ritual. On the other side of the coop, game defenders might argue that the hens do have the opportunity to clobber the rooster—but still there is only one way to score.That's with the rooster in control. This game is less concerned with equality than it is with showing traditional barnyard scenes.

Uptown Barnyard Graphics

As for barnyard realism, this game's birds are expertly drawn and animated. They strut, flutter and fly with an intricacy of detail unsurpassed and seldom equalled in computer games. Yes, for all you nit-pickers plucking at details, chickens can fly to some extent; the

HOPPER

A Review by Greg Roberts
HCM Staff

Name:	**Hopper**
Program Type:	**Arcade game**
Authors:	**Michael D. Archuleta and John M. Phillips.**
Machine:	**TI-99/4A**
Distributor:	**Texas Instruments, Inc. P. O. Box 53 Lubbock, TX 79408**
Price:	**$39.95, cartridge**

	Poor	Fair	Good	Excellent
Performance:				
Engrossment:				
Documentation:				

Long ago, when the TI Home Computer was very young, a small band of users wandered over the outback looking for games. Tied to their bleak terrain, these aboriginal arcadians would pick up anything that looked loadable, digging with their joysticks the most primitive software.

Over the past few seasons, however, the desert has bloomed. Games are plentiful. Tonight the tribe sits well-satisfied round a billabong overflowing with fat, colorful tapes and diskettes, and each game hunter's CPU is warm and full.

In times of plenty the games people like to tell stories: "The rarest beast of all is the plug-in cartridge from the dry plains of the south," an elder tribesman explains to the young men who squat around a gray-backed monitor. "Such games come so seldom, each one brings the whole tribe out to see it—and once in a pink moon there comes a trophy that makes the whole clan celebrate."

But a cartridge of such a calibre has not shown its prongs for many seasons, and the games people have been craving to try their remote controllerangs on something new. Memories have grown dim of the last notable game from that remote place, and the tribe's chief—a user so positively ancient that he still remembers rubbing his fingers against the Chiclet keys of a 99/4—tries to recall the legendary coming of the last great cartridge. "Its name was Parsec," he explains, "and I seem to remember a great rejoicing in the desert, but it has been so long—and my mind is scrambled from staring endlessly at the computer-generated rocks and gullies." Suddenly the withered one cuts short his story, and all eyes stare out into the birds of this flock usually move quite realistically as they fight and flap through the nuptial dance. As for the chickens bombing through the clouds, eggs falling from the sky, and sudden death by hen-smothering, we have no choice but to switch gears and bear in mind that this is a computer game, not a natural history text. In any case, the program is a good example of the stunning graphics that can be achieved with a self-contained ROM cartridge which requires no additional memory packages, disk drives, or other expensive peripherals.

It is unfortunate that these superb graphics are without a game plan of equal worth. Regardless of expertly-drawn characters, a half-hour at the keyboard will reveal that there isn't enough going on here. Like professional actors in a weak play, the birds are all dressed up with no place to go. And as they float around on the screen, they sometimes fail to respond to the joystick, even when it's right on target.

Unwritten Rules

Many buyers of this cartridge may have to take some time to figure out the object of the game. The cartridge comes without documentation of any kind except for vague remarks on the carton such as "You as the rooster have a choice: to rule the roost or become totally henpecked." These loosely-stated rules of the game point to a program with loose ends here and there—idiosyncrasies that may annoy some players. Others will appreciate the game just for its graphics. HCM

---

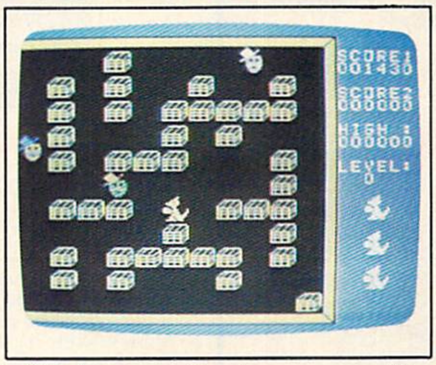
blackness to see a wonderful beast lope into the light of the campfire . . .

We who are stationed at that remote outpost called the *HCM* review desk also have full tucker bags of fine gameware these days. And as we load game after game, it is easy to feel a bit jaded, even fire-opaled, sometimes, from so much rich fare. When a game comes up on the screen, our first reaction is to pigeonhole it as to place of origin—whether it be The Land of Kong, Froggerania, or Pac-Man-Du. But this time we are unable to peg the program. Yes, believe it or not, there's something new under the joystick, a new TI Command Cartridge called *Hopper*.

Speeding through a maze of packing crates, three evil circus trainers are trying to grab Chadly, the pink kangaroo. You can make Chadly kick the crates to form barricades, or let him drop the cargo on his enemies. He can smash crates too, in order to make a trap for the trainers—permanently fencing them in. With this movable maze, the strategies are unlimited, giving this game near immunity to boredom.

This is a fast-paced game. You must think and move quickly to protect your little pal joey as you devise ways to trap the trainers. By some bizarre quirk of fortune, I managed to capture all three trainers together in one box where I could put the squeeze on them—Pit and the Pendulum style—and come up with a thrilling score of 5,870 on my first of the three kangaroos. As I bragged that this was the highest score possible for a first inning, someone more perceptive than I observed that, in fact, it would be easy to rack up a higher score based on an oddity of the game: You get ten points for kicking a crate, no matter what. So, once you've captured at least one trainer and squashed the others, you can sit there kicking crates until they are all lodged in the corners of the square. Then you can smash the crates for an additional sixty points each. These easy points can be taken as an extra reward for capturing the trainers, or as an annoyance. Fortunately, they are not significant enough to greatly alter a high score.

Hopper's graphics are good, but not extraordinary—except perhaps for the feis-



ty little kangaroo. You have to move him right, left, forward, or back in order to take action on the crate of your choice. These precise joystick maneuvers add to the challenge.

The game's sound effects are logical and simple. The high-pitched beeping might get annoying after a while, but the volume control is always handy.

The documentation (which at press time was in manuscript state, but adequate) mentions nine levels of play. The progressively-higher levels are simply faster and more difficult, with no change in scenery except to offer a different crate configuration. Only at Level 10 do you encounter a new and very difficult challenge. This is a "surprise" screen, so all we can tell you is to keep your eyes open for some flash of recognition, or it will be "lights out" for that round and back to Level 1.

Designers Phillips and Archuleta came up with *Hopper* as part of a TI employee-incentive program in which they designed the product at home, on their own time, with off-the-shelf equipment. No special TI-proprietary development systems were used on this one. That's right, lads, this particular big game was not bagged with the high-powered weapons of the Lubbock laboratories, but with the slings and arrows of that outrageous little device called the Editor/Assembler package. The program takes up only about 6K-bytes of memory.

This game is just the sort of program cartridge that makes the 99/4A such a good buy—even if you have nothing but a console and monitor. The best choice, mate—definitely the dinkum oil. HCM

FLAK ATTACK

by Mark Moseley

and the HCM staff

The best defense is a good offense. Or is it the other way around? In *Flak Attack,* either maxim could apply. Your job is to shoot down an attacking plane with your missile launcher before it blasts you three times. The plane attacks from left and right at random altitudes. The enemy's speed and frequency of fire vary with the skill level you choose. Your missile launcher can move back and forth and hide behind a barrier. Your launcher should be kept moving because the plane "remembers" its last position and is likely to fire at that spot.

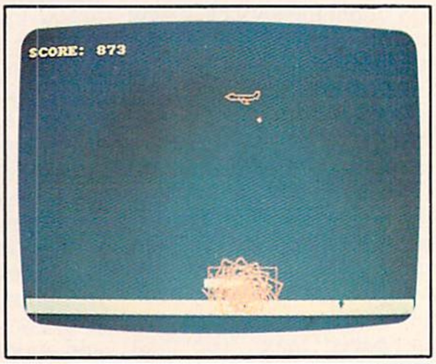
But don't expect the barrier to last forever—each time the pilot hits the barrier, he makes a hole that exposes you. Likewise, your missile launcher can fire through the holes blasted by the plane. Three keys—the S, D, and E keys—control your moves. The S key moves the missile launcher to the left, and the D key moves it right. The E key fires the missile. The fate of your government depends on your ballistic skills. Good luck, Captain.

Note: TI readers who have been with us from the beginning will recognize Flak Attack *as a new, improved version of* Anti-Aircraft Gun. *It originally appeared in Volume 1, Number 1 (May/June 1981), of* 99'er Magazine.

The moving plane is created with the DRAW command on the Apple IIe. DRAWing the plane at its first location in the same color as the background erases the plane; then reDRAWing it at a new location in the plane's color gives the plane the illusion of motion. The DRAW command also creates the exploding tank at the bottom of the screen. Rotating and expanding the shape with the ROT= and SCALE= commands creates the illusion of an explosion.

The IBM PCjr uses the DRAW command to create the plane and the exploding tank. The explosion effect is created by passing a series of new values to the DRAW string. These new values change the rotation angle and the scale size of the object.

Rather than drawing high-resolution lines, the Commodore C-64 and VIC-20 and the TI-99/4A create the effect of an explosion by making a series of alterations in the character definitions displayed at that location. The Commodore machines use POKEs; the TI uses CALL HCHAR.



The Apple version of *Flak Attack* uses shape tables and the DRAW command to produce animated graphics. The DRAW command lets you create shapes and then expand or rotate them. To use this command, you must first generate the code for the shape. These codes are listed in the Applesoft BASIC Programmers Reference Manual.

You will also need to create a shape table index so that the computer knows where you have stored the shape data in memory. The shape table index must include the number of shapes being defined. In addition, each entry needs an offset from the beginning of the index to the beginning of each shape table. The index created in line 1490 contains six shapes, with the first shape starting 15 bytes after the beginning of the index. The second shape starts 31 bytes after the start of the index, directly after the first shape. When the program says DRAW 1 AT 10,10, it will draw shape 1 (the cannon) at screen location 10,10.

Creating animation with the DRAW command is easy. Simply erase the old shape with a color that matches the background, and then redraw the shape at its new location. The exploding tank is quite a show. It was accomplished by expanding and rotating the tank shape at the same time, without erasing every time it's redrawn. The result is a swirling pool of expanding graphics animation.

Continued on p. 63

This game offers our readers a unique example of Auto Sprite Motion in practice. That powerful interrupt-driven machine-language program creates all the animation in this game, except for the firing of the bombs. For an in-depth explanation of this routine, see "Don't Be a SlowPOKE" in this issue. For now, we will talk about ways of using this valuable utility in a BASIC program.

The first consideration is loading the machine language program into Commodore memory. First type in the short BASIC program accompanying the article on Auto Sprite Motion. This will let you easily test your machine language program. Once you have it working, create a sequential data file on disk or tape with the machine language program as data. Then, whenever you want to use this utility, let your program read the sequential file and load it into its proper place in RAM. Procedures for creating and reading a sequential file can be found in the "Don't Be a SlowPOKE" article.

Once the loading is accomplished, the routine must be initialized and an interrupt vector loaded. Lines 1100 and 1110 perform this function.

Now the Auto Sprite routine is ready to be used by the BASIC program. *Flak Attack* may have up to three sprite images in motion at any one time—the airplane, the missile, and the tank.

The tank motion may look simple, but it uses an important aspect of the Auto Sprite Motion routine: It frees the BASIC programmer from having to monitor the X coordinate of sprites.

In moving the tank to the right, line 4250 checks to see if the D (move-right) key has been pressed, and also checks the position of the tank relative to the screen border. Border checking is necessary because a sprite contact with the background will set the sprite/background coincidence bit for the tank sprite—and that tells the program that the tank has been hit by a bomb.

Line 4200 begins missile sprite motion and reinitializes the Auto Sprite Motion subroutine. Location 50432 is the control byte for all motion of sprites. Each bit corresponds to a sprite. For example, if bit 0 is set equal to 1, then motion for sprite 0 is turned on. If bit 0 is reset to 0, then motion for sprite 0 is disabled. Bit 1 corresponds to sprite 1, and so on.

Look closely at this program to get a clear understanding of Auto Sprite Motion. Once you see how it works, I'm sure you will find many applications for Auto Sprite Motion in your own programs. Good luck, and may your sprite motion always be smooth.

Continued on p. 78

The VIC-20 version of this program is a little different from some of the others, although the rules are virtually identical. Points are awarded depending on the distance of the plane from the surface: 5 points for the closest planes, and 20 points for those further away.

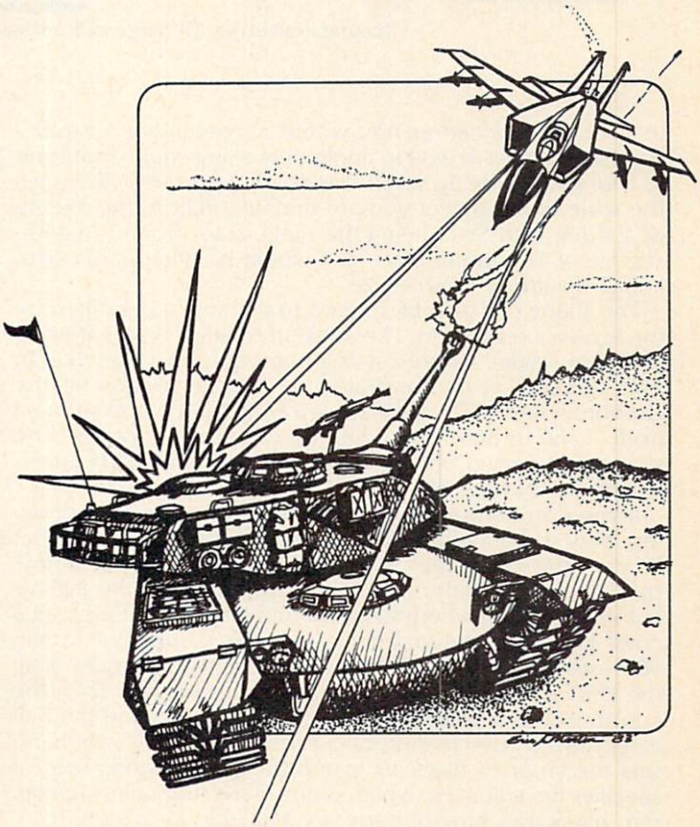
The program redefines the standard character set to create shapes of the tank, rocket, plane, etc. In order to redefine the characters for the patterns, the character set has to be moved out of ROM and into RAM. There, the actual pixel-by-pixel definitions of the characters can be altered. The "top of BASIC memory" pointers at locations 52 and 56 also need to be changed. Lines 300 and 310 make these adjustments.

At the end of line 310, we POKE the value 255 into location 36869. This tells the Video Interface Controller (VIC) that the character definitions are now located in RAM starting at 7168. Then in lines 320 through 380, we POKE in the values to define the characters we need to create the graphics. When we wish to return to the regular characters in line 850, we POKE 240 into memory location 36896. Then the VIC chip will look for character definitions at the old location

After making changes in the character definitions, we draw the playing screen in lines 390-430 by POKEing the necessary character values into screen memory.

The game section of the program comes next. After we randomly determine the plane's height and direction (lines 440-470) we enter a long FOR-NEXT loop (lines 480-770) that moves the plane across the screen. All other decisions (keyboard scanning, laser firing, tank movement, and occurence of hits) are made within this loop. By handling the program in this manner we have made the plane's motion automatic, depending only upon the variable of the FOR-NEXT loop.

Continued on p. 70



The IBM PCjr version of *Flak Attack* takes full advantage of the machine's color graphics capability. The DRAW command lets you create shapes easily, and then turn, expand, or even shrink them. You create shapes by using strings of characters as commands to the DRAW statement. You are, in effect, using commands within a command. Graphics representations are easy to implement using commands such as L for left, U7 for up seven, and so on. You can also change the scale of the object you are drawing right in the middle of a string with S*x* (*x* being the scale factor from 0 to 255). A scale of 4 is normal size, so 1 would be one-quarter size, and 8 would be double-size.

The shape can also be rotated to any angle permitted by the screen's resolution. The angle of rotation is set with TA*x* (for Turn Angle), where *x* is any number from −360 to 360. This function is demonstrated by the demolition of the cannon in line 1170. This line is a FOR–NEXT loop running from −360 to 360, expanding the shape as it rotates. The shape isn't erased before redrawing, so the effect is an expanding explosion of color.

If you don't erase your animated graphics before redrawing, they will leave a trail, and parts of the new shape that lie over the old shapes might be erased. You erase simply by changing the color with the C*x* command in the DRAW command string, where *x* is any number representing a legal color for the resolution being used. The cannonball in line 900 uses this process. First the coordinate is chosen using the PSET statement, and the old shape is erased. Then the coordinate is moved four pixels up the screen and the ball is redrawn. A designation of C0 after the DRAW command sets the color to black to match the background. The C2 specifies the color red, which is one of the four colors (green, red, black and brown) currently available in this mode.

One of the toughest bugs in this program showed up in the keyboard buffer. If a key is held down too long, the cannon will take off and not stop until the buffer of sixteen characters has been used. This problem is taken care of in line 710, which erases the keyboard buffer. If used every time the keyboard is read with INKEY$; as in line 700, the buffer will be cleared. This method lets you hold down a key for as long as you like, so that you can stop the cannon and fire the instant you release the key.

Continued on p. 72

The TI version of *Flak Attack* takes advantage of TI BASIC's graphics and color. Because there are no sprites in TI's console BASIC, it is not possible to get the smooth action of Extended BASIC. However, this does not hamper the game significantly. The scenario is the same as in the other versions, with a fighter plane firing at your surface guns. You have a barrier to hide behind (until it gets destroyed). If the barrier gets hit, part of it disappears, and the next shot at the same spot will go all the way through. But this hole in the barrier will also allow you to fire through it. You have only three surface guns, and once all three are destroyed the game ends. The S and D keys are used to move the cannon back and forth, and the E key is used to fire.

Flak Attack **(TI-99/4A)**
Explanation of the Program

Line nos	
100-160	Program header.
170-230	Title screen.
240-290	Initialize variables.
300-350	Input level of difficulty.
360	Stop the program if option 4 is selected.
370-430	Set up graphics characters and initialize game.
440-510	Set up graphics colors and display the playing screen.
520-610	Move plane and missile.
620-660	Read keyboard and branch to subroutines.
670-720	Move gun left.
730-780	Move gun right.
790-860	Initial firing of the cannon.
870-980	Initialize altitude and direction of plane.

Continued on p. 80

Flak Attack . . . from p. 61

Flak Attack (Apple)
Explanation of the Program

Line nos.	
100-170	Program header.
180-290	Initialize game; display options page.
300-370	Display playing screen.
380-450	Initialize plane and place on screen.
460-570	Player's move.
580-650	Move plane and check laser fire.
660-720	Initialize cannonball and place on screen.
730-810	Cannonball routine; check for hit.
820-930	Plane explosion routine.
940-1040	Fire laser.
1050-1140	Hit tank.
1150-1230	End of game options.
1240-1270	Read and display data.
1280-1320	Key scan routine. Reads keyboard but doesn't wait for key press. Scans once, returns to game.
1330-1410	Data for title screen.
1420-1740	Data for shape tables.
1490	Shape table index.
1530	Cannon.
1570	Cannonball.
1610-1620	Plane/eastbound.
1660	Plane/westbound.
1700	Plane explosion.
1740	Ground explosion.
1750-1800	Data for skill-level screen.
1810	End of program.

APPLE II Series

```
100 REM *****************
110 REM *   FLAK ATTACK   *
120 REM *****************
130 REM BY M. MOSELEY AND
140 REM THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.1.1
170 REM   APPLE II SERIES APPLESOFT
```

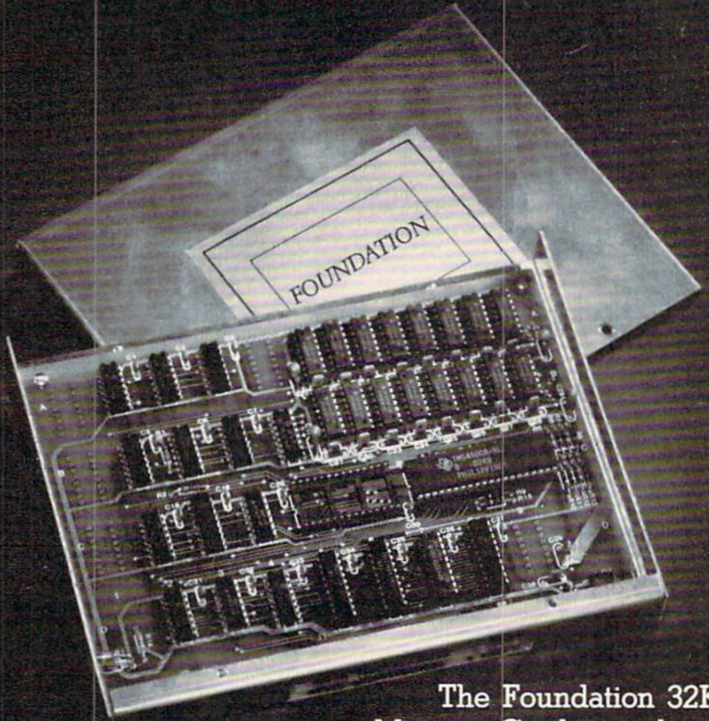
APPLE II Series

```
180 REM **********
190 REM INITIALIZE
200 REM **********
210 TEXT
220 YP = 141: ROT= 0:PS = 3:PT = 0: SCA
    LE= 1
230 HO(0) = 1:HO(10) = 1
240 HOME :NL = 6: GOSUB 1270: GET A$
250 FOR X = 24576 TO 24740: READ A: POK
    E X,A: NEXT X
260 POKE 232,0: POKE 233,96
270 HOME :NL = 5: GOSUB 1270
280 GET A$: IF A$ < "1" OR A$ > "3" OR
     LEN (A$) > 1 THEN GOTO 280
290 D = VAL (A$)
300 REM *****************
310 REM DISPLAY OPTION PAGE
320 REM *****************
330 HGR : HOME : HCOLOR= 6
340 VTAB 23: HTAB 1: PRINT "SCORE: ";
350 FOR X = 149 TO 159: HPLOT 0,X TO 27
    9,X: NEXT X
360 FOR X = 125 TO 130: HPLOT 124,X TO
    156,X: NEXT X
370 XDRAW 1 AT YP,147
380 REM ********************
390 REM INITIALIZE THE PLANE
400 REM ********************
410 ALT = INT ( RND (1) * 100 + 10)
420 DIR = INT ( RND (1) * 3 - 1): IF D
    IR = 0 THEN GOTO 420
430 IF DIR = - 1 THEN P = 3:PY = 16
440 IF DIR = 1 THEN P = 4:PY = 260
450 XDRAW P AT PY,ALT
460 REM ***************************
470 REM PLAYERS MOVE / SCAN KEYBOARD
480 REM ***************************
490 IF C = 1 THEN GOSUB 760
500 IF SD = 1 THEN SD = 0: GOTO 410
510 GOSUB 1310: VTAB 23: HTAB 8: PRINT
    PT;"          ": IF ST = 0 THEN GOTO
     610
520 IF KEY = 83 THEN HCP = - 4: GOTO 5
    60
530 IF KEY = 68 THEN HCP = 4: GOTO 560
540 IF KEY = 69 AND C = 0 THEN GOSUB 6
    90
550 GOTO 610
560 IF YP < 5 AND HCP = - 4 OR YP > 27
    4 AND HCP = 4 THEN GOTO 610
```

Continued on p. 69

NEED MORE MEMORY?

ORDER THE 128K MEMORY CARD BY FOUNDATION.



If you're writing programs and can use more memory. Or if you want to take advantage of large memory programs. The Foundation 128K Memory Card is just what you need.

Foundation has been shipping memory cards all over the country. And our customers have been calling us with nothing but praise. We're delighted, but not surprised.

Both our 32K and 128K Memory Cards begin and end with quality. From the design by our MIT-trained engineers. To using the finest components and prime integrated circuits from top manufacturers. Encased in a durable metal case that plugs into your Peripheral Expansion Box.

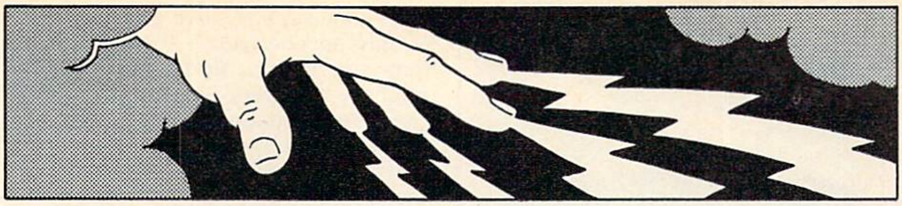
The Foundation 32K and 128K Memory Cards are designed from the beginning to be compatible with the TI99/4A Computer. The 128K Card comes with solid-state software that makes it easy to use the additional memory. Just write or call us and we'll send you a descriptive brochure.

The choice is yours. Order now and take advantage of Foundation's Special Introductory prices.*

The Foundation 32K Memory Card - $150.
The Foundation 128K Memory Card - $230.
Both with a full three month warranty.

FOUNDATION
74 Claire Way
Tiburon, CA 94920
415/388-3840

*Add 3% shipping and handling charges. California residents add 6% sales tax. Allow 2-4 weeks for delivery.



ZEUS

A Review by Greg Roberts

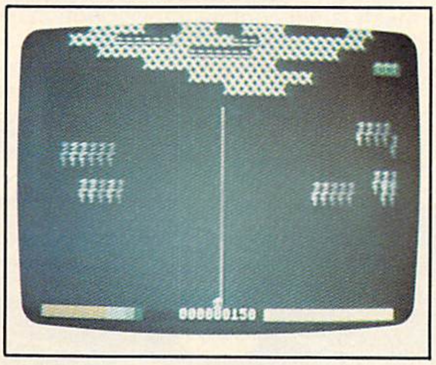
HCM Staff

Name:	Zeus
Program Type:	Arcade game
Machine:	Commodore 64
Distributor:	Aardvark Systems, Ltd. 2352 S. Commerce Rd. Walled Lake, MI 48088
Price:	$29.95, diskette $24.95, cassette

System Requirements:
Joysticks

Poor Fair Good Excellent

Performance
Engrossment
Documentation

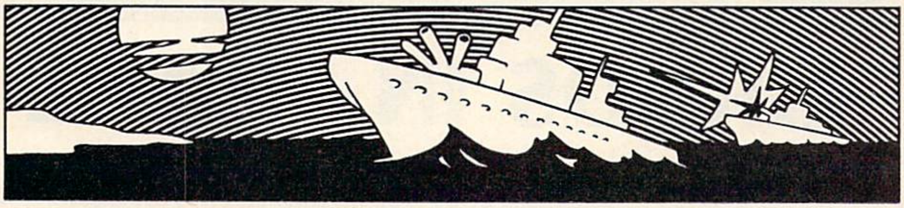


As advertised, this game is truly "fast and furious." The human eye and hand have a hard time following and blocking thunderbolts hurled down from the sky by an angry Zeus. You, the mighty wizard, can destroy them with cold rays, but if you hold down the fire button for more than a second or two, your power starts to fizzle. The strategy then, is to blast away when the bolts are dense and right overhead, and to hold fire when they are higher and more scattered. If you get in a real jam—with a thunderbolt at face level and a sick ray gun—you can pull back on the joystick and enclose yourself in a protective shield. To make this pay off, your timing must be extraordinary. We're talking about tenths of a second.

The first few levels of play are quite easy to achieve—a show of programming psychology I always welcome. There is no difference, other than speed, between one level and the next. The scene stays the same: a small Zeus figure in the clouds and a wizard on the ground. The clouds frequently flash different colors, but that's about the extent of the fireworks.

Zeus is a very simple action game that is best suited for the beginner. It does not require much strategy; rather it is a good diversion when you just want to blast at an enemy and to see how long you can survive.

HCM



BEACH-HEAD

A Review by Greg Roberts

HCM Staff

If Dad or Grandpa was part of the action in the Pacific in WWII, don't put on *Beach-Head* when these old boys are trying to sleep. It's bound to give them nightmares. For a computer game, it's as realistic as you can get.

Your first challenge in this program is to maneuver as many ships as you can (up to 10) through a hidden passage to the battle site. The channel here is full of mines and torpedoes, and the number of ships you bring through the passage helps determine your military strength

Continued on p. 66

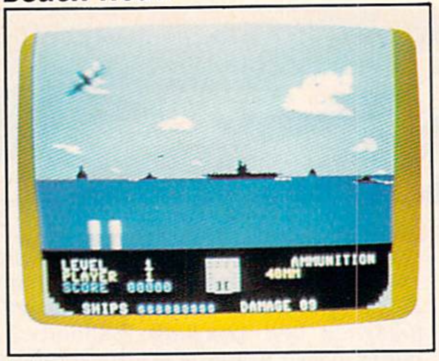
Name:	Beach-Head
Program Type:	Conventional Warfare
Author:	Bruce Carver
Machine:	Commodore 64
Distributor:	Access Software 925 East 900 South Salt Lake City, UT 84105
Price:	$34.95, diskette

System Requirements:
Joystick

Poor Fair Good Excellent

Performance
Engrossment
Documentation

Beach Head



for the next scene—in fact for the rest of the game.

In stage two, you find yourself manning the anti-aircraft guns. Looking out over calm seas, you see pinpoints enlarging into fighter planes on the attack. You try to lead their flight with your guns as they come closer and closer and finally fly directly overhead to shell you without letup.

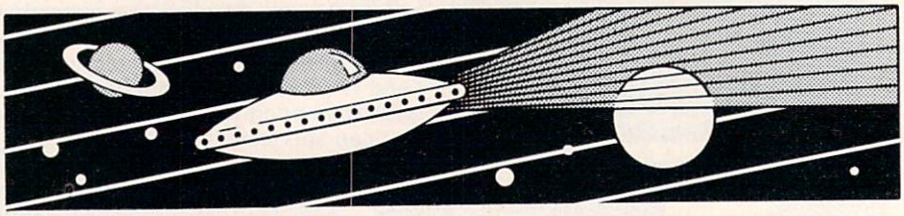
If you can destroy enough planes without losing all your ships, you are ready to use your heavy guns on the enemy fleet while they fire back. Once the enemy ships are destroyed, you can make a beach landing. Each of your remaining ships carries two tanks, and you must take them one-by-one through hazardous terrain towards a final fortress. In this climactic battle you must hit ten targets on the fortress in order to destroy it. (The targets appear only intermittently.) Oh, yes, the enemy's cannon occasionally fires back—and it never misses.

A cliche such as "five games in one" is only an annoyance these days, but that expression does describe the radically different screens in *Beach-Head*. Any one screen would make a decent game all by itself. The animation is outstanding, especially the way the incoming planes tilt and turn away from your ship. And the Commodore's unique reservoir of sounds is indulged in wisely. Carver is a master at making engines and missiles fade in and out in realistic fashion.

Another sophisticated feature: The ships saved in game one determine your staying power in screens two and three, setting your tank quota for game four, and ultimately measuring your military might for attacking the fortress. To my mind, this kind of continuity puts *Beach-Head* in a class above the other battle games on the market.

This program comes with a little booklet that explains the play in detail. It offers tips for improving your score, plus instructions for adjusting the sound and color on your console.

Some people may dislike this game simply because it's about war and destruction. That's certainly a valid point, and will deter some players no matter how good the programming. A much larger contingent will soon make *Beach-Head* a hit. Very likely the day will come—if it is not already here—when gamers will buy a package just from reading the name Bruce Carver. **HCM**

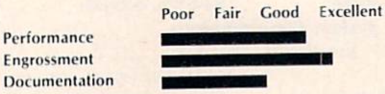


GRUDS IN SPACE

A Review by Janet LaFave

Name:	**Gruds In Space**
Program Type:	**Adventure game**
Authors:	**Chuck Sommerville and Joe Dudar**
Machine:	**Apple II,II+,IIe**
Distributor:	**Sirius Software 10364 Rockingham Dr. Sacramento, CA 95827**
Price:	**$39.95, diskette**

System Requirements:
48K memory, 1 disk drive



Adventure games have come a long way in the past year or so. Once upon a time you had to live the adventure through text and imagination, with infrequent visits from crudely drawn graphics characters. Throughout most of the game—and this could add up to weeks or months of play—the screen stayed static and dull, showing only its enigmatic text.

These days, adventure programmers have a hard time competing in the games market unless their scenarios are fully illustrated—a point well understood by Sirius Software. Their latest effort, *Gruds in Space*, uses excellent graphics from beginning to end. These range from stark high-tech rooms and laboratories to views of planets in space. All are well rendered, as if by an architectural draftsman, at the least. Best of all, the Gruds, military commanders, and other

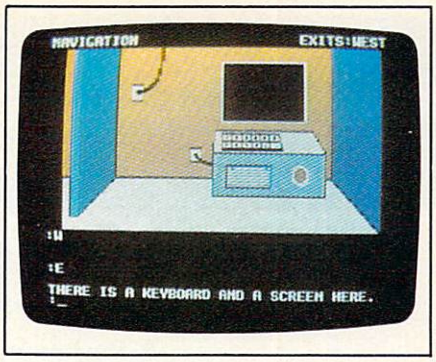
Gruds

cartoon characters are designed with a humorous touch.

The game certainly has an expansive feel: Not only does it take you through buildings with many rooms, passageways and stairwells, but it forces you to teleport yourself to other planets. You are called upon to deliver fuel to a stranded spy pilot on Pluto, and you end up planet-hopping via a teleport device that asks for a complex series of coordinates for each planet. Your reward for a successful mission? One million dollars. Such a grand interplanetary scale of play is more appealing, at least to this reviewer, than scenarios confined to a few claustrophobic crypts.

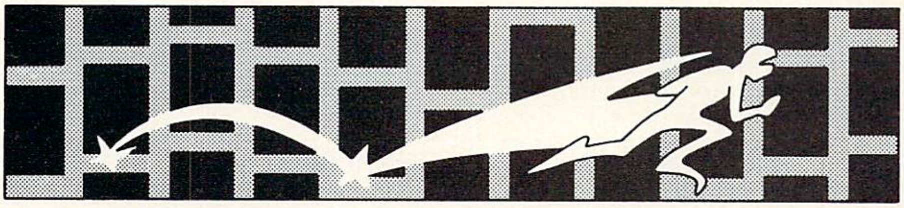
Like all good adventure games, *Gruds* gives up its goods grudgingly. One of the first rooms you walk into is fitted out with a keyboard and screen—but learning how to use that keyboard is not at all obvious. Note: You will need a color monitor for this game, not only because the graphics show good use of the spectrum, but because you have to choose between different colored buttons on control panels featured in the play.

This game has one merciful feature designed to cut down on frustration: It shows the available directional exits in



the upper right-hand corner of the screen. It should keep you from having to put up with a barrage of CAN'T GO THAT WAY messages.

As in any good adventure game, you should be ready to commit considerable time to playing it. Be sure to keep a detailed map of your progress, and save your game on a separate diskette. If you are ready to immerse yourself in such play, you will find *Gruds In Space* one of the top-shelf programs of its kind. Even hardened adventure fans will find this game sufficiently entangling to make it playable for weeks. . .or years? HCM



JUMPMAN JUNIOR

A Review by Judy Sanoian
HCM Staff

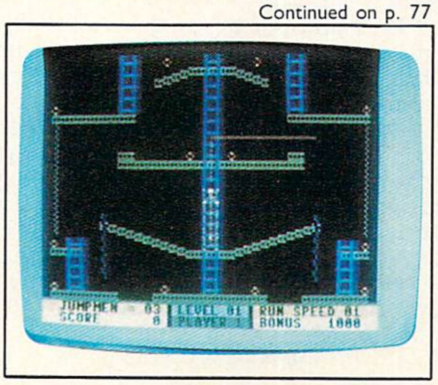
Name:	**Jumpman Jr.**
Program Type:	**Ladder Climbing**
Author:	**Randy Glover**
Machine:	**Commodore 64**
Distributor:	**Epyx Computer Software, 1043 Kiel Court, Sunnyvale, CA 94086**
Price:	**$39.95, cartridge**
System Requirements:	**Joysticks**

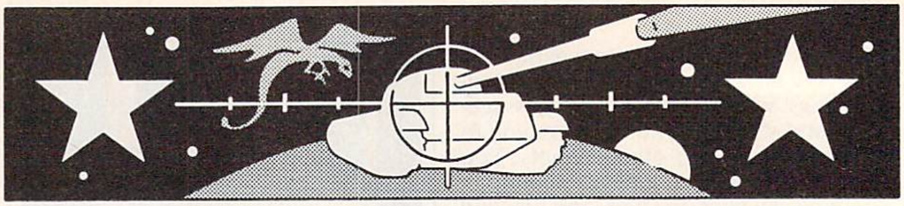
Poor Fair Good Excellent
Performance
Engrossment
Documentation

If you're an old-timer in the computer game game, the first screen of *Jumpman Jr.* will immediately ring a bell. The blinking character scurrying up the rungs of the ladder, the dramatic leaps from one tier to the next. . .all that's missing are the hairy ape and the fair maiden. But *Jumpman Jr.* can't be shrugged off as just another *Donkey Kong* rip-off. For one thing, it offers a completely different scenario; we're defusing bombs instead of rescuing bombshells. For another, its sound is superior: harmonious tunes and crackling fires instead of quick blips and bleeps. *Jumpman Jr.* also excels when it comes to graphics. Compare the action of the falling jumpman—who tumbles head over heels, bouncing and ricocheting off tiers and ladders on his way down—with the quick splat in *Donkey Kong.*

For the game player, however, the important question is not whether *Jumpman Jr.* resembles *Donkey Kong* or the legions of *Kong* look-alikes. What really matters is whether it is an exciting game in its own right. I found *Jumpman Jr.* challenging, fun to play, and visually and aurally attractive to boot. Its animation and response are quite good, and

Continued on p. 77





STELLAR SEVEN

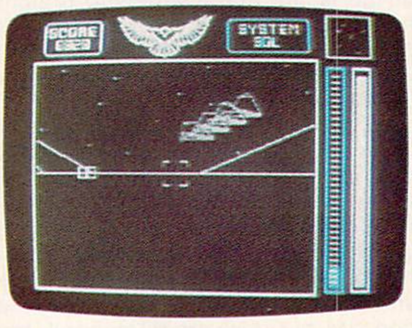
A Review by Greg Roberts
HCM Staff

Name:	Stellar Seven
Program Type:	Space War
Author:	Damon Slye
Machine:	Apple IIe
Distributor:	Software Entertainment Co. 541 Willamette St. Eugene, OR 97401
Price:	$34.95, diskette

	Poor	Fair	Good	Excellent
Performance				
Engrossment				
Documentation				

This one is a direct take-off of the well-known arcade game, *Battle Zone*, although the alien enemies have been redesigned in this new home computer version.

In *Stellar Seven* you see the galaxy through the window of an armored vehicle, the Raven, which was sent to destroy enemies on seven separate moons or satellites. You fire your biphasal thunder cannon until you've killed all the foes on a satellite, at which time you can teleport to the next planet via a strange phenomenon called a Warplink. If you can kill enough aliens to achieve the seventh Warplink, the one to Arcturus, you win the game.



The graphics in this program will really turn your head. They are not colorful (in fact, the simple light lines are just as interesting on a monochrome monitor as on a color model), but they are three-dimensional. Beasts resembling pterodactyls swoop from behind you, looking huge for the moment, then disappearing on the horizon. Bizarre tanks and other instruments of destruction roll in, out, and away from your window in realistic fashion. Each kind of enemy is intricately described in an excellent illustrated briefing feature, and further details are furnished in a brief but adequate little booklet.

You locate the enemies with your gravitic scope, confuse them with your inviso cloak, and protect yourself with protonic shields. Unfortunately, it is not always convenient to get at some of these defense mechanisms. You have to use three separate keys in addition to your joystick. If you are playing the keyboard alone, you will find it just about impossible to manage some of the defense keys.

Positioning your crosshairs on the enemy can be tricky, but don't be afraid to try long shots. You will be surprised at your ability to destroy enemies at extreme ranges.

Take a Ride

After you've killed a certain number of aliens, you get a chance at the Warplink. The sight of this star-shaped object is always the high point of any screen; you see it grow larger and larger, filling the display. Then the terrain starts moving faster and faster as you get ready to teleport from one planet to the next.

You arrive on the next planet with your protonic shields partially restored, and certainly you'll need all the protection you can get. Even on the second level the enemy is much faster and more deadly than before. Most players, upon reaching this level for the first time, last for just a few seconds.

As an example of traditional Apple graphics, this game is very good. But now that the graphics for this machine are being thoroughly revamped, some elements of *Stellar 7* may seem primitive by comparison. Besides the near absence of color, the animation in this game is often jerky and uneven. The "explosion" of an alien ship is simply the breaking apart of a few straight lines, accompanied by sound effects resembling the weak and uneven crackling of my grandpa's old Sylvania radio.

Nonetheless, the game's three-dimensional spaceships and all-terrain vehicles, drawn with an eerie spareness, will always be fascinating—no matter what other graphics options become available on the Apple.

HCM

Flak Attack . . . **from p. 63**

APPLE II Series

```
570  XDRAW 1 AT YP,147: XDRAW 1 AT YP +
     HCP,147:YP = YP + HCP
580  REM ****************
590  REM MOVE THE AIRPLANE
600  REM ****************
610  IF PY < 10 OR PY > 270 THEN HCOLOR
     = 0: DRAW P AT PY,ALT:PT = PT - D *
     2: GOTO 410
620  HCOLOR= 0: DRAW P AT PY,ALT: HCOLOR
     = 5: DRAW P AT PY + (D * 2 * ( - DI
     R)),ALT:PY = PY + (D * 2 * ( - DIR)
     )
630  IF PY - 15 < LP AND PY + 15 > LP AN
     D INT ( RND (1) * (16 - D * 2)) =
     2 THEN GOTO 970
640  IF INT ( RND (1) * (180 - D * 40))
     = 10 THEN GOTO 970
650  GOTO 490
660  REM **********************
670  REM INITIALIZE CANNON BALL
680  REM **********************
690  IF YP > 119 AND YP < 158 THEN IF H
     O( INT ((YP - 120) / 4) + 1) = 0 TH
     EN RETURN
700  FOR X = 1 TO 10:ZZ = PEEK ( - 1633
     6): NEXT X
710  LP = YP
720  C = 1: HCOLOR= 3: DRAW 2 AT LP + 3,
     141:CB = 141: RETURN
730  REM ********************
740  REM MOVE THE CANNON BALL
750  REM ********************
760  HCOLOR= 0: DRAW 2 AT LP + 3,CB: HCO
     LOR= 3: DRAW 2 AT LP + 3,CB - 4:CB
     = CB - 4
770  IF CB > ALT + 2 THEN RETURN
780  IF CB < ALT - 6 OR CB < 6 THEN HCO
     LOR= 0: DRAW 2 AT LP + 3,CB:C = 0:
     RETURN
790  IF DIR = - 1 AND LP < PY + 15 AND
     LP > PY - 1 THEN GOTO 850
800  IF DIR = 1 AND LP < PY + 1 AND LP >
     PY - 15 THEN GOTO 850
810  RETURN
820  REM ************************
830  REM PLANE HIT BY CANNON BALL
840  REM ************************
850  HCOLOR= 0: DRAW 2 AT LP + 3,CB
860  DRAW P AT PY,ALT:P = 5
870  IF PY > 120 AND PY < 160 THEN LD =
     115: GOTO 890
880  LD = 148
890  R = 0: FOR ALT = ALT TO LD STEP 3:R
     = R + 16: IF R = 64 THEN R = 0: RO
     T= R: HCOLOR= 5: DRAW P AT PY,ALT
900  ZZ = PEEK ( - 16336): HCOLOR= 0: D
     RAW P AT PY,ALT: NEXT ALT
910  C = 0: HCOLOR= 0: DRAW P AT PY,ALT
920  SD = 1
930  ROT= 0:PT = PT + (200 - ALT):R = 0:
     HCOLOR= 3: DRAW 1 AT YP,147: RETUR
     N
940  REM *****************
950  REM PLANE FIRES LASER
960  REM *****************
970  IF PY > 123 AND PY < 157 THEN IF H
     O( INT ((PY - 126) / 4) + 1) = 0 TH
     EN LA = 125: GOTO 990
980  LA = 148
```

APPLE II Series

```
990  HCOLOR= 3: HPLOT PY,ALT TO PY,LA: F
     OR X = 1 TO 10:ZZ = PEEK ( - 16336
     ): NEXT X: HCOLOR= 0: HPLOT PY,ALT
     TO PY,LA
1000 HCOLOR= 5: DRAW 6 AT PY,LA: FOR X =
     1 TO 10:ZZ = PEEK ( - 16336): NEX
     T X: HCOLOR= 0: DRAW 6 AT PY,LA
1010 IF PY > 123 AND PY < 157 THEN HO( I
     NT ((PY - 126) / 4) + 1) = 1
1020 IF LA = 125 THEN GOTO 490
1030 IF PY - 5 < YP AND PY + 5 > YP THEN
     GOTO 1080
1040 GOTO 490
1050 REM *****************
1060 REM TANK HIT BY LASER
1070 REM *****************
1080 FOR X = 1 TO 20: HCOLOR= 5: DRAW 1
     AT YP,147:R = R + 8:ZZ = PEEK ( -
     16336)
1090 HCOLOR= 0: DRAW 5 AT YP,147: ROT= R
     : SCALE= INT (X / 4) + 1: NEXT X
1100 PS = PS - 1: IF PS = 0 THEN GOTO 1
     180
1110 PT = PT - 100:YP = 141:R = 0: SCALE
     = 1
1120 ROT= 0:C = 0
1130 FOR X = 0 TO 10:HO(X) = 0: NEXT X
1140 GOTO 310
1150 REM *******************
1160 REM END OF GAME OPTIONS
1170 REM *******************
1180 HOME : TEXT : PRINT "YOUR FINAL SCO
     RE IS :";PT
1190 FOR X = 0 TO 10:HO(X) = 0: NEXT X
1200 PRINT : PRINT "WOULD YOU LIKE TO PL
     AY AGAIN (Y/N)?";
1210 GET A$: IF A$ = "N" THEN GOTO 1810
1220 IF A$ = "Y" THEN CLEAR : GOTO 220
1230 GOTO 1210
1240 REM *****************************
1250 REM READ DATA AND DISPLAY ROUTINE
1260 REM *****************************
1270 FOR PD = 1 TO NL: READ A,B,A$: VTAB
     A: HTAB B: PRINT A$;: NEXT PD: RET
     URN
1280 REM ***********************
1290 REM KEY SCAN ROUTINE / DON'T WAIT
     FOR KEY LIKE THE "GET" STATEMENT DO
     ES
1300 REM  ***********************
1310 KEY = PEEK ( - 16384): IF KEY > 12
     7 THEN ST = 1:KEY = KEY - 128: POKE
     - 16368,0: RETURN
1320 ST = 0: RETURN
1330 REM *****************************
1340 REM DATA FOR TITLE SCREEN DISPLAY
1350 REM *****************************
1360 DATA 10,10,*********************
1370 DATA 11,10,*                   *
1380 DATA 12,10,*     FLAK ATTACK   *
1390 DATA 13,10,*                   *
1400 DATA 14,10,*********************
1410 DATA 24,15,PRESS ENTER
1420 REM ******************************
1430 REM
1440 REM THE FOLLOWING DATA IS USED IN
     THE BUILDING OF THE SHAPE TABLES
1450 REM
```

Continued on p. 70

Flak Attack . . . from p. 69

APPLE II Series

```
1460 REM   ****************************
1470 REM   SHAPE TABLE INDEX
1480 REM   *****************
1490 DATA    6,0,15,0,31,0,38,0,75,0,112,0
,143,0
1500 REM   **************
1510 REM   PLAYERS CANNON
1520 REM   **************
1530 DATA    63,63,55,45,45,45,45,37,63,63
,39,36,44,54,54,0
1540 REM   ***********
1550 REM   CANNON BALL
1560 REM   ***********
1570 DATA    45,36,63,54,6,0,0
1580 REM   *****************
1590 REM   PLANE / EASTBOUND
1600 REM   *****************
1610 DATA    45,45,45,45,45,45,45,44,39,63
,60,63,63,63,63,60,60,60,54,54,46,3
6
1620 DATA    44,54,46,44,45,45,45,53,45,63
,63,63,63,63,0
1630 REM   *****************
1640 REM   PLANE / WESTBOUND
1650 REM   *****************
1660 DATA    63,63,63,63,63,63,63,60,37,45
,44,45,45,45,45,44,44,44,54,54,62,3
6,60,54,62,60,63,63,63,55,63,45,45,
45,45,45,0
1670 REM   **************
1680 REM   PLANE EXPLOSION
1690 REM   ***************
1700 DATA    44,44,45,44,53,55,63,55,55,53
,45,54,63,60,60,55,54,54,39,37,37,4
4,37,36,63,60,55,53,45,46,0
1710 REM   ****************
1720 REM   GROUND EXPLOSION
1730 REM   ****************
1740 DATA    137,22,141,30,30,39,36,36,55,
54,54,39,36,36,55,54,54,231,12,220,
68,5,0
1750 REM   ***********************
1760 REM
1770 REM   DATA FOR LEVEL OPTIONS PAGE
1780 REM
1790 REM   ***********************
1800 DATA    2,1,SKILL LEVEL,5,3,1. BEGIN
NER,7,3,2. NOVICE,9,3,3. PRO,24,1,"
YOUR CHOICE?"
1810 END
```

HCM

Flak Attack . . . from p. 61

Flak Attack (VIC-20)
Explanation of the Program

Line nos.	
100-160	Program header.
180-280	Opening screen and menu.
290-380	Clear screen, turn it cyan; move character set and re-define characters.
390-430	Draw game screen.
440-470	Determine plane direction and height; set variables to move plane.
480-510	Enter main loop; move plane.
520-550	If rocket already fired, move rocket; check for hit.
560-590	Keyscan; go to appropriate subroutine.
600-630	Routine to initiate rocket fire.
630-660	Randomly decide if laser should fire; go to indicated routine.
670-760	Check location of laser fire: Branch if necessary.
770-790	End of mainloop; delete plane and go back to 750.
800-880	Routine for tank hit; destroy tank; see if three have been destroyed. If yes, display final score, go to menu; if no, start with new tank at line 740.
2000-2110	Routine to destroy plane, award points, display score, reinitialize variables and RETURN.
3000-3030	Routine to move tank left.
4000-4030	Routine to move tank right.

VIC-20

```
100 REM ************
110 REM *FLAK ATTACK*
120 REM ************
130 REM BY MARK MOSELEY AND THE HCM STA
FF
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.1.1
160 REM V20 BASIC
170 REM
180 PRINT"{SHIFT CLR}          FLAK ATTACK"
190 PRINT "{3CRSRDOWN}          PRESS RETURN
 TO          BEGIN"
200 GETA$:IFA$=""THEN200
210 IFA$<>CHR$(13)THEN200
220 PRINT"{SHIFT CLR}LEVEL OF DIFFICULT
Y:{CRSRDOWN}"
230 PRINT"(1) BEGINNER{CRSRDOWN}":PRINT
"(2) NOVICE{CRSRDOWN}":PRINT"(3) IN
TERMEDIATE{CRSRDOWN}":PRINT"(4) PRO
"
240 PRINT"{CRSRDOWN}(5) END GAME"
250 GETDF$:IFDF$=""THEN250
260 DF=ASC(DF$):IFDF<49ORDF>53THEN250
270 DF=10*(VAL(DF$))
280 IFDF$="5"THEN:PRINT"{SHIFT CLR}":PO
KE36869,240:END
290 PRINT"{SHIFT CLR}":POKE 36879,59
300 POKE52,28:POKE56,28
310 FORI=7168TO7679:POKEI,PEEK(I+25600)
:NEXT:POKE 36869,255
320 FORI=7216TO7287
330 READN
340 POKEI,N:NEXTI
350 DATA 255,255,255,255,0,0,0,0,255,60
,255,60,0,255,66,255,24,24,24,24,24
,24,24,24
360 DATA 24,24,24,24,24,24,60,126,96,48
,152,255,255,152,48,96,6,12,25,255,
255,25,12,6
```

VIC-20

```
370 DATA 7,31,127,255,255,127,63,15,255
    ,255,255,255,255,255,255,255
380 DATA 224,248,254,255,255,254,252,22
    4
390 FORN=8142 TO 8185:POKEN,13:POKE N+3
    0720,5:NEXTN
400 FOR N=38400TO38861:POKEN,0:NEXTN
410 FORN=8062TO8066:POKEN,6:NEXTN
420 SC=0:TN=0
430 TK=8108:TC=10:RK=TK:POKETK,9:FORN=1
    2TO14:POKE8117+N,N:NEXTN
440 PH=INT(RND(0)*12)+2:DR=INT(RND(0)*2
    )
450 IF DR=1 THEN GOTO470
460 I=0:J=21:C=1:GOTO480
470 I=21:J=0:C=-1:GOTO480
480 FOR CM=ITOJSTEPC
490 PP=(7680+CM+(22*PH))
500 IF DR=1 THEN POKEPP,11:POKEPP+1,32:
    GOTO520
510 POKEPP,10:POKEPP-1,32
520 IFF=0THENFORD=1TO35:NEXTD:GOTO560
530 POKE RK,32:RK=RK-22:POKE RK,9
540 IF(PH-1)>(RK-TT-7680)/22THENPOKERK,
    32:RK=TK:F=0
550 IFRK=PPORRK=PP+22ORPP=RK-CTHEN2000
560 K=PEEK(197):IF K=64 THEN 630
570 IFK=41THENGOSUB3000
580 IFK=18THENGOSUB4000
590 IFK<>49THEN630
600 IFF=1THEN630
610 IFPEEK(TK-44)<>32THEN630
620 F=1:RK=TK:RK=RK-22:POKE RK,9:TT=TC
630 Q=INT(RND(0)*500)+1
640 IFCM=TTTHEN660
650 IFQ>DFTHEN 770
660 IFQ>DF*20THEN 770
670 IFPEEK(8054+CM)<>32THENTR=(8054+CM)
    :GOTO 710
680 IFCM<8ORCM>12THEN 690
690 IFCM>(TC-2)ANDCM<(TC+2)THENHT=1:TR=
    TK+1:GOTO 710
700 TR=8141
710 POKE36877,230:POKE36878,15:FORN=PP+
    22TOTRSTEP22:POKEN,8:NEXTN
720 FORN=PP+22TOTRSTEP22
730 IFN=RKANDF=1THENF=0:RK=TK
740 POKEN,32:NEXTN
750 IFHT=1THEN 800
760 POKE36877,0:POKE36878,0
770 NEXT CM
780 POKE PP,32
790 GOTO 440
800 POKE36877,200:TN=TN+1:HT=0
810 POKETK+30720,2:POKETK,7:FORN=TK+21T
    OTK+23:POKEN+30720,2:POKEN,7:NEXTN
820 FORD=1TO200:NEXTD:POKE36877,220:FOR
    D=1TO500:NEXTD:POKE36877,0:POKE3687
    8,0
830 POKE TK,32:POKETK+30720,0:FORN=TK+2
    1TOTK+23:POKEN,32:POKEN+30720,0:NEX
    TN
840 POKEPP,32:POKERK,32:F=0:IFTN<>3THEN
    430
850 PRINT"{SHIFT CLR}":RESTORE:POKE3686
    9,240:PRINT "{5CRSRDOWN}YOUR FINAL
    SCORE:";SC
860 PRINT"{3CRSRDOWN}PRESS RETURN TO
             CONTINUE"
870 GETA$:IFA$=""THEN 870
```

VIC-20

```
880 IFA$<>CHR$(13)THEN 870
890 GOTO220
2000 IFRK=PPTHENPOKERK,7:POKERK+30720,2:
     GOTO2020
2010 POKERK,32:POKEPP,7:POKEPP+30720,2
2020 POKE36878,15
2030 POKE36877,200:FORD=1TO750:NEXTD
2040 POKE36878,0
2050 POKEPP,32:POKEPP+30720,0:POKERK+307
     20,0
2060 IFPH>10THENSC=SC+5:GOTO2100
2070 IFPH>7THENSC=SC+10:GOTO2100
2080 IFPH>4THENSC=SC+15:GOTO2100
2090 SC=SC+20
2100 PRINT"{HOME}           {7SHIFT CRSRLEFT
     }";SC
2110 F=0:RK=TK:GOTO440
3000 IFTC=1THENRETURN
3010 POKETK-1,9:POKE TK,32
3020 FORN=12 TO 14:POKE (TK+N+8),N:NEXTN
3030 POKE TK+23,32:TK=TK-1:TC=TC-1:RETUR
     N
4000 IF TC=20 THEN RETURN
4010 POKE TK+1,9:POKETK,32
4020 FORN=12TO14:POKE(TK+N+10),N:NEXTN
4030 POKE TK+21,32:TK=TK+1:TC=TC+1:RETUR
     N
```

HCM

Flak Attack . . . from p. 61

Flak Attack (IBM PCjr) Explanation of the Program

Line nos.	
100-180	Program header.
190-290	Display menu screen and input player's choice.
300-530	Set up string variables for shapes to be drawn.
550-650	Set up screen, and initialize flight of plane.
650-770	Input player's moves via keyboard.
780-830	Move plane.
840-880	Initialize cannonball.
890-940	Move cannonball.
950-1040	Hit plane with cannonball, destroy plane.
1050-1130	Plane fires laser at tank.
1140-1180	Tank blows up from direct hit.
1190-1250	Option to play again.
1260-1290	End of game.

IBM PCjr

```
100 REM ***********
110 REM * FLAK ATTACK *
120 REM ***********
130 REM BY MARK MOSELEY
140 REM AND THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.1.1
170 REM PC BASICA
180 REM PCjr CARTRIDGE BASIC
190 REM ***********
```

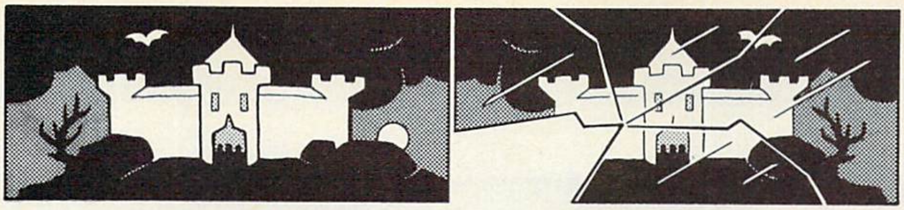
Continued on p. 72

Flak Attack . . . from p. 71

IBM PCjr

```
200 REM MENU SCREEN
210 REM ***********
220 PLAYER=3
230 CLS:SCREEN 1:LOCATE 1,14:PRINT "FLAK ATTACK"
240 LOCATE 3,1:PRINT "LEVEL OF DIFICULTY":LOCATE 5,4:PRINT "1. BEGINNER":LOCATE 7,4:PRINT "2. NOVICE":LOCATE 9,4:PRINT "3. PRO":LOCATE 12,4:PRINT "4. END THE GAME"
250 LOCATE 22,14:PRINT "YOUR CHOICE?"
260 A$=INKEY$:IF A$="" THEN GOTO 260
270 IF A$<"1" OR A$>"4" THEN GOTO 260
280 IF A$="4" THEN GOTO 1290
290 DIF=VAL(A$):COLOR 0,0
300 REM **************
310 REM PLAYERS CANNON
320 REM **************
330 CAN$="D2R5DL10UR4U3R2D2"
340 REM ***********
350 REM CANNON BALL
360 REM ***********
370 CBALL$="U2FDGHUE"
380 REM ****************
390 REM PLANE GOING EAST
400 REM ****************
410 P$(0)="R11ERENRHLHL3GL15H3L2D3FDR3FR7EG4R4E4"
420 REM ****************
430 REM PLANE GOING WEST
440 REM ****************
450 P$(1)="L11HLHNLERER3FR15E3R2D3GDL3GL7HF4L4H4"
460 REM ***************
470 REM LASER EXPLOSION
480 REM ***************
490 EX$="RUEUHULLDGDFDRUEUHGDRU"
500 REM ****************
510 REM CANNON EXPLOSION
520 REM ****************
530 CEX$="URD2L2U3R3D4L4U4"
540 REM **********************
550 REM DRAW SCREEN/INIT PLANE
560 REM **********************
570 CLS
580 LOCATE 1,1:PRINT "SCORE:";SCORE
590 LINE (0,190)-(319,199),1,BF:LINE (144,175)-(176,180),1,BF
600 PC=160:CP=160:PSET (CP,186):DRAW "C3"+CAN$
610 ALT=INT(RND*135+20)
620 DIR=INT(RND*3)-1:IF DIR=0 THEN GOTO 620
630 IF DIR=-1 THEN P=1:PP=305:GOTO 650
640 P=0:PP=15
650 PSET (PP,ALT):DRAW "C2"+P$(P)
660 REM *********************
670 REM PLAYER MAKES HIS MOVE
680 REM *********************
690 IF C=1 THEN GOSUB 920
700 A$=INKEY$:IF A$="" THEN GOTO 810
710 DEF SEG=0:POKE 1050,PEEK(1052)
720 IF A$="S" THEN CP=CP-3:GOTO 750
730 IF A$="D" THEN CP=CP+3:GOTO 750
740 IF A$="E" AND C=0 THEN GOSUB 870 ELSE GOTO 810
750 IF CP<4 THEN CP=4:PC=4:PRINT CHR$(7);
760 IF CP>316 THEN CP=316:PC=316:PRINT CHR$(7);
770 PSET (PC,186):DRAW "C0"+CAN$:PSET (CP,186):DRAW "C3"+CAN$:PC=CP
780 REM **************
790 REM MOVE THE PLANE
800 REM **************
810 PSET (PP,ALT):DRAW "C0"+P$(P):PP=PP+(DIR*2*DIF):IF PP>15 AND PP<305 THEN PSET (PP,ALT):DRAW "C2"+P$(P) ELSE GOTO 610
820 IF PP-20<LP AND PP+20>LP AND INT(RND*16/DIF)=2 THEN GOTO 1080
830 IF INT(RND*200/DIF)=10 THEN GOTO 1080 ELSE GOTO 690
840 REM ***********************
850 REM INITIALIZE THE CANNON BALL
860 REM ***********************
870 IF CP>144 AND CP<176 THEN IF HO(INT(CP-144)/3)=0 THEN PRINT CHR$(7);:RETURN
880 C=1:SOUND 110,3:PSET (CP,182):DRAW "C2"+CBALL$:LP=CP:CB=182:RETURN
890 REM ********************
900 REM MOVE THE CANNON BALL
910 REM ********************
920 PSET (LP,CB):DRAW "C0"+CBALL$:CB=CB-4:PSET (LP,CB):DRAW "C2"+CBALL$
930 IF CB<ALT-4 THEN PSET (LP,CB):DRAW "C0"+CBALL$:C=0:RETURN
940 IF CB<ALT+4 AND PP-11<LP AND PP+11>LP THEN GOTO 980 ELSE RETURN
950 REM ********************
960 REM PLANE GETS SHOT DOWN
970 REM ********************
980 IF DIR=1 THEN A$="A3" ELSE A$="A1"
990 PSET (LP,CB):DRAW "C0"+CBALL$:SCORE=SCORE+200-ALT
1000 LOCATE 1,1:PRINT "SCORE:";SCORE
1010 SOUND 110,10:PSET (PP,ALT):DRAW "C0"+P$(P)+A$
1020 FOR ALT=ALT TO 170 STEP 8:PSET (PP,ALT):DRAW "C0"+P$(P):PSET (PP,ALT+8):DRAW "C2"+P$(P):SOUND 1000-ALT*5,6:NEXT ALT
1030 C=0:PSET (PP,ALT):DRAW "C0"+P$(P)+"A0"
1040 GOTO 610
1050 REM *********************
1060 REM PLANE FIRES THE LASER
1070 REM *********************
1080 IF PP>143 AND PP<177 THEN IF HO(INT((PP-144)/3))=0 THEN L=180 :GOTO 1100
1090 L=195
1100 LINE (PP,ALT+2)-(PP,L),2:SOUND 110,1:SOUND 220,1:SOUND 110,1:PSET (PP,L):DRAW "C2"+EX$:PSET (PP,L-1):DRAW "C0"+EX$:LINE (PP,ALT+2)-(PP,L),0
1110 IF PP>143 AND PP<177 THEN HO(INT((PP-144)/3))=1
1120 IF PP-4<CP AND PP+4>CP AND L>180 THEN GOTO 1170
1130 GOTO 690
1140 REM *******************
1150 REM CANNON GETS BLOWN UP
1160 REM *******************
1170 FOR X=-360 TO 360 STEP 25:PSET (CP,186):A$="TA"+STR$(X)+"S"+STR$(INT((X+380)/10)):DRAW "C2"+A$+CEX$:SOUND RND*200+100,2:NEXT
```

Continued on p. 78



THE COVETED MIRROR

A Review by Greg Roberts
HCM Staff

Name:	The Coveted Mirror
Program Type:	Adventure game
Authors:	Eagle Berns and Holly Thomason
Machine:	Apple II, II+, IIe
Distributor:	Penguin Software Post Office Box 311 Geneva, IL 60134
Price:	$19.95, diskette

System Requirements:
48k memory, 1 disk drive

Poor Fair Good Excellent

Performance
Engrossment
Documentation

Here is another illustrated adventure game that exemplifies the trend in this genre towards color, and lots of it. The makers of *The Coveted Mirror* claim that this one features "animation in virtually every frame." In fact, the animation consists mostly of eyes rolling or fingers wagging in an otherwise paralyzed frame. In addition, nearly half of every screen is needlessly taken up by the game title and an hourglass. But that isn't so important when you consider how bright and well-drawn the illustrations are. What's more, the story is well-written and intriguing.

The village of Starbury was once protected by a kind wizard who drew his power from a dazzling mirror. Now the mirror is broken into five pieces, four of which are in the hands of the evil King Voar.The fifth piece is lost. If you, the champion of this game, can find the fifth piece, you have a chance at defeating Voar. If the evil one should get the remaining mirror piece, his power will be absolute.

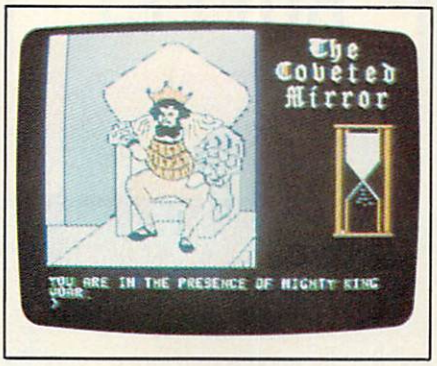
You always start your quest before the throne of Voar. No matter what you say, his dark sensibilities will be angered, and you will end up in the dungeon. Throughout the game, you risk being sent to the dungeon, and the program keeps track of how many times you go there. Twenty-five is your limit; if you reach it, the game ends.

Once you find your way out of the dungeon and into the street, the scenery goes by quickly. The buildings and the people are wonderful to behold, but your path gets unimaginably convoluted. How are you to know when to pick up such objects as horseshoes or crystal balls? You may find out too late that you need them. If you slip up and find yourself again in front of King Voar, you will lose the treasures you've picked up.

The many people you meet on the street can be most helpful. The simple commands TALK and HELP can be useful here. The game shows its flexibility in bringing on different characters each time you return to the street. *The Coveted Mirror* may be confusing, but you could never accuse it of being static. And even when the program refuses to give you what you want, it may at least give you a smile with with one of its smart aleck replies. For example, in trying to get out of the dungeon you may try to grab the ring that hangs on the wall. The computer will respond, THIS IS NOT A MERRY-GO-ROUND.

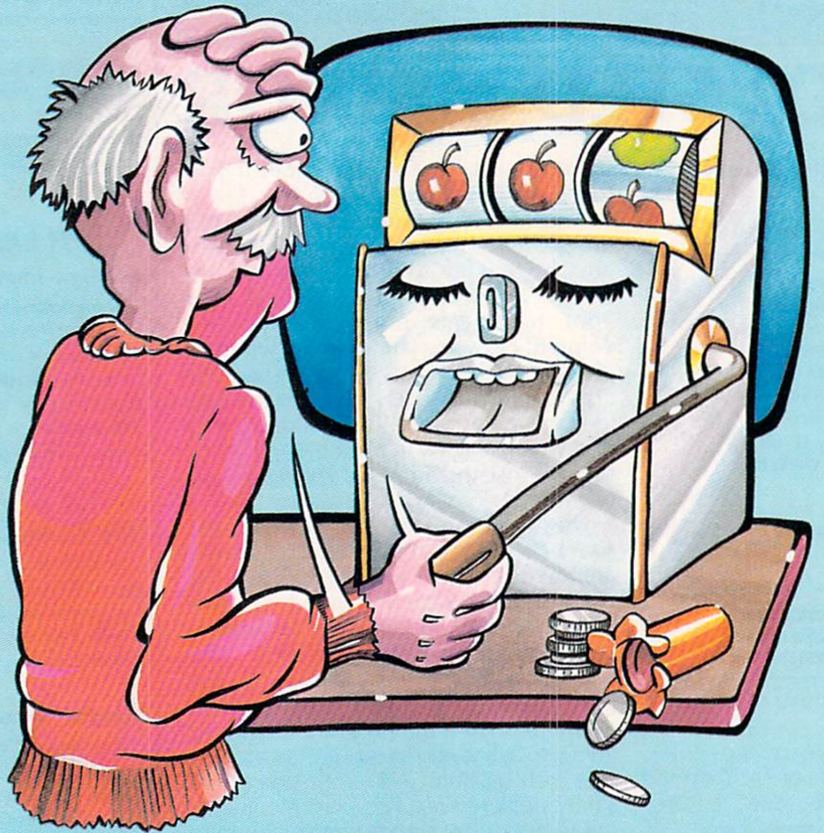
As you travel deeper into the medieval countryside, you find games within the game. A good one is the Sir Local Yokel jousting event, an arcade-type game within your adventure quest.

At this stage in gaming history, it is hard to think of an adventure that offers much more than *The Coveted Mirror* does. It should become a classic. HCM



Send in Your Photos and Anecdotes

Do you have a favorite photograph (color or black and white) featuring an unusual application of your Home Computer? Would you like to share your unusual or amusing anecdotes relevant to Home Computing? *Home Computer Magazine* will pay $25 for items it publishes. Material chosen will be subject to the same copyright treatment as "Letters to the Editor" as set forth on the Masthead page. No submissions can be returned. Send anecdotes and copies of photos to: Potpourri Editor, *Home Computer Magazine*, 1500 Valley River Drive, Suite 250, Eugene, Oregon 97401.



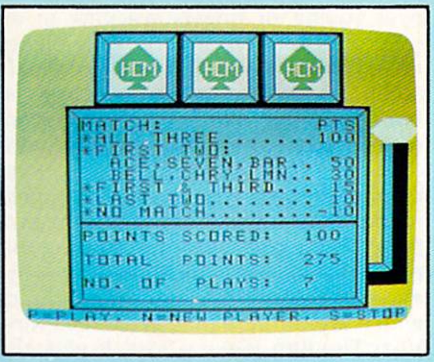
BASIC

Slots

by Bob Stoffers

GAMEWARE

BUFFET



It's your first day in Las Vegas, and you're a bit intimidated by the hubbub of an honest-to-goodness, big-time gambling casino. Deciding that the slot machines look tame enough for a grandmother like yourself, you timidly put a nickel into one of them. The woman playing next to you suddenly gets very angry and turns toward you. In almost one motion she pushes you aside, pulls down your machine's arm herself, reaches into the bag on her arm, and hands you a nickel.

Shaking your head and clucking a little over this real-life exhibition of the gambling frenzy you've only read about, you move on—to another row entirely—just to be on the safe side.

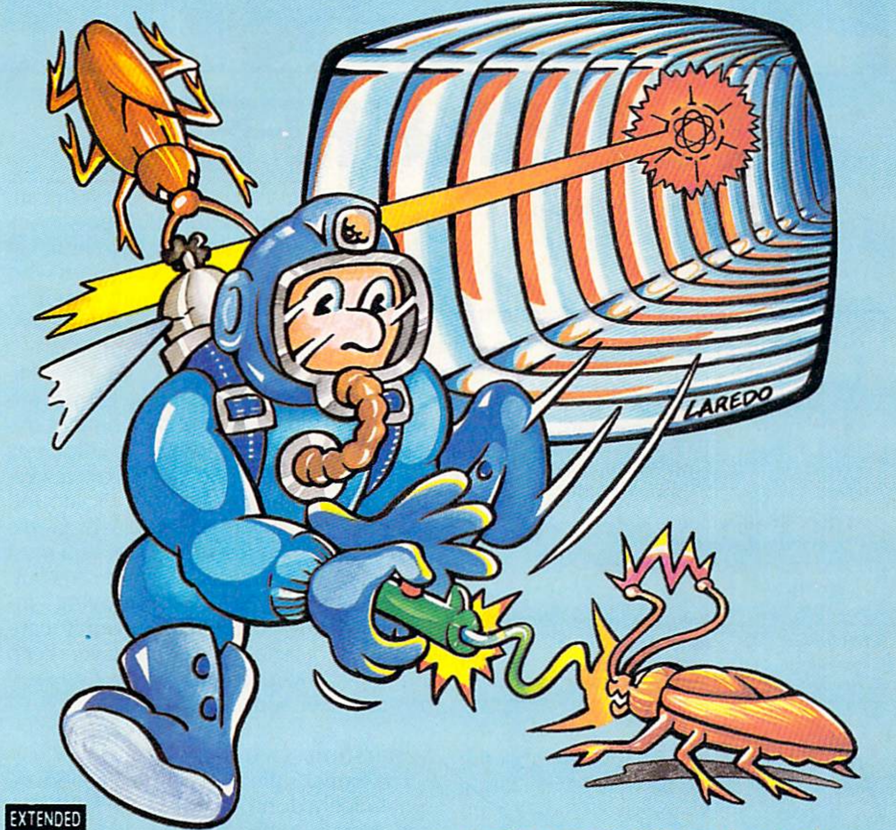
The machines in this row are different. They look so modern, like TVs with typewriter keyboards. A nice young man tells you to keep your nickels. He types the word RUN and presses [ENTER] for you. Some writing on the screen explains that you get points if the symbols in the machine's three windows match: 100 if all three windows match; 50 if the machine's first two windows show aces, 7's or bars; and 30 if they show cherries, lemons, or bells. You get 15 points if the first and the third windows match, and 10 points if the last two match. If each window shows a different symbol, you lose 10 points.

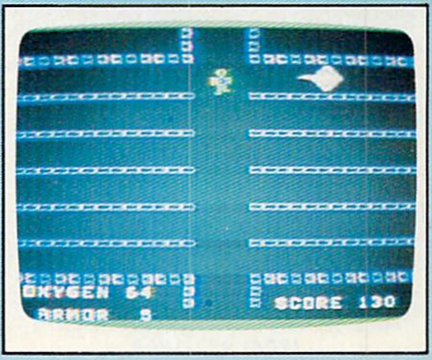
You worry that you won't remember everything you've read, but the next display is reassuring. At the front of the slot machine is a summary of it all—both the scoring system and the instructions for play, new player, and stop. You press P—it's so easy, even a grandmother can do it. The slot machine's display shows how many points you've won with each play, the total points you've accumulated, and how many plays it's taken you to get them.

There doesn't seem to be any system or temperament to it. You can't feel when the machine is "ready" to deliver a jackpot as the lady in the next row believes she can with her machines. It's always different and unpredictable: You might get 100 points with one press of P twice in a row, or you might end up with a mere 10 points for 10 plays. When you do hit the jackpot, you feel amply rewarded—not with cascading nickels, it's true, but with dazzling lights and an appropriate fanfare.

You know the grandchildren would love it too, but you're not about to bring them to a gambling hall. On the way back to the motel you resolve to talk your daughter into buying one of these home computers so you can key in this game for the children.

Continued on p. 190





EXTENDED BASIC

Meltdown

by Steve Langguth

To be awakened at three o'clock in the morning by a loudly ringing telephone is never a pleasant experience. And when you are the chief trouble-shooter for the world's first breeder reactor, you can be fairly sure that the news is going to be bad. This time, the voice on the other end of the line sounds even more rattled than usual. All you can make out is "bugs in the reactor," which is "approaching meltdown," and that you are the only person who can save the world from "total destruction."

Never one to pass up an interesting challenge, you race to the reactor site. Once there, you find that things are indeed as bad as you were told. Somehow bugs (cockroaches to be precise) have gotten into the reactor core and have fouled things up royally, so that at this very moment the reactor is running out of control. You must don your radiation-resistant armor, take your trusty "radiation neutralizer" in hand, and descend into the reactor core to clean up the subatomic particles which are floating about. Along the way you might even encounter some of those radioactive roaches which you must, of course, exterminate.

Unfortunately, a number of factors work against you:

1) The reactor core has five levels, with level one closest to the surface and level five deepest underground. The deeper you go, the more particles you will find which must be neutralized, and the more "energetic" the particles will be.
2) At all levels you will encounter gamma particles. These are a problem only when two of them come into your general vicinity at the same time. When that happens, they instantly become visible, fire off deadly gamma rays, and then disintegrate into nothingness. Luckily, they make distinctive sounds before they fire their rays, so you have a few seconds to get out of the way.
3) On levels two through five, particles of anti-matter will occasionally block your way. These are a problem only if they come into contact with normal matter (like you!). If that happens, the matter and the anti-matter tend to cancel each other out in a very large and violent explosion.
4) On levels three through five you will encounter the "nasty" roaches themselves. These little creatures seem to thrive on all that deadly radiation. In fact they have become so "energized" that it takes two hits from your neutralizer to exterminate them. (The first hit only makes them mad—and faster!)
5) Your supply of oxygen and radiation-resistant armor is limited. Spare oxygen tanks can be found along the way on all levels, but once the armor is gone, there is no more to be found.
6) And if all that were not enough, remember that the particles that you will meet in the tunnels of the reactor core are subatomic and therefore unpredictable. Our normal physical laws don't apply here, so be careful.

If you can successfully work your way through all five levels, you will be able to get back to bed. If you are not successful, it doesn't really matter because you will not have a bed to go home to anyway!

The Game

Meltdown is an arcade-style game for one player. After you load and run the program, you are given the choice of either joystick or keyboard to control the hero and the neutralizer. After you make that choice, the playing area will be drawn and the game will begin. Using the arrow keys or the joystick, move the hero character up or down through the central tunnel connecting the top and bottom of the screen. Fire your neutralizer into the corridors by pushing the joystick left or right or by using the left or right arrow key.

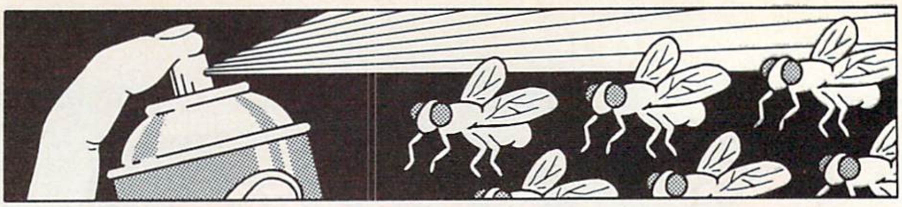
Neutralizing a particle is worth ten points, and allowing one to escape deducts five points from your score. Hitting a roach once gives you five points, changes the roach's color to white, and increases its speed. Hitting the same roach a second time counts for twenty points and neutralizes that roach.

Allowing a particle, roach, or gamma ray to touch you will cost you one of your five suits of armor. If you lose all five suits, the game is over. You begin with one hundred units of oxygen that get used up gradually as the game progresses. When a spare tank appears, touching it will transfer more oxygen to your supply. (Of course you have to stay in contact long enough for the transfer to be made!) If you should be unlucky enough to run out of oxygen, the game ends.

Anti-matter particles will occasionally block your path. Avoid them at all costs because touching one destroys everything—including you and the reactor.

There are five levels of play. As you descend deeper into the reactor, the number of particles you must face increases, and each particle moves faster than those on the level above. Only if you have the skill to survive all five levels will the reactor and the world be saved. Best of luck!

Continued on p. 110



FLY SNUFFER

A Review by Judy Sanoian
HCM Staff

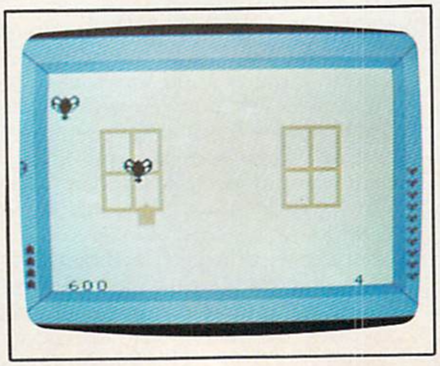
Name:	Fly Snuffer
Program Type:	Pest Control
Author:	Larry Lewis
Machine:	TI-99/4A
Distributor:	Futura Software P.O. Box 5581 Fort Worth, TX 76108
Price:	$19.95, cassette

System Requirements:
Cassette Recorder
Extended BASIC Cartridge
Joysticks

	Poor	Fair	Good	Excellent
Performance	—	—		
Engrossment	—	—	—	
Documentation	—	—	—	

Anyone old enough to remember the game *Pong* knows that fancy graphics and complex scenarios are not necessary for an entertaining video game. *Pong*, a precursor of our modern cursor entertainment, featured a moving dot which players batted back and forth between simple vertical lines. *Fly Snuffer*, an offering from Futura Software, is a game formed in the *Pong* mold, although its graphics are not nearly so primitive. Your main goal in *Fly Snuffer* is to exterminate a horde of flies with a deadly aerosol spray. It is really just another variation on the hit-dot-with-stick theme, and yet, like *Pong*, it is surprisingly engrossing.

The opening screen of *Fly Snuffer* displays a room with two windows in the center. One by one, the flies enter from all sides of the screen. Using your joystick, you stalk the metamorphosed maggots, pressing your fire button to release the toxic gas. Kill ten flies and you win a round. But flies are not the only insects to invade your abode. There are also large orange cockroaches that scurry across the floor, and menacing bumble bees. To kill the roach, you must smash him with your can. The bee succumbs to your poisonous spray, but don't be slow in killing him. He is one of the long-awaited killer bees who have finally arrived from South America. If you don't vaporize him, you will fall prey to his deadly stinger.



The flies behave much like their real-life counterparts. Some are fast, some are slow, some buzz around the window trying to get out. Even though the little pests are quite harmless, you cannot take a live-and-let-live attitude. If you sit around marveling at this tiny member of God's creation, you will lose the game. You must gas ten of them before the time runs out. And don't go around spraying indiscriminately. Even if you don't care about our deteriorating ozone layer, keep in mind that you lose the game if you run out of insect spray.

Supersonic Cyber-flies

Fly Snuffer's graphics and animation are quite nicely done. The flies are fat, black and glossy with flapping wings. The cockroaches and bees also sport appropriate features—moving legs and menacing stingers—and the spray can emits a realistic puff of DDT. *Fly Snuffer's* sound effects are also well done: The flit of the flies' wings, the *pfft* of the aerosol can, and the crunch of the squashed roach are all quite realistic. At first I was disappointed that there was no buzzing fly noise, but on second thought realized that a constant cyber-fly buzz would be as annoying as the real thing.

The end of my first game of *Fly Snuffer* left me with a cramped right hand—always the sign of an absorbing joystick game. But even though it was exciting, it was also frustrating. Maybe I am exceptionally slow on the spray button, but I could not—try as I might—get past level three (of six levels). The flies on levels four and beyond are simply too fast to catch. Also the roach is too quick to smash unless you happen to be right in its vicinity when it appears. Perhaps staying low on the screen (where the roach crawls) should be part of your game strategy. All I know is that once I got to level three, I found myself darting in all directions, futilely spraying at the speeding flies while the roaches came and went as they pleased. And after 30 seconds in a room full of dive-bombing flies, I felt more inclined to wave the white flag than make use of the Black one.

The bottom line in any game review (or at least the bottom paragraph) is whether or not the game is any fun. In the case of *Fly Snuffer*, I must say it is. Playing to win is challenging and yet possible, up to a point. . . and perhaps with practice you can even reach the highest level. The game loads easily and performs well if you follow its excellent documentation. (The instructions are among the most clear and complete I have ever seen.) *Fly Snuffer* is neither a visual masterpiece nor an exciting mental challenge. Like *Pong*, it is a good game to play when you want to give your cerebral side a rest. So if simple joystick action is your cup of tea, I'd recommend you give *Fly Snuffer* a try.

HCM

***Jumpman Jr*. . . from p. 67**

the scenario is exciting and varied enough to keep you at it until your joystick hand cramps in protest.

Space Chase Scenario

Although there are twelve levels of play and eight degrees of difficulty, your goal in *Jumpman Jr.* remains a simple one. As apprentice to Jupiter's secret agent, you must drive out the invading Alienators who have planted bombs throughout your substation. Using your joystick, you move up and down a network of ladders, ropes and elevators, working to defuse the bombs and at the same time, dodge the enemy bullets, fires and other dangers that threaten your existence. Your most effective means of movement (and your most thrilling) is the death-defying leap. By pressing your fire button and moving your joystick in the desired direction, you can execute a long, graceful jump that would put a gazelle to shame.

As soon as you complete your mission, the game moves on to the next level, where fresh obstacles await. It may take several tries just to figure out *how* to avoid or outsmart the flames, moving walls and electrocution traps that threaten you. Unfortunately, you are sent back to level 1 each time you lose. I would have appreciated a level selection option so I could skip right to the screen I wanted without having to work my way back through the preceding levels every time.

I may be a lightweight, but I also would have liked to see explanations of the various foes and obstacles (and the means of beating them) in the game's documentation. The instructions are otherwise complete, with ample descriptions of loading, movement and scoring. They describe *Jumpman Jr.* as a game of variety and suspense. I would add quick response, fast action, and superior graphics and sound to that description. I would also add a word of warning to the competition: *Donkey Kong*, watch out.

HCM

PLEASE DON'T FORGET TO RETURN THE QUESTIONNAIRE ON THE FRONT BIND-IN CARD.

Flak Attack . . . from p. 72

IBM PCjr

```
1180 LP=0:C=0:PLAYER=PLAYER-1:IF PLAYER=
     0 THEN GOTO 1220 ELSE FOR X=0 TO 10
     :HO(X)=0:NEXT :GOTO 570
1190 REM ********************
1200 REM OPTION TO PLAY AGAIN
1210 REM ********************
1220 CLS:LOCATE 1,1:PRINT "NICE GOING. Y
     OU SCORED ";SCORE;" POINTS"
1230 LOCATE 3,1:PRINT "WOULD YOU LIKE TO
      PLAY AGAIN (Y/N)?"
1240 A$=INKEY$:IF A$<>"N" AND A$<>"Y" TH
     EN GOTO 1240
1250 IF A$="Y" THEN SCORE=0:GOTO 220
1260 REM ***************
1270 REM END OF THE GAME
1280 REM ***************
1290 END
```

HCM

Flak Attack . . . from p. 61

Flak Attack(C-64) Explanation of the Program

Line nos.	
100-160	Program header.
170	Clear screen; POKE background header.
180-340	Load Auto Sprite routine.
350-360	Initialize sound and BASIC memory.
370-550	Initialize sprites and load sprite definitions
560-970	Display instructions and menu.
980	Input game level.
990-1070	Display game screen.
1080	Initialization.
1100	Clear motion bytes.
1110	Execute change interrupt vector machine language routine.
1120	Begin game.
1120-1320	Main program stream.
4000-4070	Initialize planes and set in motion.
4100-4270	Perform control key functions of move left, move right, and fire.
4500-4720	Plane fire routine.
5000-5020	Routine for destroyed plane.
8000-8010	Get character from keyboard buffer routine.
8600-8610	Sound routine.
9000-9030	End of game routine.

COMMODORE 64

```
100 REM  ***************
110 REM  * FLAK ATTACK *
120 REM  ***************
130 REM  BY MARK MOSELEY AND THE HCM ST
    AFF
140 REM  HOME COMPUTER MAGAZINE
150 REM  VERSION 4.1.1
160 REM  C64 BASIC
170 PRINT "{SHIFT CLR}":POKE 53281,1
180 FOR L=50880 TO 51116:READ A:POKE L,
    A:NEXT
190 DATA 169,255,45,0,198,240,16,169,0,
    141,0,198,162,21,189,0
200 DATA 197,157,0,198,202,208,247,162,
    1,169,1,141,80,197,173,80
210 DATA 197,45,0,197,240,3,76,243,198,
    232,232,14,80,197,208,238
220 DATA 76,49,234
230 DATA 169,0,29,0,197,208,3,76,97,199
    ,169,128,61
240 DATA 0,197,240,48,254,0,198,208,40,
    222,255,207,76,144,199,80
250 DATA 197,45,16,208,208,12,173,16,20
    8,13,80,197,141,16,208,76
260 DATA 43,199,173,16,208,77,80,197,14
    1,16,208,189,0,197,157,0
270 DATA 198,76,97,199,222,0,198,208,40
    ,254,255,207,208,29,173,80
280 DATA 197,45,16,208,208,12,173,16,20
    8,13,80,197,141,16,208,76
290 DATA 91,199,173,16,208,77,80,197,14
    1,16,208,189,0,197,157,0
300 DATA 198,169,0,232,29,0,197,208,3,7
    6,140,199,169,128,61,0
310 DATA 197,240,11,254,0,198,208,20,22
    2,255,207,76,134,199,222,0
320 DATA 198,208,9,254,255,207,189,0,19
    7,157,0,198,202,76,233,198
330 DATA 169,255,221,255,207,240,3,76,4
    3,199,173,80,197,76,17,199
340 DATA 120,169,192,141,20,3,169,198,1
    41,21,3,88,96
350 S=54272:FOR L=STOS+24:POKE L,0:POKE
    S+5,0:POKES+6,249:POKES+24,15
360 POKE52,62:POKE56,62
370 FOR I=16000 TO 16000+(6*64)-1
380 READ A:IF A>255 THEN 410
390 POKEI,A
400 NEXT:GOTO 420
410 READA:FORI2=ITOI+A-1:POKEI2,0:NEXT:
    I=I2-1:GOTO400
420 A=2040
430 FOR I=251 TO 255:POKEA,I:A=A+1:NEXT
440 POKE53287,0:POKE53288,0:POKE53289,1
    1
450 POKE 53290,0:POKE 53291,2:POKE 5326
    4,0
460 DATA 444,12,40,164,0,20,3,0,43,8,0,
    4,82,0,170,160,0,17,90,0,162,64,0
470 DATA 21,82,0,164,217,0,18,80,0,444,
    22
480 DATA 444,9,7,224,0,7,224,0,7,224,0,
    7,224,0,7,224,0,15,240,0,31,248,0
490 DATA 63,252,0,127,254,0,255,255,0,2
    55,255,0,255,255,0,444,19
500 DATA 444,12,0,128,0,0,192,0,0,224,0
    ,0,255,254,0,255,255,0,127,252,444,
    34
510 DATA 444,12,0,0,1,0,0,3,0,0,7,0,127
    ,255,0,255,255,0,63,254,444,34
520 DATA 444,12,1,128,0,1,128,0,7,224,0
    ,5,160,0,5,160,0,1,128,0,3,192,0
530 DATA 7,224,0,444,28
540 DATA 444,9,0,0,128,0,16,4,0,8,136,0
    ,4,144,0,2,160,0,1,64,0,46,186,0,1,
    128
```

COMMODORE 64

```
550 DATA 0,2,160,0,4,144,0,8,136,0,16,4,0,0,128,444,16
560 PRINT TAB(7) "{3 CRSR DOWN}{CTRL BLK}PRESS I FOR INSTRUCTIONS"
570 PRINT TAB(9) "PRESS N FOR NONE"
580 GOSUB 8000:IF K=78 THEN GOTO 920
590 IF K<>73 THEN 580
600 PRINT "{SHIFT CLR}{23 CRSR DOWN}"
610 PRINT TAB(9)"* FLAK ATTACK *"
620 PRINT ""
630 PRINT TAB(5)"THE OBJECT OF FLAK ATTACK"
640 PRINT TAB(5)"IS TO DESTROY AS MANY PLANES"
650 PRINT TAB(5)"AS POSSIBLE.  TO FIRE YOUR"
660 PRINT TAB(5)"MISSLE, PRESS E.  TO MOVE"
670 PRINT TAB(5)"LEFT, PRESS S, AND FOR  RIGHT"
680 PRINT TAB(5)"PRESS D"
690 PRINT "":PRINT
700 PRINT TAB(12)"* PLANES *":PRINT
710 PRINT TAB(5)"THE PLANE FIRES A NEWLY—"
720 PRINT TAB(5)"DEVELOPED BOMB. THE PLANE"
730 PRINT TAB(5)"MAY COME FROM LEFT OR RIGHT."
740 PRINT TAB(5)"WARNING! IT HAS A RADAR THAT"
750 PRINT TAB(5)"REMEMBERS YOUR LAST FIRING"
760 PRINT TAB(5)"POSITION AND TRIES TO GET YOU"
770 PRINT TAB(5)"THERE, SO BETTER MOVE AROUND"
780 PRINT:PRINT:PRINT
790 PRINT TAB(5)"PRESS ANY KEY TO CONTINUE"
800 GOSUB 8000:PRINT"{SHIFT CLR}{24 CRSR DOWN}"
810 PRINT TAB(11)"* BARRIER *":PRINT
820 PRINT TAB(5)"THE BOMBS CAN'T PENETRATE"
830 PRINT TAB(5)"THE BARRIER, BUT THE BARRIER"
840 PRINT TAB(5)"CAN SUSTAIN ONLY 5 DIRECT"
850 PRINT TAB(5)"HITS.  YOU CAN NOT FIRE WHEN BE—"
860 PRINT TAB(5)"HIND IT."
870 PRINT:PRINT:PRINT TAB(11)"* SCORING  *":PRINT
880 PRINT TAB(5)"PLANES ARE SCORED ACCORDING"
890 PRINT TAB(5)"TO HEIGHT: 5 FOR LOWEST, 20"
900 PRINT TAB(5)"FOR HIGHEST.":PRINT:PRINT:PRINT:PRINT:PRINT
910 PRINT TAB(5)"PRESS ANY KEY TO CONTINUE":GOSUB 8000
920 PRINT"{SHIFT CLR}{24 CRSR DOWN}"
930 PRINT TAB(5)"INPUT LEVEL OF DIFFICULTY:":PRINT:PRINT
940 PRINT TAB(5)"(1) PRO":PRINT
950 PRINT TAB(5)"(2) INTERMEDIATE":PRINT
960 PRINT TAB(5)"(3) NOVICE":PRINT
970 PRINT TAB(5)"(4) BEGINNER":PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
980 GOSUB 8000:IF K<49 OR K>52 THEN 980
990 GL=K-48:PRINT"{SHIFT CLR}"
1000 FOR X=55296 TO 56295:POKE X,0:NEXT
1010 FOR X=0TO24:POKE 1024+(X*40),160:POKE1063+(X*40),160
1020 POKE 1025+(X*40),160:POKE 1062+(X*40),160
1030 POKE 55296+(X*40),1:POKE 55335+(X*40),1
1040 POKE 55297+(X*40),1:POKE 55334+(X*40),1:NEXT
1050 FOR I=1904TO2023:POKE I,160:POKE54272+I,5:NEXT
1060 POKE 53248,172:POKE 53249,211:POKE53269,1
1070 FOR L=1801TO1805:POKEL,226:POKE54272+L,11:NEXT
1080 HA=0:AC=0:HP=0:HT(1)=0:HT(2)=0
1090 PRINT "{HOME}{CRSR DOWN}{CRSR RIGHT}{CTRL BLK}SCORE = ";HP
1100 FOR L=50432 TO 50449:POKEL,0:NEXT
1110 SYS 51104
1120 CC=PEEK(53278):FT=0:GOSUB 4000
1130 A=PEEK(197):GOSUB 4210
1140 C=PEEK(53279)AND2:IF C=0 THEN1130
1150 FOR L=1TO500*GL:NEXT:CC=PEEK(53279)
1160 GOSUB 4500
1170 IF HT(1)= 0 THEN1200
1180 HT(2)=HT(2)+1:IF HT(2)>2 THEN GOTO 9000
1190 GOSUB 5000
1200 CC=PEEK(53279)AND2:IF CC<>0 THEN1120
1210 A=PEEK(197):GOSUB 4100
1220 IF A<>14 THEN1160
1230 IF PEEK(53255)<150 THEN1260
1240 GOSUB 4500:IF HT(1)=0 THEN1260
1250 HT(2)=HT(2)+1:IF HT(2)>2 THEN 9000
1260 X=PEEK(53251):X1=PEEK(53255)
1270 C=PEEK(53278)AND2:IF C<>0 THEN 1310
1280 IF X1>X THEN1230
1290 CC=PEEK(53279)AND2:IF CC<>0 THEN1120
1300 POKE 50432,PEEK(50432)AND247:POKE 53269,PEEK(53269)AND247:GOTO1160
1310 POKE2041,255:POKE 53288,2:POKE 53269,3:HF=2:LF=163:SD=1000:GOSUB 8600
1320 POKE 53269,1:HP=HP+(5-LZ)*5:PRINT"{HOME}{CRSR DOWN}{CRSR RIGHT}SCORE = ";HP: GOTO1120
4000 TP=PEEK(53248):XP=PEEK(53264):SF=0:POKE 53269,1:POKE 53288,0
4010 POKE 53264,PEEK(53264)OR2:POKE 53250,200
4020 RD=INT(RND(0)*4)+1:LZ=RD:LV=RD*16+82:RD=INT(RND(0)*2)+1
4030 POKE53251,LV:POKE53253,LV:POKE 53269,PEEK(53269)OR2
4040 IF RD=1 THEN POKE 2041,252:POKE 50435,GL:GOTO 4060
4050 POKE 2041,253:POKE 50435,256-GL
4060 POKE 50432,2
4070 NU=PEEK(53279):POKE50688,255:RETURN
4100 IF A<>14 THEN 4210
4110 Z1=PEEK(53248)
4120 IF Z1<150 OR Z1>200 THEN 4160
```

Continued on p. 80

Flak Attack

COMMODORE 64

```
4130 Z1=INT((Z1-16)/8)
4140 Z2=PEEK(Z1+1784)
4150 IF Z2<>32 THEN A=1:RETURN
4160 POKE53254,PEEK(53248):POKE53255,PEE
     K(53249):POKE 53269,PEEK(53269)OR8
4170 N=PEEK(53264)AND1:IF N<>0 THEN POKE
      53264,PEEK(53264)OR8:GOTO 4190
4180 POKE 53264,PEEK(53264)AND247
4190 POKE50439,0:POKE50440,255:HF=16:LF=
     195:SD=25:GOSUB 8600
4200 POKE 50432,PEEK(50432)OR8:POKE 5068
     8,255:RETURN
4210 LP=PEEK(53264)AND1:IF (A<>13) THEN
     4250
4220 IF LP=0 AND PEEK(53248)<48 THEN 425
     0
4230 POKE 50433,255:POKE 50434,0:POKE504
     32,PEEK(50432)OR1:POKE 50688,255
4240 FOR ZX=1TO102:NEXT:POKE50432,PEEK(5
     0432)AND254:RETURN
4250 IF (A<>18) OR((PEEK(53264)AND1)=1 A
     ND PEEK(53248)>45) THEN RETURN
4260 POKE 50433,1:POKE50434,0:POKE50432,
     PEEK(50432)OR1:POKE 50688,255
4270 FOR ZX=1 TO 102:NEXT:POKE 50432,PEE
     K(50432)AND254:RETURN
4500 HT(1)=0:SX=INT(RND(0)*(GL*3))+1:IF
     SX=1 THEN 4520
4510 GOTO 4720
4520 SX=PEEK(53250)
4530 IF SX+8<TP AND SX+21>TP+16 THEN GOT
     O 4550
4540 SF=SF+1:IF SF>4 THEN 4720
4550 P1=PEEK(53250):P2=PEEK(53264)AND2:P
     3=PEEK(53251):GOTO4560
4560 KP=P1:IF P2<>0 THEN P1=255+P1
4570 IF RD=1 THEN P1=P1+6*(5-GL)
4580 P1=INT((P1-16)/8):P3=INT((P3-46)/8)
4590 PX=(1024+P1+40*P3):PZ=PX
4600 HF=16:LF=195:SD=20:GOSUB 8600
4610 CC=PEEK(53279)AND2:IF CC<>0 THEN112
     0
4620 POKE PX,32:SP=PEEK(PX+40):IF SP<>32
       THEN 4640
4630 POKE PX+40,93:PX=PX+40:GOTO4620
4640 IF SP<>226 THEN 4670
4650 FORLS=PZTO1903 STEP 40:POKE LS,32:N
     EXT:SD=50:HF=16:LF=195:GOSUB 8600
4660 CC=PEEK(53279):HT(1)=0:RETURN
4670 FORLS=PZTO1903 STEP 40:POKE LS,32:N
     EXT
4680 CC=PEEK(53279)AND1:IF CC<>0 THEN 47
     00
4690 POKEPX,32:HT(1)=0:RETURN
4700 HT(1)=1:POKE53269,3:POKE53287,2:POK
     E2040,250:SD=1000:HF=2:LF=163:GOSUB
     8600
4710 IF HT<3 THEN POKE 2040,251:POKE 532
     87,0
4720 RETURN
5000 POKE 2040,251
5010 OS=PEEK(53279)AND2:IF OS=0 THEN 501
     0
5020 RETURN
8000 A$="":GET A$:IF A$="" THEN 8000
8010 K=ASC(A$):RETURN
8600 POKE S+1,HF:POKES,LF:POKES+4,129
8610 FOR ZL=1TOSD:NEXT:POKES+4,128:RETUR
     N
9000 PRINT "{HOME}{CRSRDOWN}{CRSRRIGHT}
     {CTRL BLK}SCORE = ";HP
9010 POKE 53269,0
9020 A$="":GET A$:IF A$<>"" THEN 9020
9030 END
```

HCM

Flak Attack . . . from p. 62

990-1110	Move the plane and make decision about when to fire at the ground.
1120-1270	Fire laser.
1280-1300	Subroutine to delete the airplane.
1310-1510	Plane is hit by your cannon and crashes.
1520-1560	Check cannon ball's position and delete.
1570-1620	Check cannon ball for collisions and update screen.
1630-1780	Player's cannon destroyed.
1790-1870	End of the game; option to play again.
1880	End of the game.
1890-1930	Subroutine to display text without scrolling the screen.
1940-1950	Graphics character data.

TI-99/4A

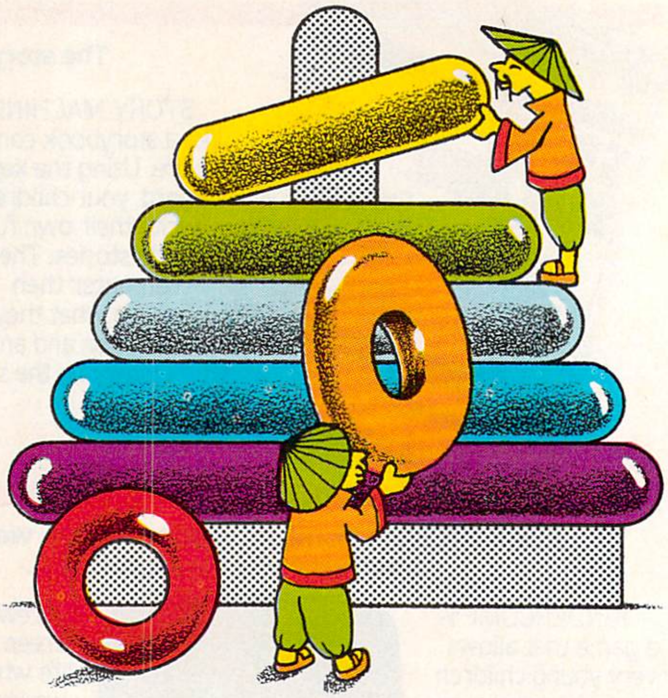
```
100 REM *************
110 REM * FLAK ATTACK *
120 REM *************
130 REM BY MARK MOSELEY AND HCM STAFF
```

```
140 REM  HOME COMPUTER MAGAZINE
150 REM  VERSION 4.1.1
160 REM  TI BASIC
170 REM
180 CALL SCREEN(8)
190 CALL CLEAR
200 RANDOMIZE
210 PRINT "          FLAK ATTACK"::::::::
::::
220 FOR X=1 TO 200
230 NEXT X
240 PLAYER=3
250 CALL CLEAR
260 TANK=16
270 PL=0
280 SCORE=0
290 CAN=0
300 PRINT "LEVEL OF DIFFICULTY":::
310 PRINT "   1. BEGINNER":::"   2. NOVIC
E":::"   3. PRO"::::"   4. END THE GA
ME"::::::"YOUR CHOICE"::
320 CALL KEY(0,K,S)
330 IF S=0 THEN 320
340 IF (K<49)+(K>52)THEN 320
350 IF K<52 THEN 370
360 STOP
370 CALL CLEAR
380 DIF=K-48
390 IF FP=1 THEN 450
400 FOR X=1 TO 9
410 READ A,AS
420 CALL CHAR(A,A$)
430 NEXT X
440 FP=1
450 CALL COLOR(9,5,1)
460 CALL COLOR(10,13,1)
470 CALL COLOR(11,7,1)
480 CALL COLOR(12,4,1)
490 CALL HCHAR(22,1,120,64)
500 CALL HCHAR(20,12,120,8)
510 CALL HCHAR(21,16,96)
520 A$="SCORE:"
530 PRP=3
540 GOSUB 1890
550 IF PL=0 THEN 580
560 GOSUB 990
570 GOTO 590
580 GOSUB 870
590 IF CAN=0 THEN 620
600 GOSUB 1520
610 CALL SOUND(-1000,-7,10)
620 CALL KEY(0,K,S)
630 IF S=0 THEN 550
640 IF K=83 THEN 670
650 IF K=68 THEN 730
660 IF (K=69)*(CAN=0)THEN 790 ELSE 550
670 IF TANK=3 THEN 550
680 CALL SOUND(-10,660,0)
690 CALL HCHAR(21,TANK,32)
700 CALL HCHAR(21,TANK-1,96)
710 TANK=TANK-1
720 GOTO 550
730 IF TANK=30 THEN 550
740 CALL SOUND(-10,660,0)
750 CALL HCHAR(21,TANK,32)
760 CALL HCHAR(21,TANK+1,96)
770 TANK=TANK+1
780 GOTO 550
790 CALL GCHAR(20,TANK,CH)
800 IF CH=120 THEN 550
810 CAN=1
820 LP=TANK
830 CLOC=19
840 CALL SOUND(-40,110,5,220,5,-7,0)
850 CALL HCHAR(CLOC,LP,116)
860 GOTO 550
870 PL=1
880 DIR=INT(RND*3)-1
890 IF DIR=0 THEN 880
900 IF DIR=1 THEN 940
910 PS=104
920 PP=30
930 GOTO 960
940 PS=105
950 PP=3
960 ALT=INT(RND*8)*2+3
970 CALL HCHAR(ALT,PP,PS)
980 RETURN
990 IF (DIR=1)*(PP=30)THEN 1280
1000 IF (DIR=-1)*(PP=3)THEN 1280
1010 CALL GCHAR(ALT,PP+DIR,CH)
1020 IF CH=32 THEN 1060
1030 CALL HCHAR(ALT,PP,32)
1040 PP=PP+DIR
1050 GOTO 1310
1060 CALL HCHAR(ALT,PP,32)
1070 CALL HCHAR(ALT,PP+DIR,PS)
1080 PP=PP+DIR
1090 IF (PP-3<LP)*(PP+3>LP)*INT(RND*15/D
IF)=1 THEN 1120
1100 IF INT(RND*40/DIF)=1 THEN 1120
1110 RETURN
1120 CALL SOUND(-500,110,5,220,5,330,3,-
4,0)
1130 CALL GCHAR(20,PP,CI)
```

Continued on p. 88

Challenging the Tower of Hanoi

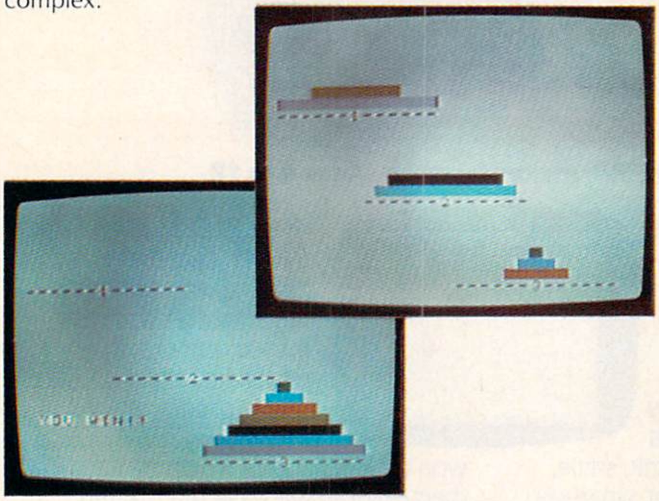
We at HCM have slightly modified Professor Holl's Tower of Hanoi *(in the February, 1983, issue of 99'er Home Computer Magazine) so as to add more rings—and thus challenge—to the puzzle. We invite our readers to further modify this program by adding a routine that will automatically solve the puzzle and display the moves as they occur. If you develop a good routine, send it to us. We'll screen your results and publish a version for each machine.*



You are in an ancient temple at the center of the earth, where three diamond needles bear eighty golden rings of graduated sizes. At the beginning of time the rings were all on one needle; but now the temple monks are transferring the rings, one at a time from needle to needle, never setting a ring on a smaller ring. When they have moved all eighty rings to one of the other two needles, the world will end . . .

Professor Holl solved the mystery of adapting this ancient Buddhist puzzle for the TI-99/4A using four rings, and now the Zen masters at *HCM* have discovered algorithms to expand the game to seven rings and convert it for the IBM, Apple IIe, Commodore 64, and VIC-20 disciples. We've also included a new expanded version for TI enthusiasts.

You may have seen something like the *Tower of Hanoi* in a baby's playpen—a set of multi-colored rings stacked on a wooden post. A child is supposed to advance from teething on the rings to trying to pile them up in size order, but your challenge in this deceptively simple brain-teaser is a bit more complex.



On the initial screen of this program, the seven colored "rings" (actually, they look more like short bars) appear on the left side of the screen, arranged in descending size order. The object of the game is to transfer the whole stack, one ring at a time, to another pile (the pegs are left out for simplicity's sake). You have three piles to work with, and one basic rule—you may never lay a larger ring on top of a smaller one.

Stacks

Now that you know how to play the game, let's think about the logic the computer uses to keep track of the stacks of rings. No matter what method is used, two conditions must be met. First, the rings must be kept in order within each stack. And rings may only be added to or removed from the top of the stack.

These conditions are easily met by representing a stack as a string of numeric characters. Each character represents one ring, and its numeric value corresponds to its size. When the game begins, the first peg (containing all seven rings) is "1234567", and the other two pegs are represented by the empty string (""). Keep in mind that the leftmost character is the top ring of a stack, and the numeric value of each character corresponds to the size of the ring.

Whenever we want to add or remove a ring, we must add or remove the leftmost character of the string. If, at the beginning of the game, you move the smallest ring from peg 1 to 2, peg 1 will become "234567" and peg 2 will be "1". Peg 3 remains empty (""). (The only exception to this is the TI version, which uses a floating point number instead of a string of numbers. The first stack is represented as 1.234567. The integer of the value represents the size of the top ring.)

The three stacks are stored as elements in an array called PEG$( ) with subscripts (1), (2), and (3). (This is PEG( ) on the TI version.) When you play the game you enter the numbers (1 to 3) of the peg each ring is to be taken from, and the peg that it is to be transferred to. These numbers are then used as the subscripts of the PEG$ array, to select the proper peg.

So far we have been talking about the part of the program you never see as you play the game. The computer's internal manipulation of the rings is actually quite simple and applies to all versions of the program. But to play the game, you need to see something on the screen. Because each computer has different methods and capabilities for displaying graphics, we'll give you a brief overview for each machine.

In IBM's medium resolution graphics mode you have control of 64,000 points of light arranged in a grid 320 wide by 200 high. Keeping track of all those pixels is easier than you might expect. The LINE statement makes it easy to draw the different colored bars in the screen locations needed for the *Tower of Hanoi*.

In its simplest form, the LINE statement draws a line between any two points on the screen. You specify the points by naming the X and Y coordinates corresponding to the rows and columns. For example, LINE (0,0)–(319,199) would draw a diagonal line across the screen from the upper left to the lower right corner.

The LINE statement can also be used to draw a box and color it in. When we use LINE in this way, we use the same sets of coordinates, but this time they determine the opposite corners of the box instead of the endpoints of a line segment.

The first statement in line 210 draws the top ring of the tower in its initial position on the first peg. The upper left corner of this bar is 36,79, and the lower right is 48,85. The number 1 is the code for the color blue. BF indicates two things: B tells the computer that a box (rather than a line) should be drawn, and F says the box should be filled in with color.

While you're playing the game, the LINE statement in program line 410 draws the rings in any of the legal positions on the three pegs. Line 390 erases the old ring. You must indicate which peg the ring is being moved FROM as well as the SIZE and the vertical position of the ring, specified by TP(FROM). Line 410 uses TOO to draw the bar at the destination peg.

TOWER OF HANOI (IBM PCjr)
Explanation of the Program

Line nos.	
100-160	Program header.
170-190	Turn off the KEY line, and set up the graphics screen mode as medium resolution with sixteen available colors.
200	Dimension arrays.
210-260	Display playing screen.
270-360	Input player's move.
370	Check for legal moves.
380-470	Move ring.
480-520	End of the game option to play again.

Continued on p. 86

The TI version of *Tower of Hanoi* uses the HCHAR command to display the rings. On the 99/4A, colors are assigned to character sets in groups of eight characters. Two characters are chosen from each set to make two rings. One of the characters is defined as a solid block of the foreground color. The other ring character is a blank and so displays the background color. The PC( ) array keeps track of which character is used for each of the seven rings. The PEG( ) array keeps track of which rings are on the three pegs. It does this with a number which starts out as 1.234567. When a ring is taken off of a peg, the integer that represents that ring is subtracted from PEG(FROM), the floating point number that represents that peg. The ring integer is then stored in the SIZE variable. PEG(FROM) is multiplied by 10, moving the decimal point one place to the right. It then has the value 2.34567, and SIZE becomes equal to 1. The peg that the ring is moving to, PEG(TOO), is multiplied by .1, and SIZE is added to that value. This way you can keep track of which rings are on which pegs, and the order of the stacks. The TOP( ) array keeps track of the top of each peg. This is necessary so that the computer will know where to place the peg.

TOWER OF HANOI (TI-99/4A)
Explanation of the Program

Line Nos.	
100-170	Program header.
180-320	Initialize arrays, characters, and color.
330-440	Display playing screen.
450-560	Input the peg to move from and the peg to move to.
570-670	Move ring.
680-740	End of game message.
750	Game data for arrays and color.
760	End.

Continued on p. 87

Because the Commodore computers have no built-in "line-drawing" function, we wrote our own subroutine, which begins at line 1000. To draw the bar for a particular ring, the subroutine needs three pieces of information: the size of the ring being moved, the peg number it will be moved to, and the ROW of the stack where it will be displayed.

First the size (SZ) of the ring to be moved is determined in line 490 by getting the VALue of the leftmost member of the from string. Then, in line 570, this character is added to the left of the string selected by the T variable. Line 510 also uses SZ to determine the values needed to erase and draw the rings. Notice that the color code variable (CC) is used to choose the color and the size of the ring in line 250.

There is a slight idiosyncracy in Commodore BASIC which creates the need for line 580. When you use the STR$ function on a number to get its equivalent string value, a blank is added to the left side. If you then take the leftmost character (using the LEFT$ function) you'll get a blank instead of the 1 through 7 you expect. Line 580 strips these added spaces one at a time until the LEFT character is not a space.

Tower of Hanoi (C-64 and VIC-20)
Explanation of the Program

Line Nos.	
100-160	Program header.
170-210	Dimension arrays, initialize variables.
220-310	Display initial game position.
320-470	Get input and check to see if it is legal.
480-580	Move rings and change arrays.
590-600	Check to see if game is over.
610-660	Play again routine.
1000-1040	Subroutine to draw rings.

VIC-20 continued on p. 88, C-64 continued on p. 87

For the Apple IIe version of the *Tower of Hanoi,* we selected the low resolution graphics mode because it is well-suited for putting bars of color on the screen. We used the low resolution command HLIN, which displays a row of block characters between specified columns at a specified row.

In line 410 we derive the size of the ring to be erased and moved. We use HLIN to erase the old bar (in line 430) by setting the color to black, the same as the background color. Then the F variable tells the computer which peg will have its top ring erased. The proper row is stored in the TP array. SZ specifies the width of the bar.

We use HLIN again in line 450 to display the bar in its new position. This time we use T to specify the peg. Because each ring is a different size, we use the SZ variable to derive different colors for each of the rings. Otherwise, line 450 uses HLIN in the same way as line 430 did.

TOWER OF HANOI (APPLE)
Explanation of the Program

Line nos.	
100-170	Program header.
180-290	Display playing screen.
300-370	Input player's move.
380-400	Check for legal moves.
410-500	Move the ring from one peg to another.
510	Error routine.
520-550	End of the game option to play again.
560	End of the program.

Continued on p. 86

Tower of Hanoi . . . from p. 85

APPLE II Series

```
100 REM    ******************
110 REM    * TOWER OF HANOI *
120 REM    ******************
130 REM    BY PROF. HOLL AND HCM STAFF
140 REM    HOME COMPUTER MAGAZINE
150 REM    VERSION 4.1.1
160 REM    APPLE II SERIES APPLESOFT
170 REM
180 DIM PEG$(3),TP(3)
190 GR : HOME : PRINT "PRESS 1 FOR THE
    FIRST TOWER": PRINT "PRESS 2 FOR TH
    E SECOND TOWER": PRINT "PRESS 3 FOR
     THE THIRD TOWER"
200 COLOR= 13
210 VLIN 30,34 AT 7: VLIN 30,34 AT 19:
    VLIN 30,34 AT 21: VLIN 30,34 AT 31:
     VLIN 30,34 AT 33: VLIN 30,34 AT 35
220 COLOR= 6: HLIN 7,7 AT 1
230 COLOR= 7: HLIN 6,8 AT 2
240 COLOR= 8: HLIN 5,9 AT 3
250 COLOR= 9: HLIN 4,10 AT 4
260 COLOR= 10: HLIN 3,11 AT 5
270 COLOR= 11: HLIN 2,12 AT 6
280 COLOR= 12: HLIN 1,13 AT 7
290 PEG$(1) = "1234567":PEG$(2) = "":PE
    G$(3) = "":TP(1) = 1:TP(2) = 16:TP(
    3) = 24
300 VTAB 24: HTAB 1: PRINT "FROM = ";
310 GET F: IF F < 1 OR F > 3 THEN  GOTO
     310
320 HTAB 8: PRINT F;
330 PRINT  CHR$ (7);
340 HTAB 20: VTAB 24: PRINT "TO = ";
350 GET T: IF T < 1 OR T > 3 OR T = F T
    HEN  GOTO 350
360 HTAB 25: PRINT T;
370 PRINT  CHR$ (7);
380 IF PEG$(F) = "" THEN  GOTO 510
390 IF PEG$(T) = "" THEN  GOTO 410
400 IF  ASC (PEG$(F)) >  ASC (PEG$(T))
    THEN  GOTO 510
410 SZ =  VAL ( LEFT$ (PEG$(F),1))
420 TP(T) = TP(T) - 1
430 COLOR= 0: HLIN (F - 1) * 13 + 8 - S
    Z,(F - 1) * 13 + 6 + SZ AT TP(F)
440 TP(F) = TP(F) + 1
450 COLOR= 5 + SZ: HLIN (T - 1) * 13 +
    8 - SZ,(T - 1) * 13 + 6 + SZ AT TP(
    T)
460 IF  LEN (PEG$(F)) = 1 THEN PEG$(F)
    = "": GOTO 480
470 PEG$(F) =  RIGHT$ (PEG$(F), LEN (PE
    G$(F)) - 1)
480 PEG$(T) =  STR$ (SZ) + PEG$(T)
490 IF PEG$(2) = "1234567" OR PEG$(3) =
     "1234567" THEN  GOTO 520
500 GOTO 300
510 PRINT  CHR$ (7); CHR$ (7); CHR$ (7)
    ; CHR$ (7);: GOTO 300
520 HOME : PRINT "CONGRATULATIONS, YOU
    HAVE WON THE GAME!": PRINT : PRINT
    "WOULD YOU LIKE TO PLAY AGAIN?"
530 PRINT "(Y/N)";: GET A$: IF A$ = "N"
     THEN  GOTO 560
540 IF A$ = "Y" THEN  CLEAR : GOTO 100
550 GOTO 520
560 TEXT : HOME : END
```

HCM

Tower of Hanoi . . . from p. 85

IBM PCjr

```
100 REM ******************
110 REM * TOWER OF HANOI *
120 REM ******************
130 REM BY PROF. HOLL AND HCM STAFF
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.1.1
160 REM IBM PCjr
170 KEY OFF
180 CLEAR,,,32768!
190 CLS:SCREEN 5
200 DIM PEG$(3),TP(3)
210 LINE (36,79)-(48,85),1,BF
220 LINE (30,86)-(54,92),2,BF:LINE (24,
    93)-(60,99),3,BF:LINE (18,100)-(66,
    106),4,BF:LINE (12,107)-(72,113),5,
    BF:LINE (6,114)-(78,120),6,BF
230 LINE (0,121)-(84,127),7,BF
240 PEG$(1)="1234567":PEG$(2)="":PEG$(3
    )="":TP(1)=79:TP(2)=128:TP(3)=128
250 LOCATE 18,6:PRINT "1              2
     3"
260 LOCATE 20,1:PRINT "1 - FOR THE FIRS
    T PEG":PRINT "2 - FOR THE SECOND PE
    G":PRINT "3 - FOR THE THIRD PEG"
270 LOCATE 24,1:PRINT "FROM = ";
280 A$=INKEY$:IF A$="" THEN GOTO 280
290 IF A$<"1" OR A$>"3" THEN GOTO 280
300 FROM=VAL(A$):IF PEG$(FROM)="" THEN
    PRINT CHR$(7);:GOTO 280
310 LOCATE 24,7:PRINT A$;
320 LOCATE 24,20:PRINT "TO = ";
330 A$=INKEY$:IF A$="" THEN GOTO 330
340 IF A$<"1" OR A$>"3" THEN GOTO 330
350 TOO=VAL(A$):IF TOO=FROM THEN GOTO 3
    30
360 LOCATE 24,25:PRINT A$;
370 IF PEG$(TOO)<>"" THEN IF ASC(PEG$(F
    ROM))>ASC(PEG$(TOO)) THEN PRINT CHR
    $(7);:LOCATE 24,1:PRINT "
                             ";:GOTO 2
    70
380 SIZE=VAL(LEFT$(PEG$(FROM),1)):TP(TO
    O)=TP(TOO)-7
390 LINE ((FROM-1)*120+42-(6*SIZE),TP(F
    ROM))-((FROM-1)*120+42+(6*SIZE),TP(
    FROM)+6),0,BF
400 TP(FROM)=TP(FROM)+7
410 LINE ((TOO-1)*120+42-(6*SIZE),TP(TO
    O))-((TOO-1)*120+42+(6*SIZE),TP(TOO
    )+6),SIZE,BF
420 REM
430 PEG$(FROM)=RIGHT$(PEG$(FROM),LEN(PE
    G$(FROM))-1)
440 PEG$(TOO)=RIGHT$(STR$(SIZE),1)+PEG$
    (TOO)
450 IF PEG$(2)="1234567" OR PEG$(3)="12
    34567" THEN GOTO 480
460 LOCATE 24,1:PRINT "
                             ";
470 GOTO 270
480 CLS:LOCATE 15,1:PRINT "CONGRADULATI
    ONS ON COMPLETING THE TOWER.":PRINT
     "THE PREISTS OF HANOI ARE PROUD."
490 PRINT "PLAY AGAIN (Y/N)?"
500 A$=INKEY$:IF A$="" THEN GOTO 500
510 IF A$="N" THEN CLS:PRINT "BYE BYE":
    END
520 IF A$="Y" THEN CLS:GOTO 210
```

HCM

Tower of Hanoi . . . from p. 85

TI-99/4A

```
100 REM  *****************
110 REM  * TOWER OF HANOI *
120 REM  *****************
130 REM  BY PROF. HOLL AND THE HCM STAF
F
140 REM  HOME COMPUTER MAGAZINE
150 REM  VERSION 4.1.1
160 REM  TI BASIC
170 REM
180 CALL SCREEN(16)
190 CALL CLEAR
200 DIM PEG(3),TOP(3),PC(7)
210 CALL COLOR(7,1,1)
220 CALL COLOR(8,2,2)
230 READ PEG(1),PEG(2),PEG(3),TOP(1),TO
P(2),TOP(3)
240 FOR COL=9 TO 12
250 READ A,B
260 CALL COLOR(COL,A,B)
270 NEXT COL
280 READ PC(1),PC(2),PC(3),PC(4),PC(5),
PC(6),PC(7)
290 FOR X=96 TO 120 STEP 8
300 CALL CHAR(X,"00")
310 CALL CHAR(X+1,"FFFFFFFFFFFFFFFF")
320 NEXT X
330 CALL HCHAR(8,3,45,13)
340 CALL HCHAR(16,10,45,13)
350 CALL HCHAR(24,17,45,13)
360 FOR X=1 TO 8
370 CALL COLOR(X,2,1)
380 NEXT X
390 CALL HCHAR(8,9,49)
400 CALL HCHAR(16,16,50)
410 CALL HCHAR(24,23,51)
420 FOR X=1 TO 7
430 CALL HCHAR(X,10-X,PC(X),X*2-1)
440 NEXT X
450 CALL KEY(3,FROM,STATUS)
460 IF STATUS=0 THEN 450
470 CALL KEY(3,DUMMY,STATUS)
480 IF STATUS=-1 THEN 470
490 FROM=FROM-48
500 CALL SOUND(100,110,3)
510 CALL KEY(3,TOO,STATUS)
520 IF STATUS=0 THEN 510
530 CALL KEY(3,DUMMY,STATUS)
540 IF STATUS=-1 THEN 530
550 TOO=TOO-48
560 CALL SOUND(100,262,2)
570 IF (FROM<1)+(FROM>3)+(TOO>3)+(TOO<1
)THEN 450
580 IF (PEG(FROM)=0)+((PEG(TOO)<>0)*(PE
G(FROM)>PEG(TOO)))THEN 450
590 SIZE=INT(PEG(FROM))
600 TOP(TOO)=TOP(TOO)-1
610 CALL HCHAR(TOP(FROM),((FROM-1)*7+10
)-SIZE,32,SIZE*2-1)
620 TOP(FROM)=TOP(FROM)+1
630 CALL HCHAR(TOP(TOO),((TOO-1)*7+10)-
SIZE,PC(SIZE),SIZE*2-1)
640 PEG(FROM)=10*(PEG(FROM)-SIZE)
650 PEG(TOO)=.1*PEG(TOO)+SIZE
660 IF (PEG(2)=1.234567)+(PEG(3)=1.2345
67)THEN 680
670 GOTO 450
680 A$="YOU WIN!!"
690 FOR X=1 TO 9
700 CALL HCHAR(20,3+X,ASC(SEG$(A$,X,1))
)
```

TI-99/4A

```
710 NEXT X
720 CALL KEY(0,K,S)
730 IF S=0 THEN 720
740 GOTO 760
750 DATA 1.234567,0,0,1,16,24,6,4,12,10
,8,7,1,14,96,97,104,105,112,113,120
760 END
```

HCM

Tower of Hanoi . . . from p. 85

COMMODORE 64

```
100 REM *****************
110 REM * TOWER OF HANOI *
120 REM *****************
130 REM BY S.T. HOLL AND THE HCM STAFF
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.1.1
160 REM C64 BASIC
170 REM INITIALIZE
180 DIM PG$(3),TP(3):POKE 53281,12
190 PRINT "{SHIFT CLR}{CTRL BLK}"
200 PG$(1)="1234567":PG$(2)="":PG$(3)="
"
210 TP(1)=4:TP(2)=16:TP(3)=22
220 REM INITIAL DISPLAY
230 SZ=1:CS=7:CH=160
240 FOR RW=4 TO 10
250 CL=CS:CC=7-((SZ*2)-1)/2
260 GOSUB 1000
270 SZ=SZ+1:CS=CS-1
280 NEXT RW
290 SS=1024:SC=55296
300 POKE SS+293,49:POKE SS+546,50:POKE
SS+839,51
310 POKE SC+293,0:POKE SC+546,0:POKE SC
+839,0
320 REM ACCEPT INPUT
330 PRINT"{HOME}
                                "
340 PRINT"{CRSR DOWN}
                          ":PRINT"{HOME}
                     FROM";
350 GET F$: IF F$="" THEN 350
360 FR=ASC(F$)-48:IF FR<1 OR FR>3 THEN
350
370 PRINT FR
380 PRINT"{CRSR DOWN}                TO";
390 GET T$: IF T$="" THEN 390
400 T=ASC(T$)-48:IF T<1 OR T>3 THEN 390
410 PRINT T
420 IF (PG$(FR)="")THEN 460
430 IF PG$(T)="" THEN 490
440 IF ASC(PG$(FR))>ASC(PG$(T)) THEN 46
0
450 GOTO 490
460 PRINT "{HOME}      CAN'T DO THAT! TRY
AGAIN!"
470 FOR DE=1 TO 500:NEXT:GOTO 320
480 REM MOVE RINGS
490 SZ=VAL(LEFT$(PG$(FR),1))
500 TP(T)=TP(T)-1
510 C=((FR-1)*12)+1:CC=(7-SZ):CL=C+CC:C
H=32 :RW=TP(FR)
520 GOSUB 1000
530 IF LEN(PG$(FR))=1 THEN PG$(FR)="":G
OTO 550
540 PG$(FR)=RIGHT$(PG$(FR),LEN(PG$(FR))
-1)
```

Continued on p. 88

Tower of Hanoi . . . from p. 87

COMMODORE 64

```
550 TP(FR)=TP(FR)+1:C=(((T-1)*12)+1:CL=C+CC:CH=160:RW=TP(T)
560 GOSUB 1000
570 PG$(T)=STR$(SZ)+PG$(T)
580 IFLEFT$(PG$(T),1)=" "THENPG$(T)=RIGHT$(PG$(T),LEN(PG$(T))-1):GOTO 580
590 IF PG$(2)="1234567" OR PG$(3)="1234567" THEN 610
600 GOTO 320
610 PRINT "YOU HAVE WON THE GAME!"
620 PRINT "WOULD YOU LIKE TO PLAY AGAIN?"
630 PRINT "Y/N"
640 GET A$:IF A$="" THEN 640
650 IF A$="Y" THEN 190
660 PRINT "{SHIFT CLR}":END
1000 REM SUBROUTINE TO DRAW A ROW
1010 FOR RS=1 TO SZ*2-1
1020 POKE(55296+(RW*40)+CL),CC
1030 POKE (1024+(RW*40)+CL),CH
1040 CL=CL+1: NEXT RS :RETURN
```

HCM

Tower of Hanoi . . . from p. 85

VIC-20

```
100 REM ******************
110 REM * TOWER OF HANOI *
120 REM ******************
130 REM BY S.T. HOLL AND THE HCM STAFF
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.1.1
160 REM V20 BASIC
170 REM INITIALIZE
180 DIM PG$(3),TP(3):POKE 36879,251
190 PRINT "{SHIFT CLR}"
200 PG$(1)="1234567":PG$(2)="":PG$(3)=""
210 TP(1)=0:TP(2)=14:TP(3)=22
220 REM INITIAL DISPLAY
230 SZ=1:CS=6:CH=160
240 FOR RW=0 TO 6
250 CL=CS:CC=7-((SZ*2)-1)/2
260 GOSUB 1000
270 SZ=SZ+1:CS=CS-1
280 NEXT RW
290 SS=7680:SC=38400
300 POKE SS+55,49:POKESS+236,50:POKE SS+416,51
310 POKE SC+55,0:POKESC+236,0:POKE SC+416,0
320 REM ACCEPT INPUT
330 PRINT "{HOME}{16CRSRDOWN}":PRINT "{CRSRDOWN}"
340 PRINT "{HOME}{16CRSRDOWN} FROM";
350 GET F$: IF F$="" THEN 350
360 FR=ASC(F$)-48:IF FR<1 OR FR>3 THEN 350
370 PRINT FR
380 PRINT "{CRSRDOWN} TO";
390 GET T$: IF T$="" THEN 390
400 T=ASC(T$)-48:IF T<1 OR T>3 THEN 390
410 PRINT T
420 IF (PG$(FR)="")THEN 460
430 IF PG$(T)="" THEN 490
440 IF ASC(PG$(FR))>ASC(PG$(T)) THEN 460
450 GOTO 490
460 PRINT "{SHIFT CRSRUP}TRY AGAIN!":FOR DE=1 TO 500:NEXT
470 PRINT "{2SHIFT CRSRUP}":GOTO 320
480 REM MOVE RINGS
490 SZ=VAL(LEFT$(PG$(FR),1))
500 TP(T)=TP(T)-1
510 C=(FR-1)*4+1:CC=(7-SZ):CL=C+CC-1:CH=32 :RW=TP(FR)
520 GOSUB 1000
530 IF LEN(PG$(FR))=1 THEN PG$(FR)="":GOTO 550
540 PG$(FR)=RIGHT$(PG$(FR),LEN(PG$(FR))-1)
550 TP(FR)=TP(FR)+1:C=(T-1)*4+1:CL=C+CC-1:CH=160:RW=TP(T)
560 GOSUB 1000
570 PG$(T)=STR$(SZ)+PG$(T)
580 IFLEFT$(PG$(T),1)=" "THENPG$(T)=RIGHT$(PG$(T),LEN(PG$(T))-1):GOTO580
590 IF PG$(2)="1234567" OR PG$(3)="1234567" THEN 610
600 GOTO 320
610 PRINT "{SHIFT CLR}{2CRSRDOWN}YOU HAVE WON THE GAME!"
620 PRINT "WOULD YOU LIKE TO PLAY AGAIN?"
630 PRINT "Y/N"
640 GET A$:IF A$="" THEN 640
650 IF A$="Y" THEN 190
660 PRINT "{SHIFT CLR}":END
1000 REM SUBROUTINE TO DRAW A ROW
1010 FOR RS=1 TO SZ*2-1
1020 POKE(38400+(RW*22)+CL),CC
1030 POKE (7680+(RW*22)+CL),CH
1040 CL=CL+1: NEXT RS :RETURN
```

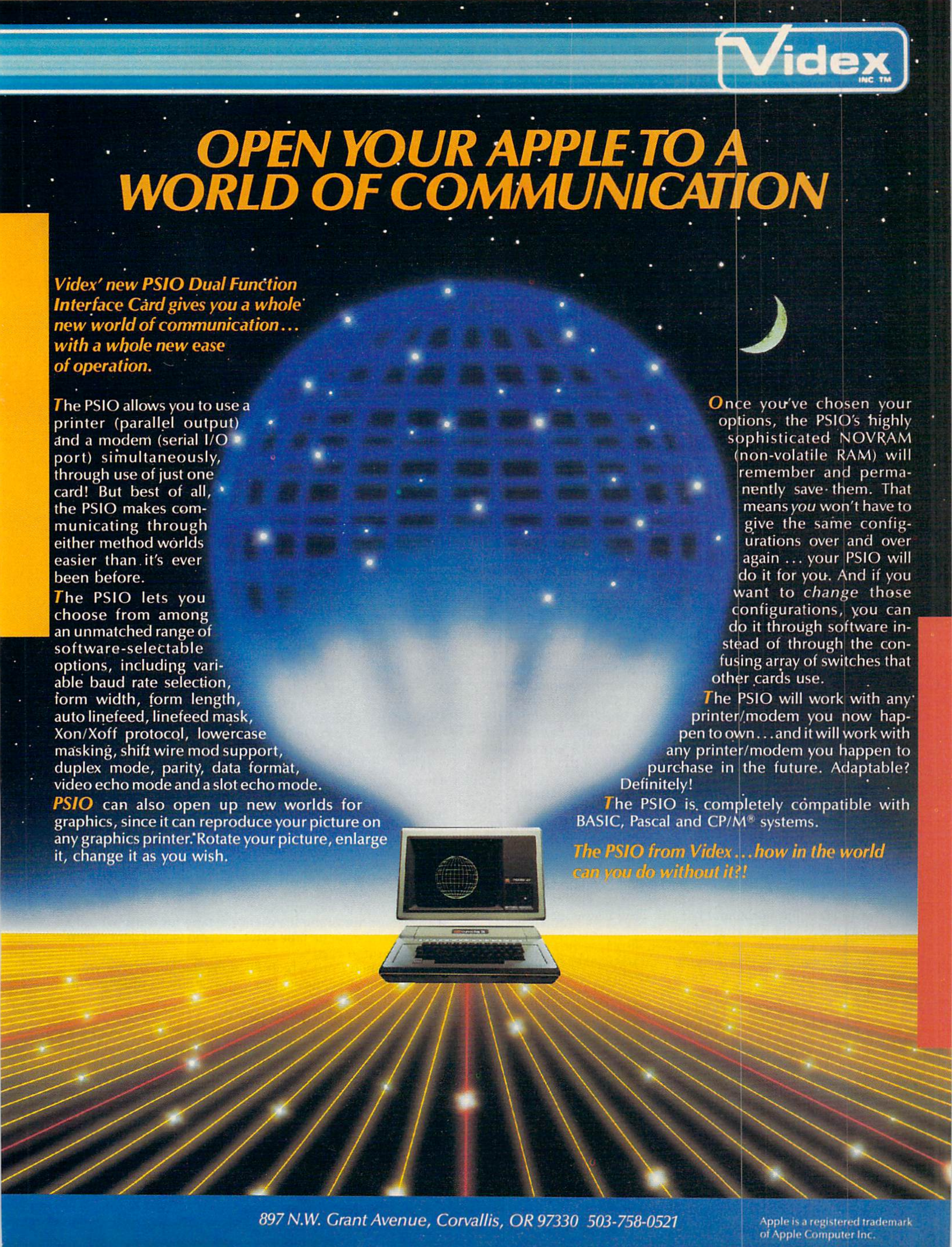
HCM

Flak Attack . . . from p. 81

TI-99/4A

```
1140 CALL GCHAR(20,TANK,CH)
1150 IF CI=120 THEN 1180
1160 Z=21
1170 GOTO 1190
1180 Z=20
1190 CALL VCHAR(ALT+1,PP,112,Z-ALT)
1200 CALL VCHAR(ALT+1,PP,32,Z-ALT)
1210 CALL SOUND(-1,44000,30)
1220 IF (PP=TANK)*(CH<>120)THEN 1640
1230 IF CI=120 THEN 1260
1240 CALL HCHAR(22,PP,114)
1250 RETURN
1260 CALL HCHAR(20,PP,114)
1270 RETURN
1280 CALL HCHAR(ALT,PP,32)
1290 PL=0
1300 RETURN
1310 CALL HCHAR(ALT,PP,32)
1320 CALL HCHAR(CLOC,LP,32)
1330 CALL HCHAR(ALT,PP,115)
1340 CALL SOUND(300,-8,0)
1350 CALL SOUND(400,-4,0)
1360 SCORE=SCORE+25-ALT
1370 FOR ALT=ALT TO 21
1380 CALL HCHAR(ALT,PP,113)
1390 CALL SOUND(-500,110,20,220,20,1000-(ALT*25),25,-8,0)
1400 CALL GCHAR(ALT+1,PP,CH)
1410 IF (CH=120)+(CH=96)+(CH=114)THEN 1440
1420 CALL HCHAR(ALT,PP,32)
1430 NEXT ALT
1440 CAN=0
1450 PL=0
1460 IF ALT>21 THEN 1480
1470 CALL HCHAR(ALT,PP,32)
1480 CALL HCHAR(21,TANK,96)
1490 A$=STR$(SCORE)
1500 GOSUB 1890
1510 RETURN
1520 IF CLOC>ALT THEN 1570
1530 IF CLOC=ALT THEN 1630
1540 CALL HCHAR(CLOC,LP,32)
1550 CAN=0
1560 RETURN
1570 CALL GCHAR(CLOC-2,LP,CH)
1580 IF CH=PS THEN 1310
1590 CALL HCHAR(CLOC,LP,32)
1600 CALL HCHAR(CLOC-2,LP,116)
1610 CLOC=CLOC-2
1620 RETURN
1630 IF LP=PP THEN 1310 ELSE 1590
1640 CALL HCHAR(21,TANK,115)
1650 FOR X=1 TO 5
1660 CALL SOUND(-500,110,10,220,10,RND*1000+300,30,-8,0)
1670 CALL HCHAR(21,TANK,96)
1680 CALL HCHAR(21,TANK,115)
1690 NEXT X
1700 CALL HCHAR(21,TANK,32)
1710 CALL HCHAR(ALT,PP,32)
1720 CALL HCHAR(CLOC,LP,32)
1730 PLAYER=PLAYER-1
1740 IF PLAYER=0 THEN 1790
1750 PL=0
1760 CAN=0
1770 TANK=16
1780 GOTO 490
1790 CALL CLEAR
1800 PRINT "NOT BAD, YOU SCORED";SCORE::"POINTS AGAINST THE ENEMY."
1810 PRINT ::::"WOULD YOU LIKE TO PLAY AGAIN"::"(Y/N)?"
1820 CALL KEY(0,K,S)
1830 IF S=0 THEN 1820
1840 IF (K<>89)*(K<>78)THEN 1820
1850 IF K=78 THEN 1880
1860 SCORE=0
1870 GOTO 190
1880 STOP
1890 FOR X=1 TO LEN(A$)
1900 CALL HCHAR(24,PRP+X,ASC(SEG$(A$,X,1)))
1910 NEXT X
1920 PRP=9
1930 RETURN
1940 DATA 96,00001C1C1C3E7F7F,116,0808081C1C14,104,01033FFF3C,105,80C0FCFF3C,112,0808080808080808
1950 DATA 113,F070383838381010,114,0081C243C3C281,115,915254A954242810,120,FFFFFFFFFFFFFFFF
```

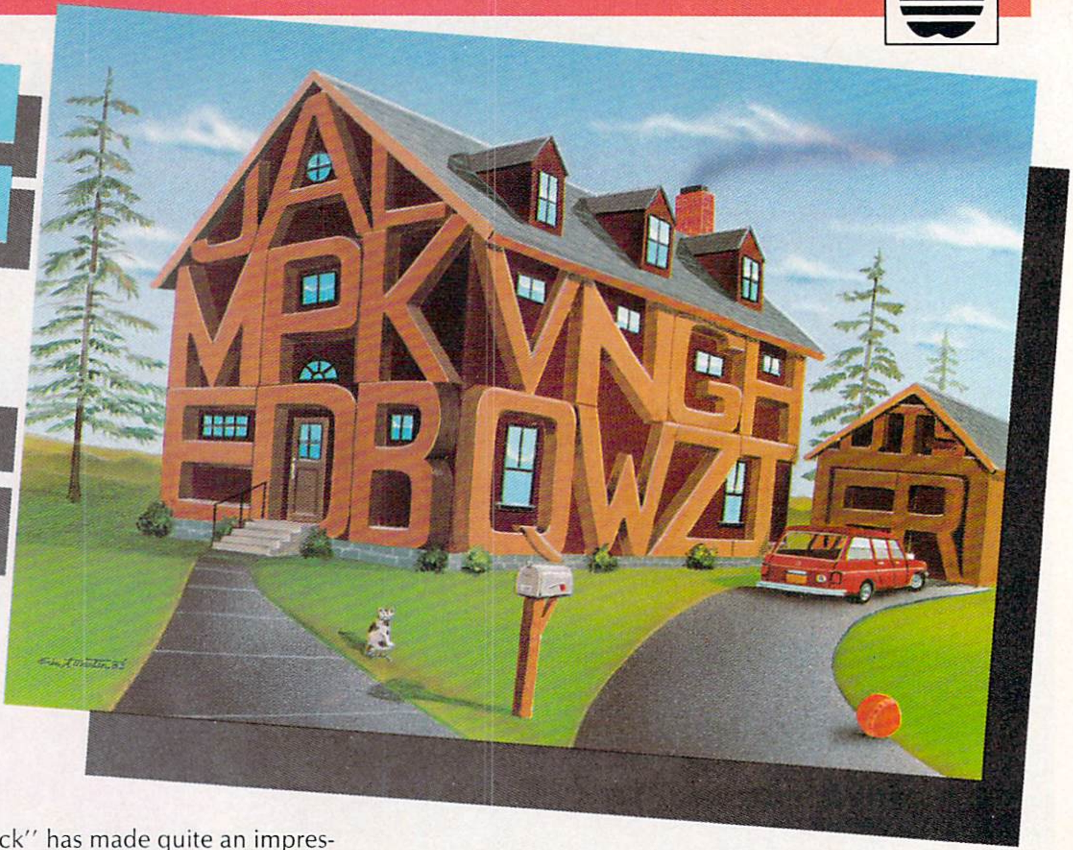
HCM


Videx INC TM
OPEN YOUR APPLE TO A WORLD OF COMMUNICATION
Videx' new PSIO Dual Function Interface Card gives you a whole new world of communication... with a whole new ease of operation.
The PSIO allows you to use a printer (parallel output) and a modem (serial I/O port) simultaneously, through use of just one card! But best of all, the PSIO makes communicating through either method worlds easier than it's ever been before.
The PSIO lets you choose from among an unmatched range of software-selectable options, including variable baud rate selection, form width, form length, auto linefeed, linefeed mask, Xon/Xoff protocol, lowercase masking, shift wire mod support, duplex mode, parity, data format, video echo mode and a slot echo mode.
PSIO can also open up new worlds for graphics, since it can reproduce your picture on any graphics printer. Rotate your picture, enlarge it, change it as you wish.
Once you've chosen your options, the PSIO's highly sophisticated NOVRAM (non-volatile RAM) will remember and permanently save them. That means you won't have to give the same configurations over and over again ... your PSIO will do it for you. And if you want to change those configurations, you can do it through software instead of through the confusing array of switches that other cards use.
The PSIO will work with any printer/modem you now happen to own...and it will work with any printer/modem you happen to purchase in the future. Adaptable? Definitely!
The PSIO is completely compatible with BASIC, Pascal and CP/M® systems.
The PSIO from Videx...how in the world can you do without it?!
897 N.W. Grant Avenue, Corvallis, OR 97330 503-758-0521
Apple is a registered trademark of Apple Computer Inc.

HOMEWORD BOUND

A Review of HomeWord

by Sharyn Lyon
Education Editor



This "new kid on the block" has made quite an impression on Mom, Dad, and the kids in the neighborhood. *HomeWord, The Personal Word Processor* is an extremely easy-to-use program designed with the inexperienced user in mind. This latest brainchild of Sierra On-Line's Ken Williams is a word processing system for occasional writers such as students (as young as nine years old), home business persons, newsletter writers/editors, and those who write things like an annual holiday letter. Thanks to *HomeWord*, even users who are not professionals can have sophisticated word processing techniques at their fingertips.

Name:	HomeWord, The Personal Word Processor
Author:	Ken Williams
Program Type:	Word Processor
Machine:	Apple II, II+, IIe, with versions in the making for the Commodore 64 and the IBM PCjr
Distributor:	Sierra On-Line Sierra On-Line Building Coarsegold, CA 93614 (209) 683-6858
Price:	$69.95 ($75 for the PCjr version)

	Poor	Fair	Good	Excellent
Performance				
Ease of Use				
Documentation				

Funny, You Don't Look Like a Word Processor

The first thing everyone notices about *HomeWord* is its unique screen display—an extraordinary 3-in-1 screen concept. The top three-quarters of the screen is the writing area, with room for 20 lines of type, 38 characters long. Before you count this sparse display as a disadvantage, wait to see how the rest of the screen is put to work. The bottom quarter is divided into two parts. The larger part is a kind of fuel gauge with receding horizontal bars to show how much memory and disk space you have left.

The smaller portion of the screen display is the Page Sketch or layout section. As you type, your page takes shape before your eyes (see Figure 1). This feature effectively compensates for the 38-column per line screen display which has plagued Apple word processing until now. Fast typists need not worry about this layout feature slowing them down. The page isn't actually sketched until the writer pauses (to reflect on what to write next, perhaps), and the reformating takes no time at all. In addition, this program displays upper- *and* lowercase type implementation—a feature that has not been too readily available on an Apple until recently.

More Than Just A Pretty Package

If you have never used a word processor before, you may find *HomeWord's* package of instructional goodies helpful in getting started. When you open the durable and attractive package, amid the usual advertising enclosures and business reply cards, you will find:

- *The HomeWord Story*—a user's manual;
- an audio instruction tape that will teach you how to use the system in half an hour;
- the transcript of the instructional tape for those who do not have a cassette tape player or who would rather learn from written text;
- a Reference Card which fits conveniently above your keyboard, just a quick glance away from the monitor and the keys; and
- a Disk Use and Care Checklist.

If you are inexperienced, you might want to look over the first few pages of *The HomeWord Story*. Here you will find the basics—from what a computer is, to how to use certain important keys. If you have already worked with a word processor, you will probably be able to start typing without consulting these in-package supports.

Just Like Typing, Only Better!

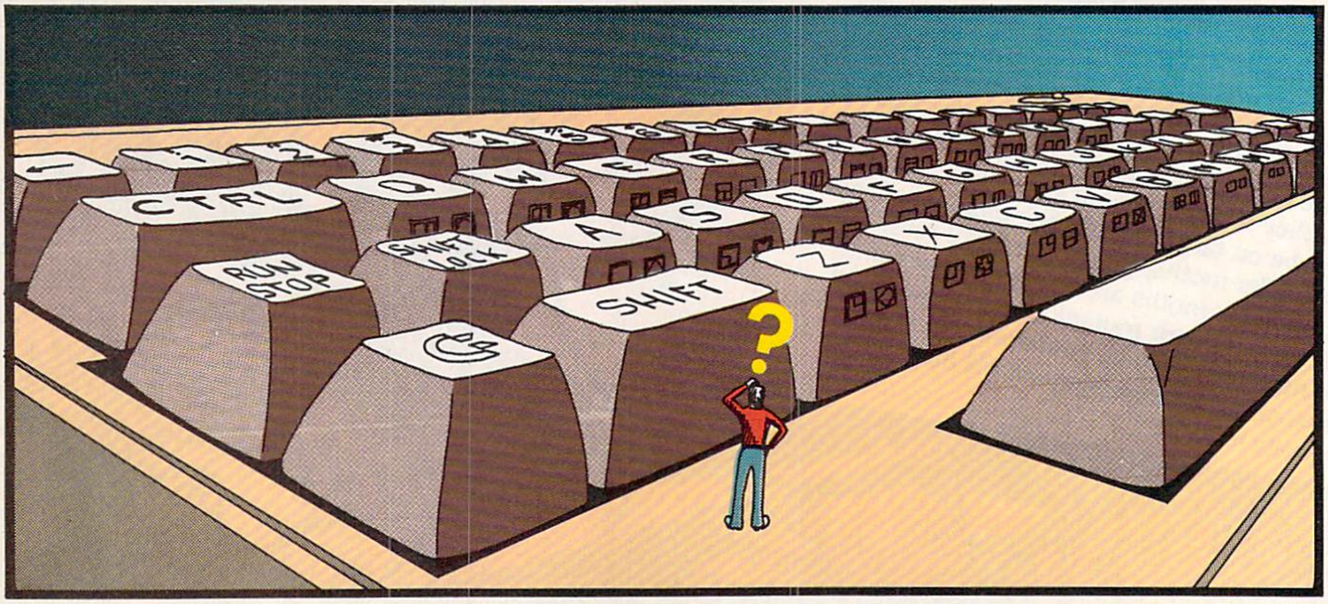
The closer a word processor comes to old-fashioned typewriter typing, the easier it is for most of us. *HomeWord* has many of traditional typing's good points. For example,

Continued on p. 96


for the
COMMODORE® 64
KILLER PILLER
ONLY
*9.99
SUGGESTED INTRODUCTORY RETAIL PRICE
Take careful aim at the caterpillar eating it's way through your orchard... and ... FIRE! Uh oh... looks like your hit was not placed too well because where there was once one caterpillar there are now two... each growing at an alarming rate as they hungrily devour your trees. Will your next hit be on target or will their offspring, those devious killer moths, get to you first? And, if you think the killer moths are tough, just wait until you encounter their mutant cousins! But don't get tempted to overuse your spray because as sure-as-shoot'n those beasties will build up an immunity to it.
100% Machine Code
7 Screen Levels
2 Skill Levels
*CASSETTE VERSION
Disk version $12.75
Also available for Vic 20® Cassette only at $9.99
Write or phone for catalog—Dealers invited to order, see your local dealer or order direct from us. Check, M.O., Visa and Master Card welcome! Add $2.00 for Shipping.
CASTLE SOFTWARE
P.O. Box 350
New Castle, DE 19720
(302) 429-8565

66 Keys to Graphics Success

A PRIMER FOR THE COMMODORE C-64 AND VIC-20



by Will Schick
Technical Editor

For a newcomer to the world of home computers, the prospect of learning how to use a home computer may seem overwhelming at first. Even the clearest instruction manuals are full of strange terms and jargon—computer specialists are notoriously insensitive to the plight of the poor beginner. This article is intended to help you get to know your new Commodore computer and to show off some of its best features. It is not a substitute for a careful reading of the *User's Guide*, but it will introduce you to most of the keys and some of the terms you will encounter again and again as you become a real computer enthusiast.

If you haven't already done so, go ahead and get your computer hooked up and READY to go. If you're wondering "READY for what?" you're asking the right question. The answer is simple—the computer is ready to follow your instructions.

Look at the keyboard for a moment. You will see most of the keys normally found on a typewriter in their regular places. You will use these to type in your program instructions. But there are also a number of special keys which can be used to give instant commands to the computer. To see how this works, find the [CTRL] (for ConTRoL) key on the left side of the keyboard near the Q key. Hold the [CTRL] key down and press the 5 key. Look at the screen now, and you will see that the flashing square, called the *cursor*, has turned purple. Simply by pressing two keys at the same time, you gave the computer an instruction to change the color of the cursor and all the type that will appear on the screen.

Pressing combinations of keys is one way to give the computer instructions. Another way is to type them in. Your computer will respond to instructions in the BASIC language, and you will soon discover that typing errors will not be overlooked. For the time being, if you make a mistake, simply type the line (be sure to include the line number) over again. The computer is picky, but also very patient about giving you another try.

One common BASIC instruction begins with the word POKE. Programmers use POKE in a wide variety of situations to fill a *memory location* with a *value*. For example, suppose we want a green screen to go with the purple letters we just selected. There is a five-digit number which represents the memory location that controls the color of the screen. There is also a value, or code number, for each of the available colors. Let's POKE the value for green into the memory location that controls screen color. Type in:

POKE 53281,5 *(Commodore 64)*
POKE 36879,61 *(VIC-20)*

Make sure that you typed the instruction correctly, and then hit the [RETURN] key (on the right side of the keyboard). The computer will pay no attention to the instruction until you *enter* it with the [RETURN] key.

Did it work? Instead of the green screen you wanted, you may be looking at the words SYNTAX ERROR or ILLEGAL QUANTITY ERROR. Don't worry; no harm has been done. This is just the computer's way of telling you that it didn't understand your command. Probably you made a slight mistake in typing it in—a misspelled word, a wrong number, or a missed comma. All you have to do is retype the line correctly and enter it. You will occasionally type in something that is pure gibberish to the computer. Every time you do, the computer will print an appropriate error message and wait patiently for you to try again.

If you're not crazy about a purple cursor on the green screen, you can try other colors by using the [CTRL] key with any of the number keys from 1 through 8. (The colors are printed on the front of these keys in abbreviated form.) Notice in particular that if you try [CTRL] [6], the cursor seems to disappear. This is because you've made it the same color as the screen.

To return the screen to its original color, type:

POKE 53281,6 *(Commodore 64)*
POKE 36879,27 *(VIC-20)*

Your *User's Guide* will show you more locations you can POKE and the values to be used.

Moving Around the Screen

When you were typing these instructions, did you notice how the cursor moved along, marking the place where the next letter or number would appear on the screen? You can move the cursor anywhere on the screen by using the two [CRSR] keys in the lower right corner of the keyboard.

Find these two keys, and notice that the one on the left is marked with up and down arrows, and the one on the right has arrows pointing from side to side. Hold the right [CRSR] key down and watch the cursor move rapidly to the right side of the screen and then jump to the beginning of the next row. To move the cursor to the left, hold down the [SHIFT] key and press the right [CRSR] key. The other [CRSR] key, used to move the cursor up and down the screen, works the same way. Use the left [CRSR] key by itself to move the cursor down, and the [SHIFT] and [CRSR] keys together to move the cursor up. If you move the cursor all the way to the top of your screen, it simply stops and flashes rapidly, but if you move it to the bottom, it will cause everything on the screen to scroll up and disappear. Practice moving the cursor around until you're ready to explore Commodore's excellent graphics.

Instant Graphics

Now for the fun part. Using only a few more of the Commodore's special keys, you can experiment with a wide variety of graphics patterns, textures and colors. First, find the [CLR/HOME] key in the top row of the keyboard, second in from the right. If you press this key by itself, it will send the cursor "home," that is, to the upper left corner of the screen. Pressing [CLR/HOME] while you hold the [SHIFT] key down will both clear the screen and send the cursor home.

Now hold [CTRL] while pressing 9. As we have already seen with colors, using the [CTRL] key with another key activates the function printed on the front of that key. In this case, we are activating [RVS ON] (reverse on), and the patterns of the characters and their backgrounds will be reversed as we type. ([RVS OFF] is immediately to the right of [RVS ON].)

Now hold down the SPACE bar, and you can easily fill an area with color, row by row. Fill the top half of the screen, and then switch to another color (using [CTRL] with any number, 1 through 8). Finish the background, then move the cursor back to the middle of the screen and try designing a house, a tree, a dog, all three, or any scene you can imagine. You can use the SPACE bar or any combination of letters and symbols, switching colors whenever you like. Make sure you don't move the cursor down below the bottom of the screen, or the top of your design will scroll up and disappear, never to be recovered.

Built-In Graphics Characters

Playing with colored blocks is fun for a while, but there is much more your computer can do for you. Over 100 ready-to-use graphics characters are available with just a few keystrokes. On the front of all of the letter keys (and some of the symbol keys) you can see pairs of graphics patterns—lines, edges, triangles, a heart, and so on.

To show the character on the right side of a key, hold down [SHIFT] while pressing that key. For instance, [SHIFT] [S] displays a heart. These keys will not "auto repeat" as the SPACE bar and [CRSR] keys will; so you must hit the key over and over to get the number of characters you want.

To display the left-hand character, use the [COMMODORE] key (lower left corner of the keyboard, marked by the Commodore logo) the same way you used the [SHIFT] key for the right-hand character.

Programming Graphics

Pressing those keys all the time is getting to be a lot of work. Why don't we have the computer do it for us? When we write a program, we are in effect telling the computer to press its own keys. And while we're typing in a short program, we'll see how easy it is to edit (modify) program instructions on the Commodore computers. Before we begin, press [SHIFT][CLR] to clear the screen and put the cursor in its starting place. Use [CTRL] and color numbers to pick a cursor and type color you can easily see on your monitor.

Now you're ready to type in a *program*, which is just an ordered collection of instructions. We'll start our first line with the number 10, which tells the computer that what follows is the first part of a program. All of your program lines will get line numbers that make the computer read and follow your instructions in the order you want. (Be sure to use number 0, not letter O.)

Let's make the first instruction of the program the CLEAR/HOME function. We will create different versions of the program as we go, but each time we will want to clear the screen and send the cursor home.

When you type in a Commodore program from our *HCM* listings, note that words inside hands call for pressing those keys rather than typing in the words and the hands themselves. So in the first line of our program,

```
10 PRINT "[SHIFT CLR]"
```

for example, we will type 10, the word PRINT, the quotation mark, and then press the [SHIFT] and the [CLR] keys together. Then we will close the line with the quotation mark. A heart will appear between the quotes. This double key press will ensure that every time you RUN the program it will start off in the right place, with a clean slate.

Although we will learn other ways of editing later on, at this point if you should make a mistake, hit [RETURN] and start the line over. This is a foolproof method for correcting any line.

Hit [RETURN] to enter line 10, and you'll find your cursor in position for typing the next line of your program:

```
20 FOR K=1 TO 440
```

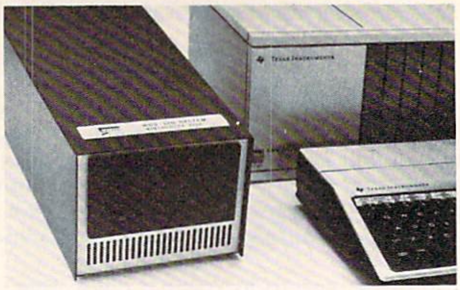
That was fairly easy and straightforward, but in the next line (line 30) be sure to remember to hit the keys inside the hands rather than type in the words. Don't forget the semicolon that appears outside the quotation mark:

```
30 PRINT "[CTRL RVSON][CTRL GRN][SHIFT M][CTRL BLK][SHIFT N]";
40 NEXT K
```

Line 30 does all the work here. A series of five keys selects REVERSE ON, selects two built-in characters, and colors them black and green. Lines 20 and 40 make this happen 440 times, all the way down the screen. In line 30, the computer will print graphics between the quotes, but these won't necessarily match the pattern on the keys, so be careful.

Screen Editing

If you haven't already mistyped something, you will soon enough. Being only human, even the best programmers make



plenty of errors. But you can correct them quickly using Commodore's editing features.

Suppose you mistyped line 20 so that it read 20 FER K=1 TO 440. In this case you could simply move the cursor over the E, replace it with an O, and then hit the [RETURN] key to re-enter the line.

But what if you typed 20 FOUR K=1 TO 440? Now you'll need to use the [INST/DEL] key in the upper right corner of your keyboard. Move the cursor to the R, the position after the U, which is the character we want to delete, and hit the [INST/DEL] key.

There is yet another type of common error. You may have typed 20 FR K=1 TO 440 and need to insert another character. This time, position the cursor over the R, hold down [SHIFT], hit [INST/DEL] to make a space, and then type in the missing O. Don't forget to re-enter the corrected line by pressing [RETURN].

Editing line 30 presents a small problem. Because the Commodore computers allow you to put cursor commands into PRINT instructions, they will not allow you to move the cursor to the left or right while you are in quote mode (following an open quote). If you try to edit between quotation marks, the computer thinks you are trying to put your cursor commands into the quotes. To get around this problem, press [SHIFT] and [RETURN] at the same time and then move the cursor back up to the line. Now you can insert any characters *except* the cursor commands. There is no way around this Catch-22, so you may as well hit [RETURN] and type line 30 again.

When you get these first four lines right, type RUN and enter it. If you did everything correctly, a colorful geometric pattern will be displayed on the screen, row by row. If you have any problems, type LIST, then press [RETURN] to see your program again. Check each line carefully, and keep editing until your program RUNs properly.

Now let's see how we can use the cursor keys inside quotes to make this display more intricate. First, type the word LIST and enter it to have a look at the program we just ran. The next lines we type will be part of the same program and will be performed after the geometric pattern is displayed. To maintain this sequence our next instructions should begin at line 50:

```
50 FOR L=1 TO 11:REM VIC 20
50 FOR L=1 TO 19:REM COMMODORE 64
60 PRINT "[SHIFT CRSRUP][2 SHIFT CRSRLEFT][CTRL BLK]";
70 FOR M=1 TO 100:NEXT M
80 NEXT L
```

The 2 before SHIFT CRSRLEFT in line 60 is a subscript telling you to press [SHIFT] and [CRSRLEFT] together twice.

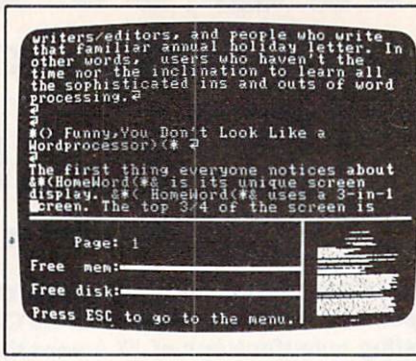
Enter these lines and RUN this latest version. We put line 70 in only to slow the computer down so that we could see how the cursor is moved up and to the left just before a black block is put on the screen. There is one more important key you should know about. As the black blocks "step" up the screen, hit the [RUN/STOP] key on the far left side of the keyboard. This stops the program dead in its tracks. As you learn to program, you will need this key from time to time to stop a program that has gone astray. To start again, simply type RUN.

We now have a stairway going up; we can just as easily put in a stairway going down:

```
90 FOR N=1 TO 10:REM VIC 20
90 FOR N=1 TO 18:REM COMMODORE 64
100 PRINT "[2 SHIFT CRSRLEFT][CRSRDOWN]";
110 FOR M=1 TO 100:NEXT M
120 NEXT N
```

Study these lines carefully to make sure you understand how they work. But don't stop there. Experiment with new colors, shapes, patterns, and always try to have the computer do as much of the work for you as it can. HCM

HomeWord Bound . . . from p. 90



The Page Sketch feature lets you see the layout of the page as you write.

Figure 1

HomeWord does not separate the Edit mode from the Type-in mode. Many word processors separate these two modes even for minor editing functions like inserting words and going back through text to make deletions. On the Apple IIe, you have full use of the arrow keys, and can go back in the text as far as you need for insertions or deletions. *HomeWord* has a separate Edit mode for major text changes like erasing, moving, finding and replacing, or copying large blocks of text. Also, unlike many word processors, *HomeWord* does not separate the Format/Layout mode entirely from the Type-in mode. Layout considerations such as spacing between lines, boldface, underlining, and margins are all controlled from within the document.

HomeWord's Boundaries

In an effort to make *HomeWord* easy to use, Williams has unfortunately sacrificed speed of execution in some commands. *HomeWord* manipulates text by using a technique it calls "painting." This simply means putting the cursor at the beginning of the text you want to alter (i.e., move, copy, delete, etc.) and then moving the cursor across all the text you want to change. Although this is a good, concrete way of showing a writer what he is doing, the process sometimes takes a long time. When manipulating large blocks of text, it might be easier to use line number designations or another faster system for spotting the beginning and end of the text segments to be altered. The system's boldface feature is also cumbersome; to change one portion of text to boldface takes ten steps! The manual does not mention that the boldface command is treated like part of the text. You must paint the word BOLDFACE as well as any text that you want to manipulate. If you do not, the command will be left behind in the move like a mislabeled box lost by the Bekins man.

Documentation

HomeWord's documentation is in three parts: the audio instruction tape, the transcript of that tape, and *The HomeWord Story*. The audio tape is a well-paced introduction which familiarizes the user with both the computer and *HomeWord*. Its interactive approach has you using the hardware and software right away, giving you that I-Can-Do-It feeling. Tape is a good medium for instructing children, but parents are advised to guide their child through the instructions the first time. The transcript is a handy reference for reviewing procedures not discussed in the manual, such as setting up *HomeWord* to access your printer.

The user's manual is only 33 pages long. Each mode is covered separately and boldly marked for easy reference. The manual would be more useful if it included better coverage of the special mode for producing outlines and if compatible printers were listed. Its style is somewhat condescending, but it is easy to read and understand.

Special Features

HomeWord uses unique symbols called *icons* to represent the menu choices. The first menu that appears on the screen looks like the screen in Figure 2.

With the arrow keys, you can move the *icon cursor* (the large square over the symbols) back and forth to frame each symbol. A short explanation of that icon will appear. To make your selection final, press the [RETURN] key, and a new menu of icons appears. For example, if you pick Customize, icons representing these options will appear: Make Backup Documents, Change Preset Margins, Save Customize Choices, Configure Points, Type of Printer, and Accessories. These clear and simple icons eliminate the need to memorize a long list of commands, and make *HomeWord* friendly enough for the whole family to use.

Two other features that set *HomeWord* apart are its Help Key and Error Handling functions. Each computer has its own key or set of keys that will call up the HELP DOC, a screen version of the Reference Card. On the Apple IIe, use the [Open Apple] key plus the H key. After viewing the HELP DOC, you can print it to replace a lost Reference Card.

HomeWord is easy to use partly because the directions are so clear and complete, but also because it anticipates errors that could be disastrous. The automatic backup feature is a real lifesaver. Whenever you save your document, *HomeWord* makes a backup copy for you at the same time. Not enough can be said about how exasperating it is to lose hours of work because of a power outage. In the middle of saving this review in a HOMEWORD file, my computer was accidentally disconnected. Thanks to this back up feature, I still had the original document saved under HOMEWORD.BAK!

HomeWord also tells you what to do to prevent errors before it's too late. All word processor users dread losing corrections before they have been printed. Suppose for instance you have ESCaped from the typing mode and cataloged your disk to find the name of a file of material that you want to insert. Once you know the filename you will, naturally, want to "get" that file right away to insert the material. *HomeWord* will give you the following message:

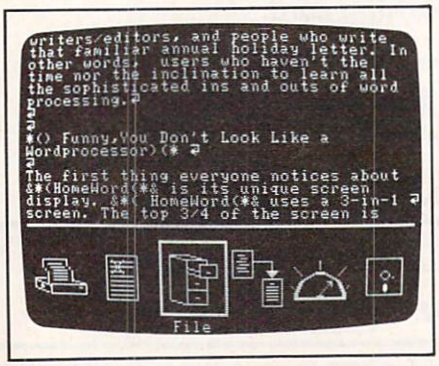
YOU CHANGED THE DOCUMENT ON THE SCREEN SINCE THE LAST TIME YOU SAVED IT. THE CHANGES WILL BE LOST IF YOU GET ANOTHER DOCUMENT.
PRESS RETURN TO GET DOCUMENT.
PRESS ESC TO CANCEL.

One final plus with *HomeWord* is its See Final Document option. It's a good idea to take advantage of it to check your formated copy before printing. You will find that this option saves you both paper and time.

Performance

Although *HomeWord* is geared to the beginner, even those accustomed to word processing will find it fun to use. With its automatic formating, constantly visible layout, and efficient error-handling techniques, it is an outstanding system for composing attractive documents. As of this writing, Sierra On-Line is testing the Commodore 64 and PCjr versions of *HomeWord*. We have seen the keyboard overlay designed for use with the PCjr (see the PCjr article in this issue), and it promises to make this version even simpler to use than either the Apple or Commodore versions.

HomeWord's performance is consistent with its advertising. It is an inexpensive, easy to use, and efficient word processor for the whole family. But more than that, it is a whole new concept in computerized writing. In fact, it may not be long before *HomeWord* will be *the* household word for word processing. HCM



Unique symbols, called icons, appear on the screen to show the various menu choices.

Figure 2

JOYSTICKS BUILT TO LAST!

MADE SPECIFICALLY FOR TI 99/4A
NO ADAPTORS NECESSARY

SPECIAL MAIL ORDER OFFERING

Limited Offer Only

$19.95 EACH PAIR

WORKS BETTER & LASTS LONGER

If you've tried to use other joysticks on your 99/4A, odds are they have not worked properly and/or broken down in a short period of time. Super Stik™ is the only joystick on the market (other than the TI joystick) made specifically for the TI 99/4A Home Computer with no adaptors necessary. What's more, Super Stik is made to last longer and work better than any other . . . backed by a one year limited warranty. No other joystick can match this claim.

WHY SUPER STIK IS BETTER

As mentioned, Super Stik is made specifically for the TI 99/4A home computer. This means that there is no need for special adaptors like other joysticks. A common 5' cord is included with each order.

The Super Stik Model J1020 has "built-in" isolation electronics that prevents "Crosstalk" between the two units. Without this isolation, joysticks will not function properly on the TI 99/4A.

J1020 units feature a simple, yet durable design with fewer moving parts. The housing is designed to completely stop handle movement before damage can occur to the contacts, providing a unique "positive stop" action. The assembly withstands a breaking strength of more than 50 lbs. and has been tested to more than 500 continuous hours without failure.

ARCADE FEEL & PERFORMANCE

Users of the Super Stik love its feel and performance. It features a long, comfort contoured handle, 8-way 45° angle control, and handle mounted firing button.

WITH THIS COUPON OFFER ONLY
SUPER STIK MODEL J1000

Also Available For Only **$9.95** EACH

The Super Stik model J1000 is compatible with Atari® Video Game, Atari® 400/800 Computers, Sears Video Arcade and Commodore VIC-20/C-64 Computers. The same rugged construction and super performance is offered. Don't miss the Super Stik at this price.

Atari, Commodore, Sears and Texas Instruments are registered trademarks. Super Stik is patent pending. Copyright, 1983. Made in U.S.A.

DON'T MISS THIS SPECIAL MAIL ORDER OPPORTUNITY

This may be your only opportunity to purchase the right joystick for your TI 99/4A at this low price. All orders will be allowed a 30-day money back guarantee if not **completely** delighted by its performance.

SEND YOUR PAYMENT WITH THE FOLLOWING COUPON TO:
RC DISTRIBUTING COMPANY
4016 Sanguinet
Fort Worth, Texas 76107

To Figure Your Payment For Each Joystick:
Super Stik J1020 (pair) .... **$19.95** each
Super Stik J1000 .... **$9.95** each
Postage & Handling for Each Item Ordered **$2.00**
For Texas Deliveries Add 5% Sales Tax, **$1.00** for each J1020
$0.50 for each J1000

Please rush my joystick(s) to the address below. My payment of $ ________ is enclosed for the appropriate Super Stik Model(s). If not completely satisfied with your purchase, return your joystick(s) within 30 days of receipt (in the box it came in) for a full and prompt refund. Offer good only while supply lasts.

☐ J1020 Super Stik Quantity ________

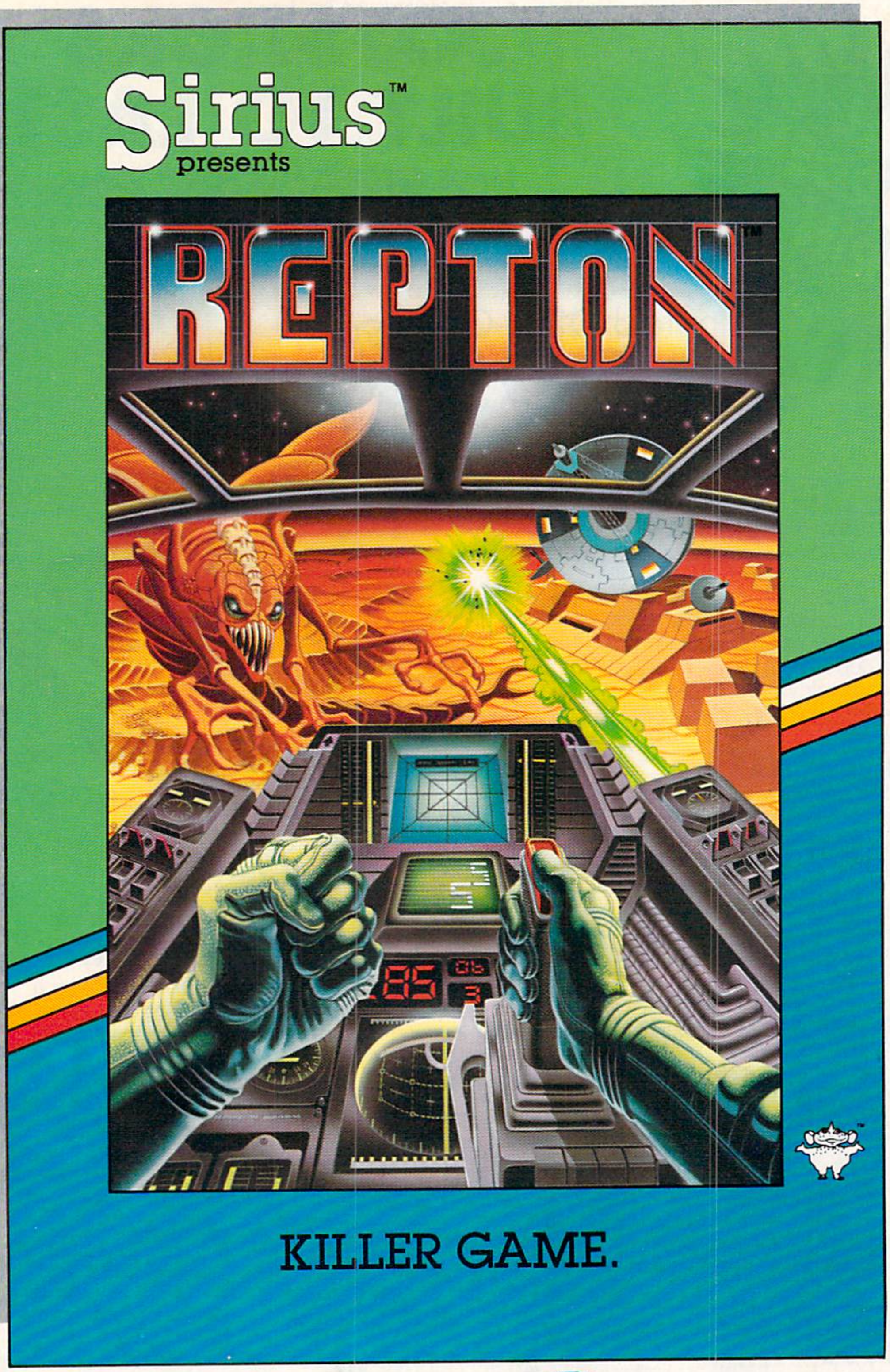
☐ J1000 Super Stik Quantity ________

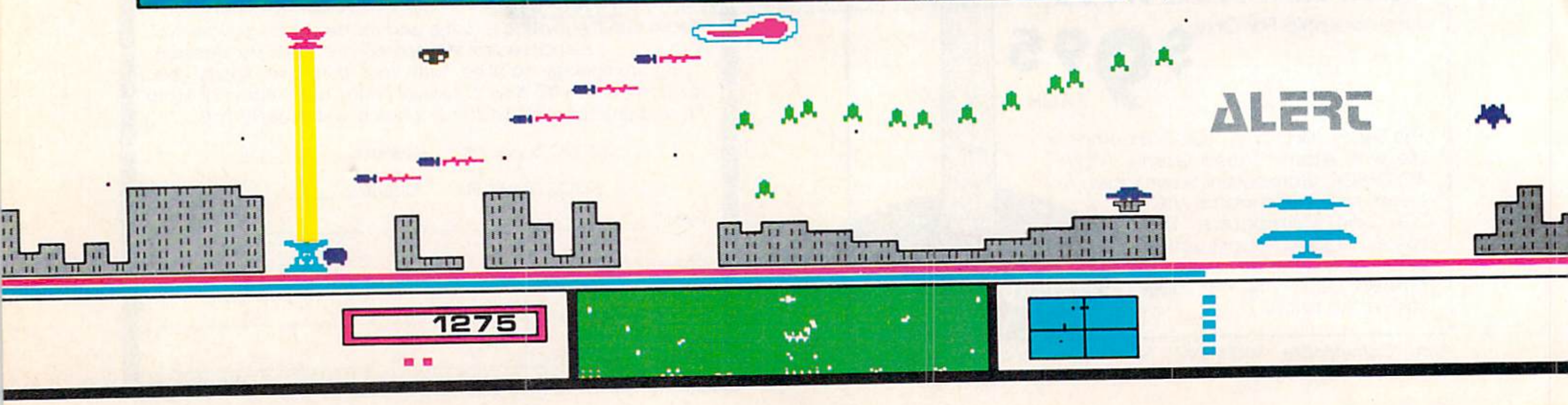
NAME ________________________

STREET ADDRESS ________________________

CITY ________________________

STATE ________________ ZIP ________________


Sirius™
presents
REPTON™
KILLER GAME.

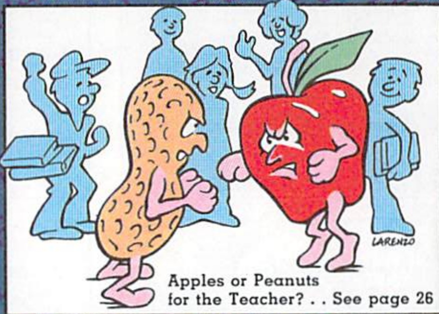

ALERT
1275

Volume 1, Number 1 $1.50

HOME COMPUTER DIGEST™

News and Happenings in the Home Computer World

- Window Wars Rage On
- Home Computer Market Shake Out
- Las Vegas Show Stars
- TI PLATO: To Be or Not To Be
- Industry Standards Made in Japan?



Apples or Peanuts for the Teacher? . . See page 26

CONTENTS

DEPARTMENTS

On Line

We are pleased to introduce the first issue of *Home Computer Digest*. This journal from Emerald Valley Publishing Co. is a companion to *Home Computer Magazine*, our long-standing resource for home users of the leading microcomputers. *Home Computer Digest* is designed to complement the practical, user-oriented nature of our larger magazine by offering readers a broader view of the ever-changing home computer industry.

Within these compact pages, we will cover all phases of the computer world. The *Digest's* regular columns—*New Tech News, Industry Watch, Software Trends, Novel Applications, Gameware Updates,* and *Computers in Education*—will compile essential industry developments in a concise, easy-to-read format. Each month, comprehensive feature articles will focus on important dimensions of the new technology to help you develop an in-depth understanding of current directions in the computer field.

Like *Home Computer Magazine*, this publication is written especially for you, the *home* computer user, and we welcome your input. We will consider carefully any questions or suggestions we receive from our readers, and we encourage those of you with inside sources or keen ears to send in relevant news for publication. Let us know how *Home Computer Digest* can help keep you well-informed in this dynamic, exciting Information Age.

HOME COMPUTER DIGEST is published by Emerald Valley Publishing Co., P.O. Box 5537, Eugene, OR 97405.

Any Questions?

The present global ignorance of computing may come, in part, from our natural aversion to asking simple questions—for fear of revealing only a shallow knowledge of vital topics.

Why not let someone else ask the questions while we sit back and benefit from the reply? That's the purpose of this column.

Q. Why do I need a data storage device? What kinds are there?

A. Unless you use your computer strictly as an entertainment machine, you will need to store your programs or data either on tape or on a disk system. The cheapest option is an ordinary cassette tape player. It is easily plugged into your computer and is fairly convenient for storing data; but sometimes it won't pick up the data unless the volume and tone are adjusted to a precise setting. In addition, taped information transfers slowly compared to other media. At this point, the most popular storage method is a disk system using floppy diskettes. These thin little wheels of mylar encased in plasticized envelopes were once a luxury for the few. Disk drives are now available for about $250, and $150 drives are in the works.

Q. I bought a "64K" computer, thinking I had a machine powerful enough for word processing and for running a small business out of my home. When I plugged it in, it turned out I had only 38K of memory available. What happened? Is this false advertising?

A. The claims of some computer manufacturers may be misleading, but they are not false. Your machine is endowed with 64K, but nearly half of that capacity is being used to carry out non-storage functions such as interpreting BASIC statements and running the hardware. Fortunately, some micros let you add on the memory you need; for $100 or so you can get extra memory in the form of a "card" that plugs into your computer console. Some users may be annoyed at having to make this extra purchase, but it may still be a cheaper option than the models that boast a large usable memory—and a big price to go with it.

Q. Sometimes I leave my computer monitor on for hours at a time. Can this practice harm the machine?

A. The back of your television screen or monitor is coated with phosphors which glow when bombarded with a stream of electrons. This electronic bombardment can be controlled to create precise images on your screen, but if an image is kept on for a long period, the phosphors of that pattern will burn out, leaving a permanent "ghost" on your screen. Some computers have an automatic switch-off device to prevent this problem—the machine goes blank if a key is not pressed within a certain period.

Q. What exactly happens when I transfer data from the diskette or tape to the computer? Can data be lost during transfer?

A. One of the sublime beauties of computing is that your disk drive or cassette player will keep your data safe, no matter how much you mess up the program loaded into the computer. Many people are unaware that loading a program from tape or disk into the computer merely copies the material on the disk into the computer's memory. This is analogous to playing a record on your stereo which doesn't affect music stored on the record. Programs are destroyed only if you forget to record them, or if you take deliberate steps to erase them from the storage medium.

Q. We are concerned about getting a computer for our child as soon as possible. We know that he will need one in order to keep up with other children his age. The problem is money. How much will it cost to buy a computer for a five-year-old?

A. For many months now, massive ad campaigns have been urging parents to buy computers for their children so that they can "compete with the others." But it isn't clear what all these millions of youngsters are expected to *do* with their machines. It seems unlikely that many will become professional programmers. As for using a computer at work, it is not likely that great skill or long training will be necessary for most jobs—especially in years to come when computers will be extremely accessible. Computers can, however, be powerful educational tools. The LOGO language has proven very beneficial in helping kids learn math and logic. And a computer is great for helping children learn to read and spell. Fortunately, many such activities can be carried out on a simple computer system. An under-$100 machine attached to the family TV set and used with a cassette player is a perfectly good means of introducing your child to computing.

Q. I need a printer. What kind should I get?

A. It all depends on your needs. Basically, there are two kinds of printers: fast ones and slow ones. People who choose fast ones get a print-out with a "computerized" look. The letters are made up of dots printed out by a matrix of tiny pins striking the printer's ribbon. These dot-matrix printers for home systems can crank out copy at 50 to 200 characters per second. If you want typewriter-quality characters, you will have to sacrifice speed. A page printed on a daisy-wheel printer is indistinguishable from hand-typed copy, but such machines write at 12 to 55 characters per second. The dot-matrix printer is normally used for utilitarian data display such as program listings or department store sales slips. Letter-quality printers are generally used for business letters, press releases, and manuscripts.

Q. I ordered a printer through a mail-order house, and now that it's here I can't get it to work because my computer doesn't have what they call an RS232. What gives?

A. One of the curiosities and curses of buying computer products is that many items come strictly a la carte. The RS232 "card" is simply another collection of integrated circuits that conveys data from a computer's memory to a printer. This accessory can cost anywhere from $50-$200. In addition, you will have to spend $20-$50 for a cable to connect your printer to the RS232 card.

Window Wars

When Apple introduced the Lisa a year ago, it was met with less than the rousing reception Apple had hoped for, primarily because of its sky-high price tag. But Lisa's use of the screen as an "electronic desktop" caught the imagination of software developers and computer users alike. Extending the desktop idea to other computers is now becoming a major focus of the big software houses.

VisiCorp has been the pioneer in developing what is called an *integrated operating environment*. Designed to free the user from shuffling back and forth among programs and files for spreadsheet, graphics and word processor, *Visi On* opens up *windows* to several programs simultaneously. The user gains access to the system by moving a "mouse" (a sort of desktop joystick) across the desk. Using *Visi On*, the user can select a spreadsheet program and enter data, then move the mouse to select a graphics program to draw a graph based on the data just entered on the spreadsheet. Then, via the mouse again, the user can select the word processor and transfer the results to a report. This program interaction is all accomplished from the same screen, with no long delays between programs! The package was developed with the first-time user in mind, so there are no special languages or codes to learn.

What Price Windows?

One major stumbling block in this scheme of easy program and file interaction is the amount of memory required. VisiCorp hoped its *Visi On* would provide a window environment requiring only 128K of RAM and two floppy-disk drives. But the product as it is now being shipped requires a hard disk, a floppy-disk drive, a minimum of 384K of RAM (with 512K recommended), and the mouse device. The package lists for $495 without the mouse; the mouse adds $250 to that price. *Visi On* is available for several IBM PC-compatible machines and the DEC Rainbow, among others (as long as they have plenty of memory).

Microsoft's answer to *Visi On* — *MS-WINDOWS* (slated for release in the second quarter of 1984) — will require only 192K of RAM and two floppy-disk drives. Because it is an extension of MS-DOS (so Microsoft claims), it will be compatible with nearly all existing application software that runs from MS-DOS. This is one-up on *Visi On* which supports only applications that are designed exclusively for use with the *Visi On* system. Microsoft expects the retail price on their package to range from $50 to $150, depending on the hardware supported, and maintains that *MS-WINDOWS* should work on any computer running MS-DOS. However, *Visi On*'s late appearance on the market and larger-than-anticipated memory and data storage requirements might cause an impartial observer to take a somewhat skeptical view of Microsoft's ambitious promises. We'll have to wait and see if *MS-WINDOWS* is the look of the future or just a promising silhouette on a shade that hasn't yet gone up.

The rest of the software world is not just standing by to see which of these two giants will emerge from all this window innuendo. The DesQ system from Quarterdeck (planned release was in December) already integrates existing programs in a window-like environment, but Microsoft's software may prove to be more flexible. Meanwhile, Digital Research (CP/M's developer) has developed an approach quite similar to the window concept with its Concurrent CP/M operating system.

Windows for the Home

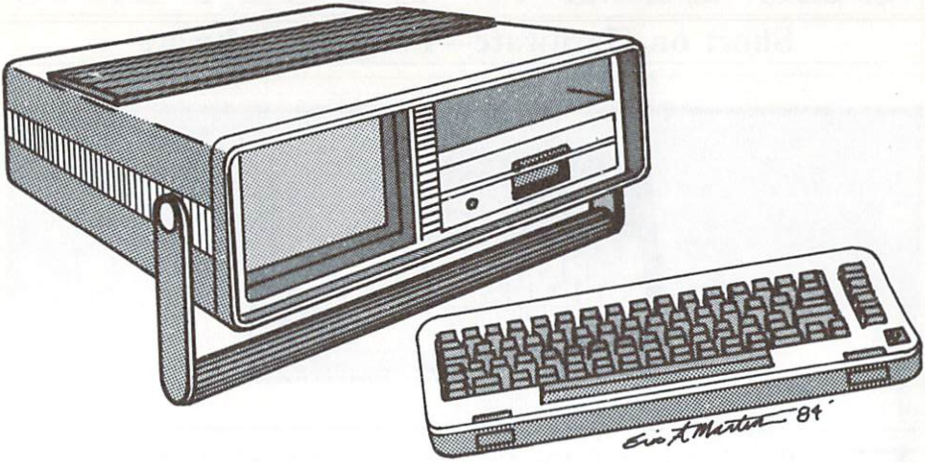
So when can the home computer user expect to be looking at his programs through these marvelous windows? The Apple Macintosh, which was introduced January 24, has a mouse and windowing built in. At $2,495, it will compete directly with the IBM PC. Apple also recently announced an *AppleMouse II* package ($150 retail) to bring mouse software to the Apple IIe in March. In addition, a program called *Appleworks,* which integrates word processing, database management, and financial modeling, will be available in March ($250 retail). While this package does not incorporate windows as such, it does boast that "users can easily move information. . . from any file to a word processing file, using the program's 'cut and paste' functions." With Apple marketing these programs, can windows be far off?

Commodore is also getting into the integrated software market. Their *3-PLUS-1* software package (a built-in option on their new home computer, the Commodore 264) boasts a windowing capability that allows simultaneous viewing of word processor and spreadsheet screens. Available as a ROM cartridge for the C-64 (and as an expansion for the 264), it will not require the large RAM of *Visi On* or *MS-WINDOWS*, but it will probably not be as flexible as these other software packages.

Software development is not restricted to large, established corporations. For example, a new company, Arktronics (established August, 1983, by two Michigan college students and the owner of the Golden Nugget casinos in Las Vegas and Atlantic City), is releasing an integrated package (February 1984) called *Jane* that runs on Apple II series and Commodore 64 computers. *Jane*'s developers claim that it could be modified to run on any computer with 64K of memory, and they hope to market versions for the IBM PC and PCjr in February 1984. *Jane* comes complete with mouse and runs with just one disk drive (Apple version) or cassette storage (C-64). This package includes word processing, spreadsheet, and file/list management, with more to come. *Jane*'s $295 price tag gives one pause, and we can hardly wait to see *Jane* RUN.

—Roger Wood

Sleeper of the Year:
The Commodore Executive



While everyone was buzzing with speculation about the Peanut, the Adam, and other "startling" announcements from the home computer front, Commodore came out with a breakthrough that has been kept surprisingly quiet, considering its importance. The Commodore 64 system has been bundled into a 5″ × 14½″ × 14½″ brown box that includes every essential but a printer. The Executive features a detachable keyboard, 5-inch color monitor, disk drive, and 64K RAM. Now the big surprise: a price tag of $995. That's little more than half the cost of the typical portable system.

When the first Executive 64 press releases came out last fall, some of the machine's specifications put me on my guard. Would a 5-inch screen even be readable? And how substantial was that detachable keyboard? Wasn't 27.6 pounds a bit heavy for a new-model portable?

When I finally got close to the machine, I was suprised by the clarity of the monitor display. It was small, yes, but the resolution was better than I had expected. The screen's readability, however, depends on your eyesight. The tiny screen is simply the portable option—this machine can easily be plugged into a standard television or monitor.

The keyboard is a joy to type on. It seems even smoother and more responsive than the standard Commodore model. The Executive is the first low-end system that is both powerful and portable. It can be used as a writing machine or business tool at the office, then brought home at night and plugged into the family entertainment center.

This model takes advantage of the huge library of software written for the original Commodore 64, and even has a slot for cartridges. It is compatible with all Commodore and VIC-20 peripherals, including modems.

Despite the extremely sparse press coverage given to this new computer, the demand for it has been overwhelming. No one should be surprised. The public has long been waiting for a reasonably-priced, powerful, portable computer.

—Greg Roberts

CES SHOW STOPPERS

Short on Hardware—Long on Software

Even though many of last year's exhibitors didn't appear at this winter's Consumer Electronics Show—or showed up as registrants but didn't exhibit—there were some who came to Las Vegas with promising new products to unveil. The big story was Commodore's introduction of its new "dedicated" computers.

According to sources inside Commodore, enthusiasm for the long-awaited 128K, 16-bit Zilog 8000 has waned. Their excitement is now centered around the new series of 64K "Specialty Computers." Commodore Market Development reps say these computers—the 264 series—will carry a price tag in the neighborhood of $500, well below prices for the Adam and PCjr home computers.

These application-oriented "cousins" of the Commodore 64 with their built-in software were the talk of CES. Reportedly, users need only turn on a particular model and its specialty program will RUN automatically. Word processing and an integrated, consumer-level LOTUS-style package called 3-PLUS-1 were on view at the show. Commodore's 3-PLUS-1 will be a cartridge option for the C-64 and will be offered as a built-in or cartridge option for the 264. The functions of the package are simplified word processing, electronic spreadsheet, and data-base management; the "PLUS 1" is a business graphics program.

There was a lot of pre-CES talk about a 364 model with built-in speech synthesis and a numeric key-pad, but the machine was a no-show at the booth.

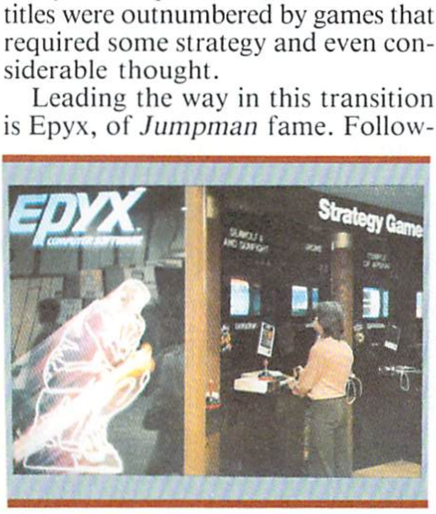
Even though these new-breed 64's look different from the original Commodore 64—sporting a stylish low profile with a streamlined tilted keyboard—they are compatible with most of the Commodore peripherals. Their software, however, will not be C-64 compatible because the new series uses a custom chip—the 7501 microprocessor—that is a combination of the 6502 and the 6510.

Commodore fans can also look forward to a built-in machine language monitor with 12 commands as well as more available memory. The new machines have a total of 64K of RAM with 60K RAM accessible through BASIC. This arrangement has more RAM available than the C-64 for users to PEEK and POKE about in to their heart's content! Other outstanding features include four separate function keys, an ESCape key, over 75 built-in commands, including RENumber and DELete, and graphics plotting using a special graphics mode. This mode uses all Super Expander Commands to give users a full 16 colors plus 8 luminance levels for a total of 128 working colors (just like Atari). The pixel resolution is the same as on the 64, but sprites are not available.

Apart from Commodore's new releases, this was a small producers' show. Timex introduced a 300-baud modem that may make a lasting industry impression with its automatic dial and answer features. A major attraction with this modem is free membership in the Source Telecomputing Network (a hundred-dollar value)! Another scene-stealer was the home version of Video Technology's Laser 3000. Yet another low-priced ($700) Apple-compatible computer, this one boasts four-channel sound, eight function keys, and built-in Microsoft BASIC. Although representatives claim that the Laser 3000 will run most existing Apple software, the user-jury is still out on whether this machine will take any sizeable bites out of big Apple's profits.

Software trendsetters on the scene at CES were showing titles that reflected a new focus. As industry analysts had predicted, arcade classic titles were outnumbered by games that required some strategy and even considerable thought.

Leading the way in this transition is Epyx, of *Jumpman* fame. Following a re-organization of their popular software line into four different categories—"action," "action-strategy," "pure strategy" and "learning fun"—Epyx representatives report that most of the company's new titles will be action-strategy games like *Pit-stop*. Not just another fastest-take-all race around a speedway, this game requires players to decide when to take a pitstop, what to do when they get there, and how much time to take. Each of these variables actually does affect the outcome of the game so that play will be different every time. This kind of software could provide users with a longer lasting challenge for their money.



While arcade-oriented publishers are putting more cognition than ammunition into their software, educational software producers are adding more action to their learning activities. CBS Software's "edutainment" line of learning games stresses "multiple-player, multiple cooperation" for users from preschoolers to adults, according to Ed Auer, CBS Software President. Look for *Strategy World*, *Design World*, and *Knowledge World* to debut soon. DesignWare is looking at the number two spot on the list of top ten producers of educational software with 24 action-learning titles.

Spinnaker, long-time leader in the educational marketplace, introduced *Trains*, an economics simulation package available on Apple, Atari,

cont. on pg. 15

Software Trends

CROSS FERTILIZATION

Simon & Schuster has joined the pell-mell rush of book publishers to jump into the computer age with the announcement of their Electronic Publishing Division to produce home computer disks. As could be expected, Simon & Schuster will concentrate on programming for education. The S&S book publishing operation has been up-dated with a new Computer Book Division. Barnes & Noble, whose main New York bookstore is the world's largest, will bow to the publishing trend with a separate store next door for computer books and software. B&N's Computer Book Shop and Software Center is a physical acknowledgement of the fast-increasing dominance of computer books in the publishing industry. Outstripping even the fiction market, computer book sales are taking over. *Publishers Weekly* has run features on computer book and software retailing weekly, and recently devoted a special issue to this hottest of all publishing topics. Software companies themselves are getting into the act. Microsoft Corporation, the big-time hardware and software company in Bellevue, Washington, will begin to publish books, many of which will be guides to Microsoft hardware and software products. It looks like it will be some time before computers kill off the age of Gutenberg, so widely predicted a few years ago. They may even have resuscitated it.

ALL THE TIME AND EVERYWHERE

ROMOX thinks the new reprogrammable cartridges they've just begun to market in convenience stores in California will become the most popular kind of software purchase. Retailers will lease ROMOX's reprogramming terminals for $160 per month, and customers will be able to choose and produce new software via electronic downloading on site at stores 'round the clock and—if ROMOX's ambitions pan out—'round the world. ROMOX plans to contract with one software distributor per country: with Japan's Aster Co. first, to be followed by distributors in Canada, England, France, Denmark, Sweden, Norway, and Finland. Domestically, ROMOX would like to persuade major companies such as Microsoft to distribute their software electronically through ROMOX's machines. Right now, ROMOX handles programs for the TI, Commodore, and Atari machines and has licensing agreements to electronically distribute programs from Creative Software, Epyx, Fox Video, Funware, HES, Mattel, Navarone, Sierra On-Line, Telsys, UMI, and Vidtec. Microsoft will be producing software for the IBM PCjr, and ROMOX would like to distribute those programs.

VENTURE-CAPITAL MUSCLE TO SQUEEZE THE LITTLE GUY?

If we're to believe Steve Axelrod, a "software agent" interviewed by Mary Alice Kellog for *ADWEEK*, the days of the kitchen table software outfit are coming to a close. According to Axelrod, marketing know-how accompanies the big-time venture capital that's now going into software companies. Axelrod foresees bigger and fewer software companies who produce sophisticated, highly competitive programs for several machines at once and who market their products with a skill and aggressiveness formerly reserved for hardware selling. As if to substantiate Axelrod's identification of a trend, TA Associates of Boston has just put $2 million venture capital into Alpha

Software of Burlington, MA. This is the first time outside capital has gone into Alpha, and they've since come out with full-page glossy magazine ads.

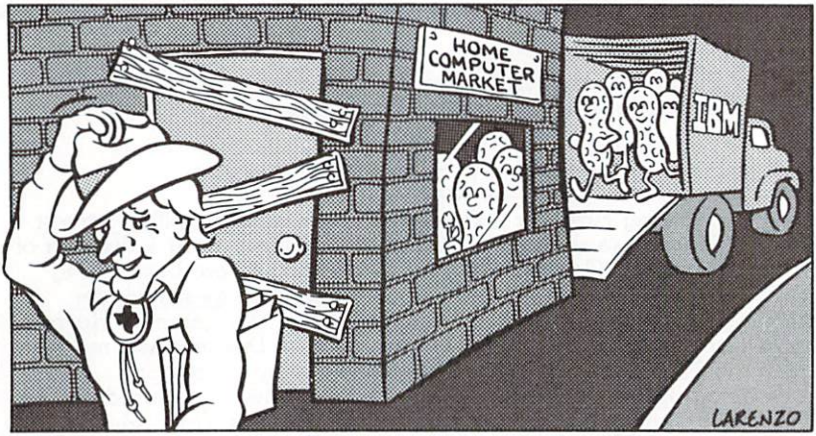
STAR-STUDDED AFFILIATIONS

Rapidly expanding First Star Software has sold a non-controlling interest to Warner Software (a division of Warner Publishing, itself a division of beleaguered Warner Communications). First Star, whose best-selling *Astro Chase* for Atari game systems was authored by award-winner Fernando Herrera, has announced *Bristles*(tm), a new game by Herrera for the Atari and the C-64. Herrera, now Head of Design and Engineering for First Star, is notable as the first software author to conduct an autograph signing. The Toronto Film Festival sponsored an evening commemorating Herrera's work, and UCLA's Video Game Conference will honor Herrera this spring. First Star will pursue its interest in tie-ins with other media via independent film producers Richard Spitalny and Bill Blake, and in a joint venture with MARVEL COMICS Group to design and market software featuring Marvel Comics characters. Warner Software/Warner Books in the meantime will affiliate with LIST Magazine to produce a line of integrated computer book and software packages, its first for IBM PC business applications. Meanwhile, the parent company, Warner Communications, will try to fend off a takeover attempt by Rupert Murdoch, the controversial Australian newspaper baron who now controls major newspapers in England and the U.S.

MILTON BRADLEY-TI AGREEMENT

Milton Bradley and Texas Instruments agreed last year that MB would market their MBX Expansion System with three of the ten games coproduced by the two companies, and that TI would sell the other 7 games. This agreement was to have gone into effect during the first quarter of 1984. When TI left the home computer market, Milton Bradley and TI came to an amicable agreement rather easily: Manufacture of the MBX Expansion System has been terminated to the mutual satisfaction of both parties, and neither company has further obligations to that project. Milton Bradley, who had produced the MBX for only two weeks prior to the TI pull-out announcement, will not market the device, but has distributed the surplus through internal employee sales at both companies. Production of the software for the MBX had begun and will be completed. The three games that require use of the MBX unit (*Terry Turtle's Adventure*, *I'm Hiding*, and *Championship Baseball*) will be sold to company employees with the MBX units. The seven titles TI was to have marketed (*Honey Hunt*, *Sound Track Trolley*, *Space Bandit*, *Sewermania*, *Big Foot*, *Meteor Belt*, and *Super Fly*) have been shipped and will be sold internally as well as through TI's regular retail channels. The suggested retail price of the software has been cut from $49.95 to $29.95. When the current software inventory is gone, MB and TI will discuss future licensing arrangements.

TI Pullout The Aftermath



A lead player walked off the set, a tardy prima donna pirouetted to center stage, and the rest of the troupe waited in the wings for their roles to be re-cast. Before the curtain had fallen on this end-of-the-year drama in the home computer world, industry analysts had filed their reviews and were braced for the big showdown. But the last act is still being written by consumers, who are clearly ad-libbing in this "who'll-buy-it" mystery.

Instability is nothing new in the high-tech market. But the concurrent demise of Texas Instruments' 99/4A and the unveiling of IBM's PCjr gave rise to predictions of an inversion of established buying patterns. Sales of low-end computers were expected to suffer from consumers' fears of being "orphaned" by the price wars that knocked out TI, whereas manufacturers of higher-priced models anticipated a boost from the reflected glory of IBM's impeccable reputation. Market analysts predicted an overall "wait and see" attitude, with consumers holding out for stabilization in the low end and postponing expensive purchases until the long-delayed arrival of the Peanut.

TI Loyalists and Converts

Instead, a wave of buying mania swept the country and left dealers struggling to keep up with the demand for lower-priced models. Heavy holiday sales depleted retailers' stocks as drastic price cuts stimulated new consumer interest in joining the ranks of home computer owners. "Fire sale" prices made TI wares irresistible to Christmas shoppers, who queued up at discount chains to purchase the remaining $149 consoles for a mere $49.

The estimated 2 million + TI user base is both a temptation and a frustration to third-party hardware and software suppliers. A number of manufacturers have expressed interest in supporting the 99/4A group, but these suppliers must seek alternate methods of distribution, as most retailers will not continue to carry TI compatibles. Only those companies that can convert to mail-order and access communication channels with TI users (e.g., mailing lists, users groups, and targeted publications) are expected to succeed.

Rumors of 99/4A look-alikes have surfaced, and many concerns are expected to cash in on the machine's popularity. But Texas Instruments may not be through with the home computer world for good. Some analysts believe that the corporation's cautious exit from the scene indicates a desire to slip back into the running at a future time. TI has been very protective of its distribution network, and even continued to support sales of the discontinued 99/4A through the Christmas season with at least half of the company's prepurchased TV ad time. Sales of other Texas Instruments products (hand calculators, educational aids, and the CC-40 portable computer) have been strong.

The Upshot Still Not Clear

IBM may have kept the public waiting too long. When the release of the PCjr was delayed beyond December's peak market, frustrated consumers who felt they had to have a micro under the tree turned elsewhere. Advance publicity for the Peanut may have over-inflated consumers' expectations; then a wave of cynical reviews hit, suggesting that the Peanut's capabilities might not justify its cost. Many sources began stressing the lack of compatible software for home applications. Executives with an IBM PC at the office might work at home in the evening on a junior, but garden variety home computer users will have to wait for IBM to woo them with software, or stick with micros that put more emphasis on the "home" in home computers.

Although IBM can be accused of neglecting the home market, its appeal to business and educational institutions has been strong. Virginia Polytechnic Institution and State University will require all entering engineering students to purchase a PCjr, PC, or XT. The College of Engineering at VPI contracted with IBM for 1,600 PCjrs sight unseen, and is willing to increase the order to 4,000 if the actual goods prove satisfactory.

Other high-end producers may profit from these year-end dynamics. Apple executives have stood firm against the tide of price slashing, discount distribution channels, and home entertainment marketing. Apple's sales approach still emphasizes user guidance and support, with sales through authorized dealers only. The delayed release of the Peanut may have created a windfall for Apple.

Flash-in-the-pan profits generated by the TI pull-out were apparent in the low-end market. Discount chains across the country reported record holiday sales of lower-priced models. Christmas shoppers stood a better chance of finding a Commodore 64 or VIC-20 on the depleted shelves, and both models moved quickly. Commodore's new line of 64K "Specialty Computers" and the Commodore Executive may be the reason Commodore did not raise prices, as did Coleco and Atari, who moved into the gap left by the TI's departure.

The sudden pull-out may have been too much for manufacturers at the extreme low end of the price scale, however. Doubts have surfaced about the longevity of manufacturers like Timex/Sinclair who fought vigorously against TI in the price slashing competition, but may not have enough of a computer to withstand market pressures. The clearance prices on the TI-99/4A have also infringed on sales of Radio Shack's TRS-80 color computers.

Advances in technology and high-powered advertising strategies will continue to sway a fickle public. But independent-minded consumers may still have a few surprises in store for industry experts who attempt to make predictions in this quickly changing market.

—Joan Killough-Miller

Novel Applications

NO SOONER SAID THAN DONE

New hardware developments will mean greater independence for handicapped users who may have difficulty with the standard computer keyboard. Voice recognition modules such as Voice Driver, designed by Voice Recognition Systems of San Francisco for use with the Apple II and IIe, allow system operation via speech commands without any special programming. The user's voice is imprinted by voice utility software, which can input a vocabulary of up to 80 words. The module can screen out ambient noise and performs with a recognition accuracy of 98%.

THE NEXT BEST THING TO BEING THERE

Rand McNally & Co. has gone beyond the boundaries of the conventional atlas with a line of educational games designed to teach geography, U.S. history, meteorology, and the time and seasons. The new skill-building software for the Apple II and Atari 800 computers is aimed at fourth to sixth graders. *Unlocking the Map Code* includes a "simulated flight plan" in which students pilot their way between world capitals managing risky terrain, limited fuel, and other flight obstacles.

WHAT NEXT?

Just in time for 1984 comes the Expando-Vision interfacing box by Stimutech, Inc. Commodore and Atari users can hook it up to their television sets and undergo self-improvement programs with subliminal messages flashed on the screen every 2½ minutes. The messages, which appear too quickly to be consciously noticed, are said to be "imprinted" on the subconscious to help the viewer reduce stress, lose weight, or stop smoking, for example. So now you can watch *Love Boat* and *The Dukes of Hazzard* and know that you will be a better human being for the experience.

BETTER HOMES AND CONSOLES

Look for new hardware and software systems that can manage home operations and act as central timers. Products are being developed to control house lights, give you a wake-up call, detect smoke, activate sprinklers, and even heat up your hot tub. Our editors have even seen a TI console turned into a burgler alarm. Now if they could only come up with something to get the kids off to school. . .

THE FAMILY THAT COMPUTES TOGETHER . . .

The Enchanted Village(tm) is a new concept in retailing that offers computer-related merchandise, in-store seminars, a library, and live performances—all dedicated to the concept of "Edutainment"(tm), a marriage of education and entertainment. Here customers can shop and learn in a family environment that even includes a supervised playroom. The first two outlets opened this fall in Pittsburgh, PA, and Fairfax County, VA, with a national chain slated for the future.

NO FLAB FLOPPIES

Good news, micro buffs. You don't have to leave your console to keep the old body in shape. Spinnaker has introduced *Aerobics*, the company's first venture into the adult software market. Led by a computer-generated figure called "Jane" (who else), this at-home fitness program offers a customized work out with challenging exercises and coordinated music. The program is available on diskette for the Apple, Atari and Commodore 64 computers.

SEND IN YOUR ANECDOTES AND JOKES

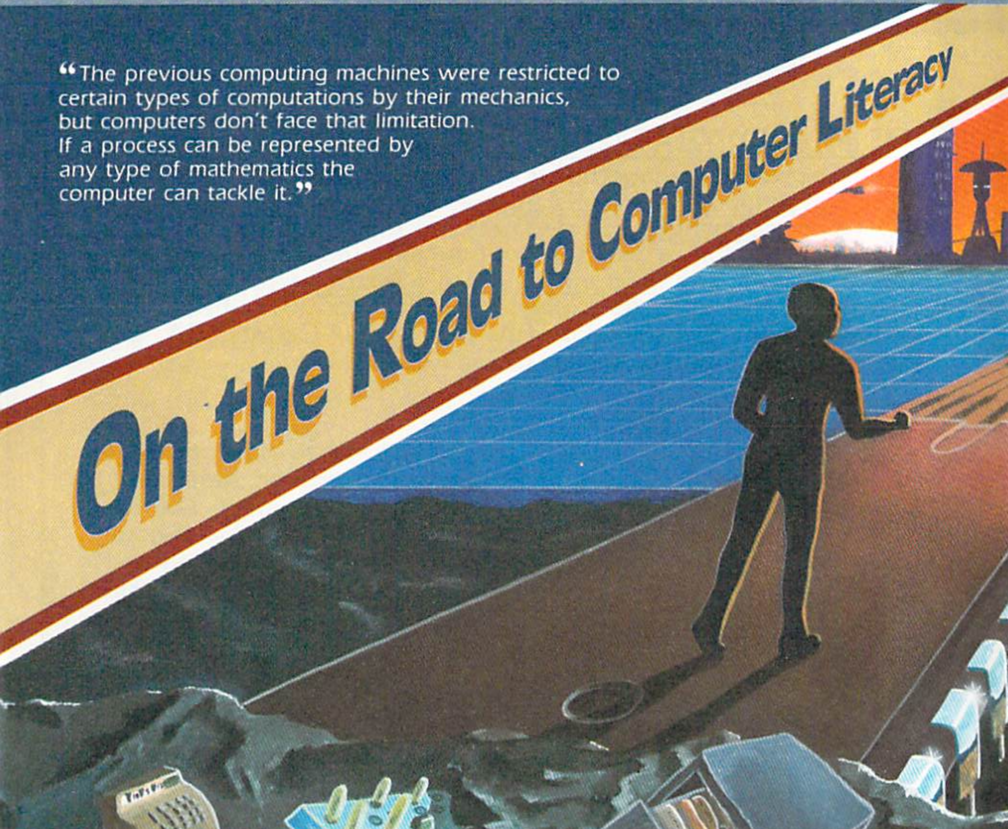
Do you have a good anecdote or joke to share? Don't limit it to the breakfast table or the office—Home Computer Digest will not only listen, but will send a $25 check to the authors of the ones we print. We are always looking for cartoons, bizarre news items, jokes and short anecdotes having to do with home computers, robotics, or any other aspect of the computer industry. Short items need not be typewritten. Be sure to put your name, age, address, and telephone number on each item you submit. Due to the large number of contributions received, none can be returned. Materials chosen will be subject to conditions set forth on the Masthead page of *Home Computer Digest*. Mail all submissions to:

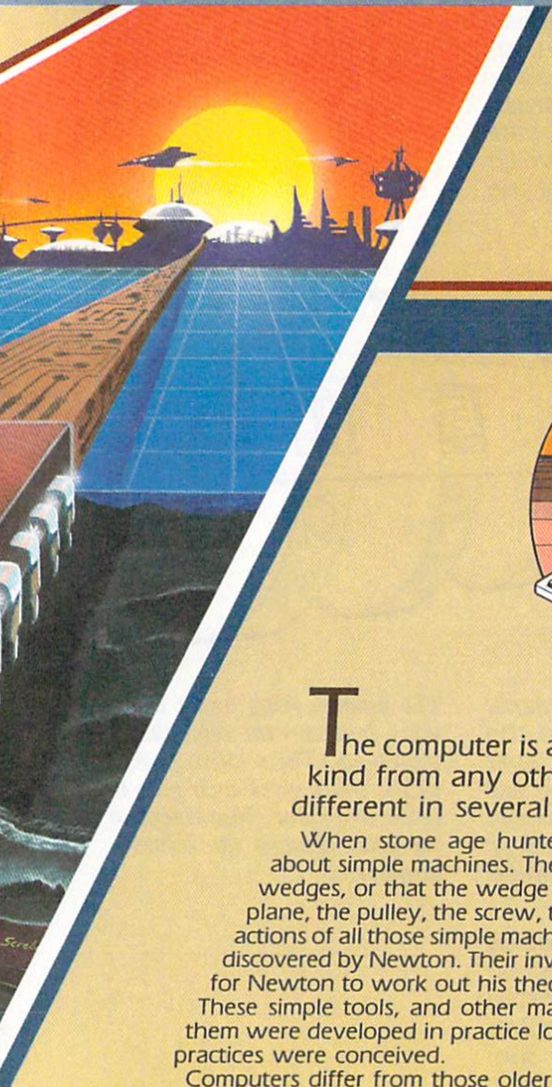
Features Editor
Home Computer Digest
44 Club Road,Suite 210
Eugene, Oregon 97401

CES Show... cont. from pg. 9

and Commodore at a modest $39.95. With each player taking on the role of railroad supervisor, mental and physical action go hand-in-hand. Spinnaker leads the pack when it comes to putting *real* physical action into home computer software. Show-goers gave *Aerobics* a real work-out (or was it the other way around!). In fact, health and self-improvement packages abounded at WCES. With products like this one and Synapse's *Relax* (for the IBM PC, soon to be available on the Commodore and Apple computers), the phrase "Let's get physical" may take on new meaning for software developers in 1984.

—*Sharyn Lyon*

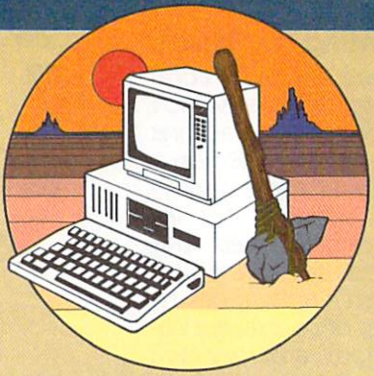

"The previous computing machines were restricted to certain types of computations by their mechanics, but computers don't face that limitation. If a process can be represented by any type of mathematics the computer can tackle it."
On the Road to Computer Literacy



COMPUTERS:
THE NEW TOOLS

ANALOG VS. DIGITAL

Part 1



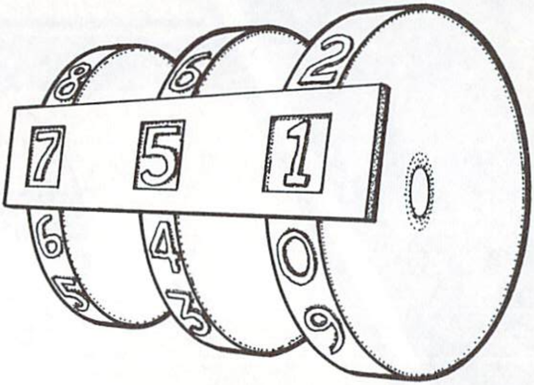
The computer is a machine. But it's a different kind from any other we have known so far—different in several important ways.

When stone age hunters made axes, they knew nothing about simple machines. They didn't realize that their axes were wedges, or that the wedge is a simple machine like the inclined plane, the pulley, the screw, the lever, or the wheel and axle. The actions of all those simple machines fall under the laws of mechanics discovered by Newton. Their inventors, however, didn't have to wait for Newton to work out his theories before they devised their tools. These simple tools, and other machines made from combinations of them were developed in practice long before the theories behind those practices were conceived.

Computers differ from those older tools in two important ways: First, they could not exist without numerous innovations in the fields of physics, chemistry and other related branches of science. But beyond that, computers deal with a new realm: the abstract rather than the concrete. A stone axe did physical work: It split open a billet of wood or an opponent's skull. The computer processes information: It splits numbers.

The manipulation of numbers is a relatively recent development in human history. We have tools from every period of history as far back as we have human remains. But the manipulation of numbers—mathematics—probably began only about the time recorded history began. And we can, in fact, date precisely a number of discoveries which had to precede the appearance of the computer.

Counting the Hours



The number lying under a given mark indicates the digit that gear represents. The relative position of the gears gives each gear its respective value. Movement of the gears changes the values displayed.

There are older computing devices which also dealt in abstractions and also depended on a variety of these discoveries. Sundials, for instance, measured the progress of the day, and divided it up into hours. Hours, of course, are abstractions, arbitrary divisions of the day devised by humans. Once the division of the day into the periods we call hours had occurred, other methods for keeping track of these periods, methods entirely independent of the sun's movement, came into being: water clocks, hourglasses, mechanical clocks. . .

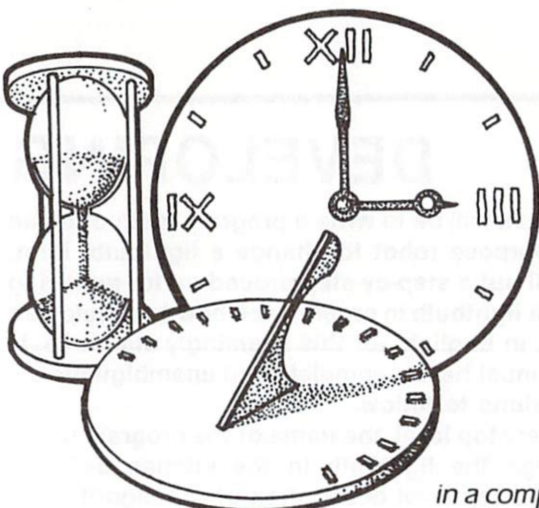
These mechanisms are all **analog** *devices. Analog devices use the position or movement of some physical object to produce their information. Sundials depended on the movement of the sun; water clocks and hourglasses on the movement of water and sand, respectively; mechanical clocks on the movement of springs, gears and pointers. The motion of the substance was, in effect, the computation; the physical location of the substance was the result—that is, the information.*

In the case of an hourglass, for instance, the movement of sand corresponds to the passage of hours in a *day. And we read the results from analog devices by measuring a quantity of the substance—to read an hourglass, we have to be able to figure out what portion of the total sand in the glass has moved from one compartment of the glass to another. On some glasses, this is easier than on others because a scale is etched on the glass. Nonetheless, we have to look at the* ***quantities*** *of sand.*

The States of Computing

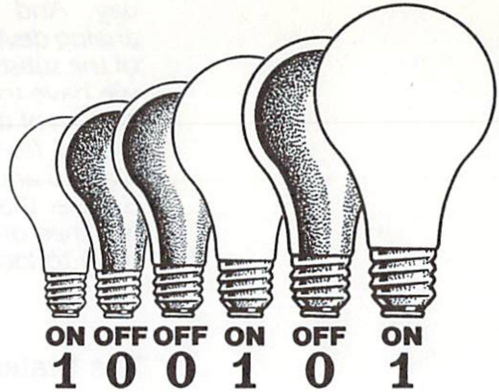
Later computing machines, like Babbage's computing engines from the mid-1800's or the more recent mechanical adding machines, were also analog devices. In addition to the movements of physical objects, however, they relied on ***positional notation,*** *which is the basis for our everyday numbering system. Every number we see has a value which depends on the position and value of the individual digits in the number. When we see 751, we know that it represents seven hundreds, five tens and one one.*

The shadow of the sundial's pointer marks the hour of the day. Hourglasses and mechanical clocks use the movement of sand and gears to compute hours. They measure only the passage of hours; when synchronized with the sun's movement, they denote a particular hour.



Most computers, by contrast, are not analog devices, but ***digital*** *devices. Rather than having moving parts, they use electronic components and changes in these components to represent their calculations and results. (There is, of course, movement in a computer, but not any kind we normally understand as movement. It takes place in the realm of subatomic particles.) These components have a special characteristic: They take on* ***states*** *—i.e., they can take on only one of two states. In essence, either they are charged or they are not. Computers use this system because there is a form of mathematics that exactly corresponds to this dual-state concept.*

This mathematics, although developed over the years by a number of people, is most often associated with the work done by an English mathematician, George Boole. His system, called Boolean algebra, reduces mathematics to its very basics. Boole said that there are two basic elements and three basic operations that can define all mathematics. Usually, the basic elements are represented as 0 and 1. We can think of the operations as addition, multiplication and negation. With combinations of these elements and operations, we can represent all of mathematics.



DEVELOPING

The task will be to write a program instructing an all-purpose robot to change a lightbulb. First, spell out a step-by-step procedure for replacing a burnt-out lightbulb in order to see how to develop an algorithm, in English, for this seemingly simple task. The robot must have a complete and unambiguous set of instructions to follow.

At the very top level, the name of the program might be "Change the lightbulb in the kitchen ceiling." Rewrite this top level description of the algorithm to read: CHANGE LIGHTBULB(KITCHEN)

Then divide the problem, or task, into its main subtasks. There is no single correct way to divide up a task, although some ways will be better than others. Try:

1. GO TO THE (KITCHEN)
2. REMOVE OLD LIGHTBULB
3. NOTE WATTAGE OF OLD BULB AND DISCARD
4. GET NEW BULB WITH SAME WATTAGE
5. INSTALL NEW BULB

The same operations used in Boolean algebra to combine elements have exact counterparts in the computer world. These counterparts manipulate the states of a computer's electronic components. With this binary-state manipulation at the component level, we can also represent all of mathematics—electronically.

This opens a vast range of potential for the computer. The previous computing machines were restricted to certain types of computations by their mechanics; computers don't face that limitation. If a process can be represented by any type of mathematics, the computer can tackle it.

Many electronic components are like light bulbs: They can only be on or off. They also can be interpreted as the digits of a number in their respective positions, but they can represent only two digits: 1 or 0, on or off. This numbering system is known as** binary **(from the Latin for two).

That immediately presents us with two difficulties central to our use of the computer: (1) finding an appropriate mathematical representation for the problem we're working on, and (2) finding a method for translating that representation into terms compatible with our particular computer.

ALGORITHMS

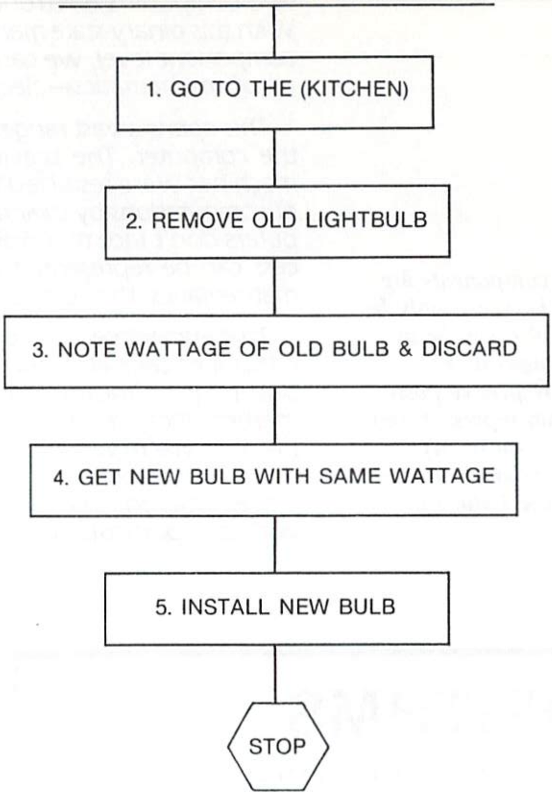
Notice that these subtasks are arranged in a natural *sequence* that cannot be changed. For example, it is simply impossible to remove the old bulb before going to the room where it is to be found. Also, there are not yet sub-tasks for handling problems—e.g., if at Subtask 4 there are no fresh bulbs with the correct wattage. Though at this level they can be ignored, such details must eventually be taken into account.

The procedure is similar to the grammar school exercise of outlining a report. Each step is broken down into sub-steps until they can be translated into instructions the robot can understand. Development of all algorithms follows this procedure of refining the subtasks into progressively smaller steps until each step is simple and unambiguous enough for the robot to execute. The flow chart on the next page is one way to represent an algorithm.

DEVELOPING AN ALGORITHM

(A Flow Chart)

CHANGE LIGHTBULB (KITCHEN)

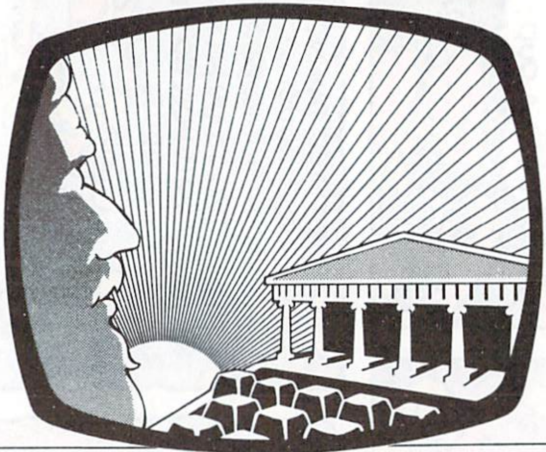


Clear Steps

Once we have defined the problem in mathematical terms, we next have to present it to the computer in clear steps. We have to devise a procedure with a definite beginning, definite steps, and a definite end. Such a procedure is called an ***algorithm****. We use algorithms in various guises in everyday life. Recipes are the most common examples: They tell how much of each ingredient to use and how to prepare them for a given dish. We don't have to use a computer language to define steps clearly, though. Clear plain English sometimes is sufficient; other cases call for mathematical notation. In most cases, however, a combination of both is required.*

Once we have an algorithm which defines all the steps necessary to solve a problem, and indicates all the data (elements and transformations) necessary for that solution, we can translate that algorithm into forms computers can use: Boolean elements and Boolean operators. HCD

PLATO Lives



Even though they've moved out of the home computer market, Texas Instruments has no plans to abandon their PLATO project. In response to the anxious inquiries coming into our offices, we sought out both Dale Osborne, head of TI's Educational Products Division, and Ken Modesitt, Manager of TI's Computer Based Learning Consumer Group. Both gentlemen assured us that TI remains committed to and will bring to market all 108 PLATO program packages as originally scheduled.

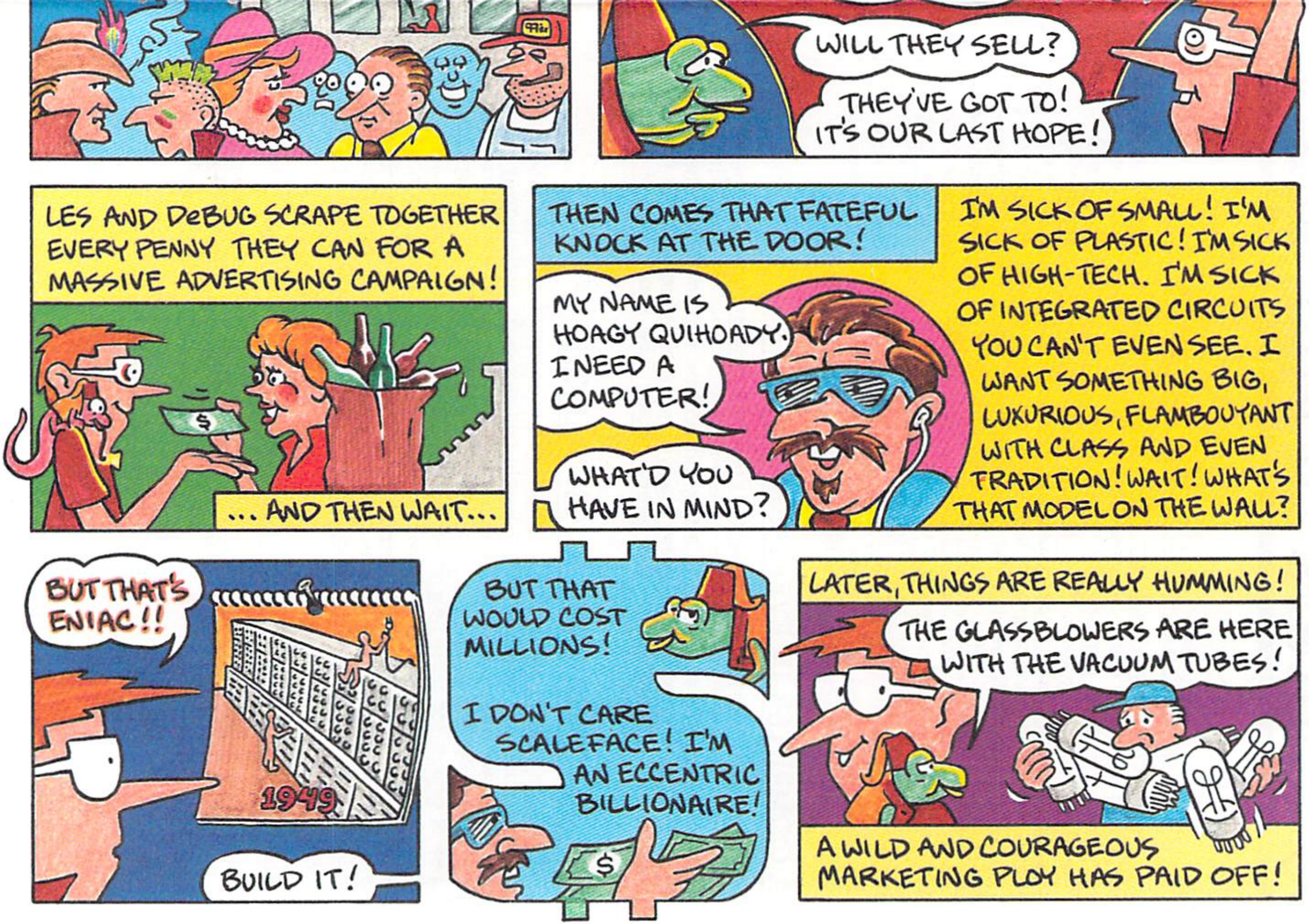
The ambitious educational software library for the TI computers grew out of a long collaboration between the University of Illinois and Control Data Corporation. TI bought the field-tested but text-heavy and expensive-to-run Illinois-Control Data educational packages (known as CENTRAL PLATO) in order to format them to run on an inexpensive computer. Although early TI PLATO releases relied heavily on text, their computer format opened up the PLATO library to a host of users who could now bypass the high telephone communication costs that had confined CENTRAL PLATO sales to a well-endowed institutional market. More recent TI PLATO releases redress the text imbalance of the earlier educational packages and capitalize on the full capabilities (including graphics and sound) of the TI-99/4A computer.

School districts and individuals can rest assured that in addition to the *Basic Skills* and *High School Skills* packages already available for the 99/4A, there will soon be a full 108 TI PLATO packages reaching from the third grade through the university level. Of special note in the upcoming TI PLATO packages is an in-depth science series (the Chemistry package alone consists of 7 double-sided floppy disks). At the high school level, packages in social studies are promised. *Basic Reading, Basic Math, Basic Skills Grammar, High School Writing*, poetry, and drama packages are among those scheduled for an early first-quarter release.

TI plans to prepare a not-too-modest 500 sets of each of the first 80 PLATO packages and will make more available when those are sold.

—Erin O'Connor


Les Izmore and DeBug
BY LAREDO & ROBERTS
©1983 EMERALD VALLEY PUBLISHING Co.
LES AND DeBUG STILL HAVEN'T FOUND THE MARKETING SCAM THAT WILL MAKE CHAMELEON COMPUTER & PETFOOD Co. #1.
WHAT'S THAT IN MY BREAKFAST?
FROOT FLIES
FREE
COMPUTER PART INSIDE
SOME COMPANIES WILL IMPLEMENT ANYTHING!
BUT EVERY MARKETING PRO WE'VE HIRED HAS BEEN A DUD!
FROOT FLIES
SOME CONSULTANTS WERE STILL PUSHING A GAMES EMPHASIS WHEN THAT MARKET HAD ALREADY LEVELED OFF!
AWESOME!
WE SPENT A FORTUNE GETTING IMPORTANT PEOPLE TO ENDORSE OUR PRODUCTS, BUT FEW CONSUMERS WERE IMPRESSED
WHOOZAT?
WE TRIED TAKING OUR PRODUCT TO FAIRS, BUT NOBODY WANTED TO TALK DATA. THEY ALL JUST WANTED TO RIDE THE OCTOPUS.
PROBLEM IS, LES, NOBODY KNOWS WHO THE AVERAGE CONSUMER IS. LOOK AT ALL THOSE FOLKS OUT THERE. COULD YOU SAY ANY OF THEM ARE AVERAGE?
I HAVE IT! DELETE ALL OLD DATA! FORGET CONSULTANTS AND MARKETEERS. WE'RE GOING TO TRY SOMETHING RADICAL — CUSTOM BUILT MACHINES — ORIGINALS TAILORED TO THE INDIVIDUAL!


WILL THEY SELL?
THEY'VE GOT TO! IT'S OUR LAST HOPE!
LES AND DeBUG SCRAPE TOGETHER EVERY PENNY THEY CAN FOR A MASSIVE ADVERTISING CAMPAIGN!
... AND THEN WAIT...
THEN COMES THAT FATEFUL KNOCK AT THE DOOR!
MY NAME IS HOAGY QUIHOADY. I NEED A COMPUTER!
WHAT'D YOU HAVE IN MIND?
I'M SICK OF SMALL! I'M SICK OF PLASTIC! I'M SICK OF HIGH-TECH. I'M SICK OF INTEGRATED CIRCUITS YOU CAN'T EVEN SEE. I WANT SOMETHING BIG, LUXURIOUS, FLAMBOYANT WITH CLASS AND EVEN TRADITION! WAIT! WHAT'S THAT MODEL ON THE WALL?
BUT THAT'S ENIAC!!
1949
BUILD IT!
BUT THAT WOULD COST MILLIONS!
I DON'T CARE SCALEFACE! I'M AN ECCENTRIC BILLIONAIRE!
LATER, THINGS ARE REALLY HUMMING!
THE GLASSBLOWERS ARE HERE WITH THE VACUUM TUBES!
A WILD AND COURAGEOUS MARKETING PLOY HAS PAID OFF!

Computers in Education

SCHOOL MARKET WIDE OPEN (BRIEFLY)

Schools used to conservatively buy Apples or frugally go with the TRS-80. Nowadays, when it comes time to buy micros for computer literacy instruction, popular choices include Atari, Commodore, and Franklin home computers. And the entry of the IBM PCjr may further change the face of the market. Despite the drawbacks of the PCjr's infrared keyboard (inappropriate for school settings where the presence of other computers will interfere with its operation), it seems likely that schools will pay the extra $20 for a keyboard cable in a tradeoff for the "serious computing" and "reliable" reputation of IBM. Companies such as The Learning Company (TLC) are going all out to alleviate the lack of PCjr-compatible educational software by September. The cost of the PCjr may be an inhibiting factor, though. School budgets have shrunk in recent years, and schools are under even more pressure than ever to seek out heavy discounting for their large volume purchases.

A COMPUTER IN EVERY CLASSROOM

In a recent message to educators, President Reagan gave hardware manufacturers more incentive to go after the education market. The proverbial promise of a chicken in every pot will be replaced with the equivalent of a computer in every classroom, in keeping with the Chief Executive's recommendation that every student take a half year of computer science.

BIG CATCH IN EDUCATION MARKET

Apple and IBM may be butting heads in a competition to get the most hardware into the schools. While Apple has stopped "giving away" its machines, we expect to see this company cast out some new bait to lure this profitable fish away from Big Blue. Meanwhile, IBM sits smugly on piles of orders for their new PCjr (sight unseen!) from educational institutions. Neither company is talking strategy yet, but buzzings have been heard that one or both may offer sizeable scholarships and grants to schools stocked with their machines. We look forward to counting the orders again in September when the real angler comes forward to claim the prize.

SOUNDS OF MUSIC AT CES

The show was alive with the sounds of music coming from Syntauri Corporation's portion of the Apple Computer Exhibit. From the joystick-controlled *Musicland*, to *Simply Music* (a total keyboard learning and performance system) crowds at the Winter Consumer Electronics Show gathered to watch as Apples were transformed into interactive instruments. Professional-sounding results came from the tuned-in and the tone-deaf alike, and everyone seemed to learn something from these newly-announced packages.

New Tech News

RELIABLE STORAGE FOR THE ADAM

Although Coleco claims that "Adam includes all the hardware and software necessary for immediate use in the home," reports of data losses from their digital tape drive cartridges indicate that long-term reliable data storage remains something of a problem. It seems that inadequate shielding around the tape cartridge and a "residual electromagnetic charge" that occurs when the tape drive is turned on can make the tapes temporarily or permanently unusable. This, coupled with the difficulty of producing prerecorded tapes in volume, spells trouble. Coleco's introduction of an optional 5¼" floppy disk drive at January's CES could signal that the end is near for the digital tape drive. The new disk drive—scheduled for release in second quarter, 1984—will be available for "under $400." Let's hope that this is all the hardware Adam users will need.

AN A.T.&T. PC?

Though it's keeping developments under wraps, A.T.&T. is rumored to be close to releasing their own home computer. Would you believe a 64K unit complete with 1 disk drive and monitor for less than $400? Ma Bell isn't making any public statements, but news of the product is leaking out slowly but surely.

KOALA WIDENS FOCUS

Koala technologies, fresh from their great Koala Pad success, will soon market a device known as the Gibson Light Pen. Designed for use with the Apple II series, IBM PC, IBM PCjr, and Commodore computers, it is expected to retail for under $300. The pen, the interface card, and software allowing the user to create and store high-resolution shapes and designs are included in the package. Properly interfaced, the pen can also be used as a pointing device in lieu of a desk-top mouse. The pen should allow the user to choose and manipulate programs as well as data displayed on the screen in a window environment.

THE KODAK DISK—BUT NOT A CAMERA

Yes, Kodak has gone into the disk drive business, and oh, what a disk drive. The 3.3 Flexible 5¼" Disk Drive—named for its 3.3 MByte capacity (2.62 MByte formated)—boasts a transfer rate of 500 Kbits/sec. (twice the IBM PC's rate) and a track-to-track access time of 3 msec. (half the IBM PC's time).

Kodak also claims the drive is downward-compatible to read conventional 5¼" diskettes because its microprocessor logic detects a conventional disk inserted in the drive and automatically adjusts motor speed to the conventional rate. At CES they had the drive interfaced to a PC via a modified 8" controller using a third-party operating system called DOS-2.08. The drive, produced by Kodak, will be marketed by Data Technology Corporation at a retail price of $495. Release is planned for third quarter, 1984.

Remembering Kodak's marketing ploy—selling cameras to sell their film—it should come as no surprise that Spin Physics of San Diego, is making special pre-formated diskettes for Kodak to market at an anticipated price of $12-$15 each. Also not surprising is that Kodak is the parent company of Spin Physics.

Industry Watch

COMPAQ TRIO LICENSED—OTHERS IN TRIPLE TROUBLE?

Now that they've gone out of the home computer business, Texas Instruments is catching up with encroachments on its always closely-held technology. The three former Texas Instruments employees who founded Compaq some two years ago recently agreed to purchase licenses from TI for personal computer technology used in Compaq's portable microcomputer. Encouraged by this settlement, TI is pushing company patent attorneys to get other personal computer manufacturers to honor three of TI's patents for widely used computer technology—2 bit-pusher patents and a patent to protect TI's combining of manual input device, single-chip personal computer, and screen display. Analysts inside TI insist that many computer manufacturers violate these patents and will be affected by TI's decision to "collect" on their earlier contributions to microprocessor circuitry.

COMPANIES STAY AWAY, OR GO BUT DON'T SHOW

It used to be that a company's appearance at COMDEX or CES signalled the industry that the company was doing all right. Failure to appear was sure to provoke rumors that the company was in trouble. But lately, with the proliferation of computer expositions threatening a schedule of one every six weeks, some companies, (confident that their absence won't be misinterpreted) are being selective about the shows their money and energy go into. Sirius, Sierra On-Line, and Broderbund, for instance, decided this year to relinquish the pleasures of their regular January CES floor exhibit space in Las Vegas in anticipation of a more productive sales effort at CES in June. Others came only to see and be seen.

TI USERS, TAKE HEART

TI will continue to support its huge Home Computer user base. March Direct Marketing (MDM) has been contracted by TI to keep users apprised of third-party hardware and software, via a quarterly catalog. TI will fully honor its warranties and maintain out-of-warranty service in addition to its helpline,800-TI-CARES. GROM licenses for TI's auto-incrementing memory feature are in the works, as are agreements for selling GROMs to interested third-party software manufacturers. TI is also negotiating with third parties interested in manufacturing 99/4A software formerly produced by TI. So far, both Sierra On-Line and Imagic have agreed to "take back" and distribute the TI software. Sierra On-Line will also manufacture and market 5 educational programs developed by Walt Disney Studios for TI, and Imagic will manufacture and market 5 of its programs for which TI had previously purchased rights.

Toward an Industry Standard

While American computer devotees curse software incompatibilities, the Japanese are doing something about the situation. No fewer than a dozen Japanese firms (including Matsushita, Mitsubishi, Toshiba, Sanyo, and Yamaha) are releasing new MSX-standard personal computers. These machines are hardware compatible, using a Z-80A microprocessor, a TI video chip (the same one as in the 99/4A), and identical sound generators. They are software compatible too, and the MSX standard designation comes from their uniform ability to run Microsoft Super Extended BASIC.

The only American companies to jump on the MSX bandwagon thus far are Spectravideo and ROMOX. Spectravideo will release an MSX-compatible model, the SVI-728, sometime in the first quarter of 1984. Although they have not settled on a price yet, reports place it within the under $500 range. The computer will feature a keyboard with 87 keys (including a numeric key-pad and 10 user-programmable function keys), 32K of ROM (which contains MSX-BASIC), and 80K of RAM. It will have CP/M capability and be expandable to 96K of ROM and 144K of RAM. This should be strong competition in the low-end home computer market.

ROMOX feels so far that the MSX is most appropriate for the Japanese market, but hopes to produce software for the MSX system. ROMOX already has arrangements with Aster Co. in Japan for its electronic software distribution system, so this would seem like a natural.

Why such a lack of American interest in the MSX? It's too late: The U.S. home computer market is already bursting with competition. The 8088 microprocessor is used in so many machines now that a Z-80A-based machine might be an out-of-date "standard" in the American market. Right now the PC-compatible computer is the de facto American "standard," although discussions of a U.S. standard these days must include the possibility of an A.T.&T. Unix-based home computer. Mysterious TV commercials featuring the undisclosed contents of a home-computer-sized box identified as "the future" add fuel to expectations that A.T.&T. will exercise its considerable clout to establish a Unix-compatible standard in our American marketplace.

But we Americans don't have the added push for standardization that accompanies language considerations in Japan. High level computer languages (like BASIC) are written in English, so for the Japanese to work in their own language, an extra translation step is necessary. Dealing with a single BASIC (like MSX) instead of several incompatible BASICs greatly reduces these translation difficulties.

— Roger Wood

Gameware Updates

WELCOME ABOARD, PAC-MAN?

The reason your flight is off-course may not be hijackers. Recently, intermittent radio interference from a passenger's Pac-Man game shut down an engine on an Eastern Airlines plane.The Radio Technical Commission for Aeronautics (RTCA) will investigate the problem of portable computers interfering with aircraft navigational systems.

EN GARDE!

Atari and Coleco are battling it out with Wico, Discwasher, and Zircon International for the best joystick at the lowest price. Although the Atari 2600 joystick is a traditional best-seller at $9-$11, short supplies have led many gamesters to make other selections from the myriad available in the $10-$50 range. Beefed-up ad campaigns, like Wico's pre-Christmas debut on national television, are expected as the contenders take jabs at Atari's 4-1 sales lead.

TRIAL OFFER

A new marketing approach by Epyx offers a "Preview Disk" that allows consumers to sample excerpts from five games (*Gateway to Apshai, Jumpman Junior, Pitstop, Seawolf,* and *Gunfight*). Epyx has mailed out a quarter-million flyers advertising the sample disks to users who can order one for $2.50, which is refundable with proof of purchase from any of the games.

SEND IN THE CLOWNS

Although games might amount to only half of a software publisher's sales, many feel that popular game titles are important in establishing brand recognition. Advertising for entertainment-oriented software can help lure consumers towards more serious applications packages such as education and home management. The games themselves can be constructed as tutorials that introduce gamesters to other areas of computer use. Watch for a glut of new game titles during the first half of 1984, and new lines of "hybrid" software that mix business with pleasure.

FROM SILVER SCREEN TO MONITOR

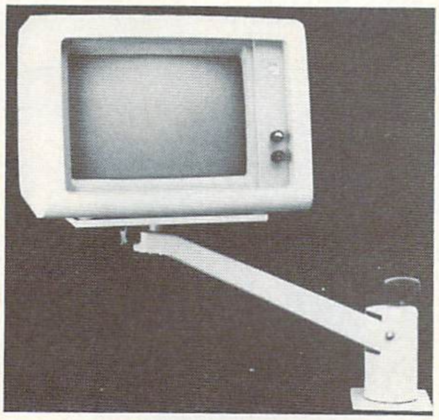
The trend in gameware for '84 is movie titles. Box office hits *Krull* and *Star Trek* are slated for release in adventure game versions, and a new game features Kung Fu fighter Bruce Lee. Even Buck Rogers will enter the 21st century immortalized in a video game.

HOME COMPUTER product news

Each month we publish items of interest and news of recently or soon-to-be released computer products. Our publication of information from manufacturers of computers, peripherals, software, and accessories is not to be construed as product endorsement. Prices quoted are the manufacturers' suggested retail prices and are subject to change.

Send press releases to:

Product News Editor
Home Computer Magazine
1500 Valley River Dr., Suite 250
Eugene, OR 97401



CRT SWIVEL ARM

Lintek, Inc., has announced its Monitor Mover, a desk-mounted, adjustable mechanical arm to hold CRTs (including IBM, Apple, TI, BMC, Princeton, Amdek, and Zenith) up and off the user's desk. Of all-steel construction, the Monitor Mover consists of heavy-duty desk clamp, 360° swivel base, 15" vertically adjustable arm, and individualized swivel CRT mounting tray that can be tilted up to 15°. Optional extended monitor cables and two other styles of desk mounts are also available. The Monitor Mover retails for $129.95.

Lintek, Inc.
P.O. Box 8056
Grand Rapids, MI 49508
(616) 241-4040.

TWO NEW TI GAMES RELEASED

Texas Instruments has announced new game software for the TI-99/4A. *Slymoids,* a one-player, five-level space invaders shoot-out, employs a scanner and a laser fireball weapon to locate and destroy two varieties of Slymoids. Joysticks are recommended. *Return to Pirate's Isle,* the fourteenth in a series of Scott Adams adventures for the TI-99/4A, features both multi-screen graphics and text that require users to decipher clues to solve the mystery. A cassette or disk storage device is recommended. Each game retails for $39.95

Texas Instruments
Consumer Relations
P.O. Box 53
Lubbock, TX 79408
(800) TI-CARES.

INSTRUCTIONAL GAME FOR BUSINESS APPLICATIONS

Knoware, Inc., has announced the availability of *Knoware* for the IBM PC, Apple IIe and II+ computers. *Knoware* is a software package directed at teaching business men and women the fundamentals of personal computing. Using a game format, users follow a career path from mail clerk to chairman of the board while learning and practicing computer applications. The applications programs which come with the software package include *Spreadsheet, Database Manager, Text Editor, Individual Retirement Account Planner, Financial Decision Support, Calendar, Pie and Bar Graphs, and Metric to English Conversion. Knoware* retails for $95.00.

Knoware, Inc.
301 Vassar St.
Cambridge, MA 02139
(617) 576-3821.

EIGHT-WAY JOYSTICK

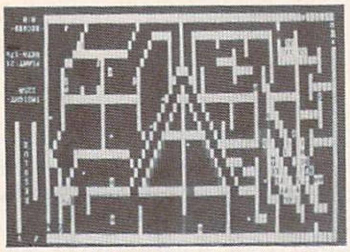
Personal Peripherals, Inc., has introduced the Super Stik(tm), an 8-way joystick controller featuring an extra-long five-foot cord, a minimum of moving parts, and a wiping action to assure no oxide build-up or corrosion of contacts. Model J1000, compatible with the Commodore 64 and VIC-20 computers, the Atari 2600 Video Game and 400/800 computers, and the Sears Video Game, retails for $9.95. For the TI-99/4A, Super Stik model J1020, packaged in pairs with common cable, is available at $24.95.

Personal Peripherals, Inc.
P.O. Box 154249
Irving, TX 75061.

PERSONALIZED GREETING CARDS FOR YOUR WORD PROCESSOR

Compucards has announced greeting cards with continuous tab feed and clean-edge perforation. The cards are designed so that a word processor can print a personal message inside. Envelopes are attached to fan-fold paper for continuous tractor-feed addressing. The initial Christmas series will be followed in February by birthday and all-occasion cards suitable for weddings, thank-you notes, birth announcements, and general invitations. The price for a package of 20 cards and envelopes is $9.95. A box of 100 cards and envelopes retails for $45.00, and a box of 300 cards and envelopes retails for $130.00.

Compucards
P.O. Box 894
Stone Mountain, GA 30086.



EVEN THE MOST RELUCTANT CAN BE PERSUADED TO PLAY

Olorin Games has announced the release of *Persuasion*, a fast-action strategy game that will run on all IBM PC computers (64K memory and disk drive required). The game offers an alternative to the themes of killing, destroying, eating dots or chasing aardvarks. Fast-action graphics, multiple difficulty levels and variable speeds make it suitable for players of all skill levels. It can be played with joysticks or keyboard, with monochrome or color display.

Olorin Games
P.O. Box 719
Amherst, MA 01004
(413) 549-4786

INTEGRATED MONEY MANAGEMENT PROGRAMS

Sundex Software Corporation has announced three integrated money management programs for the TI PC, IBM PC, and Apple II and IIe computers, with plans to offer the package for the IBM PCjr and other home computers in early 1984. *Personal Payables*(tm), the bill-paying program retailing at $49.95, handles up to 10 separate checking and saving accounts and can store over 1,000 transactions. Accompanied by Sundex's check-handling holster and a printer, the program can print on the user's own personalized checks or any type of continuous form checks. The program produces financial reports in any combination of check, date, payee, or tax implication. The *Certified Personal Accountant*(tm) program at $99.95 will pay bills and prepare budgets and reports on net worth and cash flow. It will produce income and expense statements and tax information. The *Certified Personal Investor*(tm) at $149.95 is designed for personal portfolio management and analysis and tax form preparation. The program will track up to 150 investments and produce information for the 1040B interest reporting and 1040D capital gains forms.

Sundex Software Corporation
Boulder, CO
(303) 440-3600

VDT EXPANSION BOARD & PSIO CARD THAT REMEMBERS

Videx, Inc.(tm), has released the UltraTerm(tm), an expansion board for video display to augment the power of *VisiCalc*(tm) and word processing for the Apple IIe, Apple II+, Apple III, and Franklin computers. The board features eight software-selectable modes and makes possible screen displays of as many as 160 columns by 24 lines. It is compatible with BASIC, Pascal, and CP/M. The Apple III monitor or the Amdek Video 310A amber monitor, or any CRT with a high- or medium-persistence phosphor is recommended. The UltraTerm package retails for $379.00 and includes board, utilities disk, firmware, and manual. *VisiCalc* applications of UltraTerm require a pre-boot package at $69.00, and *Applewriter II* requires a pre-boot package at $29.00.

Videx has also announced a PSIO Dual Function Interface Card for the Apple II, Apple IIe, Apple III, and Franklin computers. The PSIO card allows simultaneous use of printer (parallel output) and modem (serial I/O port) with one card. The PSIO card will work with any printer/modem and is compatible with BASIC, Pascal, and CP/M systems. It uses a Non-Volatile Random Access Memory (NOVRAM) to maintain configuration options. The PSIO card retails for $229.00.

Videx, Inc.
897 NW Grant Ave.
Corvallis, OR 97330
(503) 758-0521.

PRINTING SERVICE FOR TI USERS

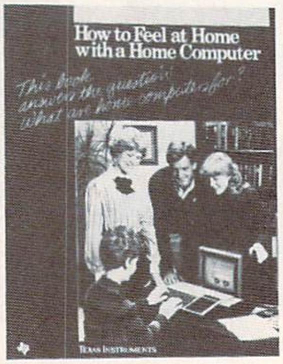
The Micros' Ink is announcing a printing service for 99/4 and 99/4A users. The service will list programs, print word processing, and plot black and white graphics from cassette or disk. For users without a printer, the service provides hard copies of programs. For users with thermal or other 40-column printers, the service provides 80-column dot matrix copy. Users may specify data processing, emphasized, enhanced, or letter-quality printing. The Micros' Ink has also developed a word processor for TI BASIC users. The user can type text on the TV or monitor screen and correct it. Cassette or disk can then be sent to Micros' Ink for printing. The introductory price for listing programs of less than 16K, printing word processing of less than 10 pages, or plotting one black and white graph is $5.00. Reduced rates are offered for multiple copies, additional pages, or graphics.

The Micros' Ink
P.O. Box 3725
Alliance, OH 44601.

ALGEBRA SELF-TAUGHT WITH NEW SOFTWARE

Eduware has announced *Algebra Volume 2*, the second part of their six-volume series for the IBM PC. The series comprises a first-year course taught in a flexible style which utilizes high-resolution color-coded maps to chart progress. Volumes 3-6 of the IBM version are scheduled for later release, and the entire series is currently available for the Apple II. The 5 1/4" floppy disks require 64K of user memory with IBM DOS 1.1 and 128K with IBM DOS 2.0. The price is $39.95.

Eduware Services, Inc.
28035 Dorothy Dr.
Agoura Hills, CA 91301



"HOW TO" BOOK FOR HOME COMPUTERS

Texas Instruments, Inc., has recently announced the publication of *How to Feel at Home with a Home Computer* by Gary G. Bitter and Roger S. Walker. The illustrated 264-page book contains step-by-step instructions, and guide maps begin and end each of the nine chapters. In addition to "Getting Acquainted" information, the book contains instruction in creating programs for education, information management, entertainment, speech, music, and graphics. The book retails for $12.95.

Texas Instruments, Inc.
P.O. Box 3640 M/S 54
Dallas, TX 75285

HOME COMPUTER product news

EFFECTIVE RESULTS AT YOUR COMMAND

Voice Machine Communications has announced the Voice Input Module, a speech recognition system designed to meet educational, professional, handicapped, and industrial needs in a wide variety of applications. The module makes it possible to bypass or augment keyboard operations, and eliminates the need to learn specific keyboard commands. The VIM runs with any commercial software package, and is currently available for the Apple II, IIe, and Franklin type computers. The VIM-1 costs $845.00 for the Apple II and II+, and $920.00 for the IIe. The VIM-2 with deluxe microphone assembly sells for $920.00 for the Apple II and II+, and $995 for the IIe version.

Voice Machine Communications, Inc.
1000 South Grand Ave.
Santa Ana, CA 92705
(714) 541-0454

REAL-LIFE ACTION IN NEW GAMES

Tronix Publishing has released four new games for the Commodore 64 that present players with challenging situations designed with realism in mind. *Motocross* and *Slalom* are aimed at aficionados of those sports, and were developed in consultation with professional athletes. *Waterline* and *Suicide Strike* are adventure games that demand quick decisions and complex strategy. The games are priced at $34.95 on disk, and $39.95 for the ROM cartridge versions that will be available soon.

Tronix Publishing, Inc.
8295 S. La Cienega Blvd.
Inglewood, CA 90301
(213) 215-0529

HELP FOR THE COLLEGE BOUND

CBS Software has announced a learning aid designed to provide individual review for students preparing for college admissions exams. *Mastering the College Board Achievement Tests: English Composition* contains over 1,000 practice problems, and provides an approximate score and error analysis to indicate areas for further study. The software is available for the IBM PC and Apple for $175.00.

CBS Software
One Fawcett Place
Greenwich, CT 06836

JUMP INTO THE FUN

Screenplay announces the release of *Pogo Joe*, a game with animated three-dimensional graphics, music, and 64 different screens. The player guides Pogo on a dangerous mission with enemy monsters in pursuit. *Pogo Joe* is available in 48-64K for the Commodore 64 and Atari at $24.95.

Screenplay
500 Eastowne Office Park
Suite 212
Chapel Hill, NC 27514

FOR PROSPECTIVE PEANUT USERS

Howard W. Sams & Co. has announced the publication of an illustrated 160-page book entitled *Introducing the IBM PCjr*, which offers an in-depth look at IBM's new home computer. The book describes set-up, available software and all of the configurations and options available for both the basic and enhanced systems. Four appendices cover infrared communications, the serial connection, protocol, and control codes, and there is a glossary for first-time users. The price is $12.95.

Howard W. Sams & Company, Inc.
4300 West 62nd St.
Indianapolis, IN 46268
(317) 298-5400

HOW TO PROFIT

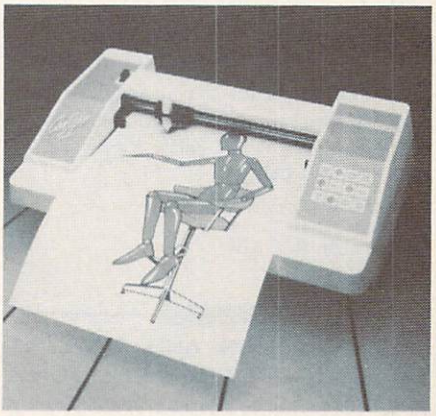
Edge Press has announced the January, 1984, publication of *Profit from the IBM PC* by Dan W. Post. The book is designed to assist PC users in creating and selling freelance services in such areas as tutoring, editing documentation, writing magazine articles, giving group workshops, consulting, leading seminars, publishing a newsletter, programming and marketing vertical software, and writing books. There are sections on networking and communications, rights and royalties, exploiting trade shows, audiovisual training tools, market research, application planning, choosing a high-level language, publishing, distribution, and program protection. Organization, planning, and time management are discussed, as are record-keeping and tax considerations. Product directories complement each section. The 224-page hardbound book retails for $17.95.

Edge Press
P.O.Box 150
Arcadia, CA 91006

NEW PORTABLE TYPEWRITER DOUBLES AS COMPUTER PRINTER

Smith-Corona has introduced an electronic portable typewriter that plugs into an optional Messenger Module to become a letter-quality computer printer. The Ultasonic III Messenger offers electronic features such as full-line memory correction, triple pitch (10, 12, or 15 characters per inch) automatic underlining, and automatic centering. With the Memory Module, the typewriter will serve as a printer compatible with virtually all personal, home, and small business computers. The typewriter retails for $635.00, and the module for $170.00.

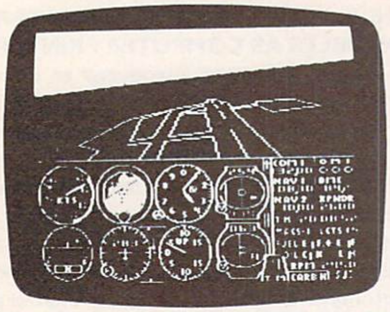
Smith-Corona
65 Locust Ave
New Canaan CT 06840
(203) 972-1471



PERSONAL GRAPHICS PLOTTER

Comrex is offering the Model CR-1810 ComScriber I plotter for the IBM PC, Apple, Epson QX-10, and most other personal computer systems. The ComScriber I will plot graphics displays on either standard paper or transparencies. Line segments are accurate to one four-thousandths of an inch, and the plotter draws at a speed of up to six inches per second. The Comscriber I will use either manufacturer-supplied or commercially available standard pens, paper, and transparencies. The basic plotting area of 8-1/2 inches by 11 inches can be extended up to 120 inches. Characters from 1/8 inch to 20 inches high can be drawn. Movements of the plotter are controlled on a 12-key keyboard. The Comscriber I retails for $695.00, and a "support pack" of pens, paper, interface cable, and tutorial software for specific computers is also available.

Comrex International, Inc.
3701 Skypark Dr.
Torrance, Ca 90505
(213) 373-0280



NEW FLIGHT SIMULATOR LANDS ON HOME COMPUTER MARKET

SubLOGIC Corporation has announced *Flight Simulator II* on diskette for the Apple II, Commodore 64, and Atari (48K minimum memory required). The user can practice take-offs and landings at over 80 airports with full flight instrumentation (avionics included) in a Piper 181 Cherokee Archer. The program comes with four scenery areas—New York, Chicago, Seattle, and Los Angeles—and additional scenery areas are available. Weather conditons are user-adjustable, and the program provides for aerobatics and includes a WWI aerial battle game. The price is $49.95.

SubLOGIC Corporation
713 Edgebrook Drive
Champaign, IL 61820
(217) 359-8482
Telex 206995

SURVEY DESIGN AND ANALYSIS

Telofacts 1 and *2* for the IBM PC has been added to the dilithium Press catalog of computer books and software. The package provides the capability to design and automate questionnaires, tests, polls and other forms. It can also be used to gather opinions, resume information, marketing data, and survey data. The software also provides analysis capability. *Telofacts* runs on the IBM PC or IBM PC XT, with a UCSD p-system, with 128K, two disk drives, and monochrome adapter or color graphics adapter with parallel printer. *Telofacts 1* retails for $49.95. *Telofacts 2*, the enhanced version that can be used with a card reader to rank, list, and score respondents, retails for $199.95. The *Telofacts* packages are also available at the same prices for the Apple II or IIe with 64K and one disk drive. 80-column screen card, second disk drive, and printer are recommended.

How to Use the Peanut for IBM PCjr users has been scheduled for January, 1984, publication.

dilithium Press
8285 SW Nimbus, Suite 151
Beaverton, OR 97005
(800) 547-1842

COMPILER FOR TI-BASIC

SST Software, Inc., has announced the release of its *BASIC Compiler Package* for the TI-99/4A. The compiler translates BASIC programs into TMS9900 machine language for speedier execution. TI console, cassette recorder and cable, and Mini Memory are required. The compiler can write and compile 150 lines of BASIC using the minimal configuration, and 500 lines of BASIC with Memory Expansion with or without disk drive. It can dimension up to 1800-element floating-point arrays with minimal configuration and up to 12,000-element integer arrays using Memory Expansion. It provides for the writing and debugging of BASIC programs using the TI console and interpreter, and then compiles without the need for retyping. *The BASIC Compiler Package* retails for $50.00.

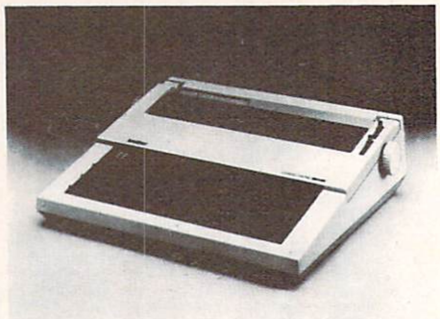
SST Software, Inc.
Box 26
Cedarburg, WI 53012.

PRODUCTIVITY & GAME SOFTWARE FOR TI-99/4A

Konrad Komputerware is announcing a new line of cassette software for the TI-99/4A. Initial offerings include low-cost packages containing games and home finance aids.

Konrad's *Kasino* is an assortment of gambling games, including Blackjack, Video Poker, Craps, and HiLo, in both BASIC and Extended BASIC. *El Hango*, a Spanish version of Hangman, is available in Extended BASIC for vocabulary practice. Each package retails at $9.95. *IQ Test* for anyone over 12 is available at $7.95. Konrad also offers *Fun with Money* in BASIC and Extended BASIC—a home financial decision-making aid for checkbook, loans, mortgages, investments, periodic savings, and future values. It retails for $9.95.

Konrad Komputerware
P.O.Box 26741
Fort Worth, TX 76126.



ELECTRONIC TYPEWRITERS DOUBLE AS PRINTING TERMINALS

Brother International Corporation has introduced the first of their series of electronic compact typewriters that can double as printer terminals by interfacing with a computer using the built-in interface port. The Correctronic 50 offers triple pitch selection, a full one-line correction memory, interchangeable cassette daisy wheel, and cassette ribbon system. The typewriter comes with a built-in carrying case. The Correctronic 50 is priced at $499.95.

Brother International Corporation
8 Corporate Place
Piscataway, NJ 08854.

ELECTRONIC ROAD ATLAS

Columbia Software has introduced *Roadsearch-Plus*, a computerized road atlas available on disk for the Apple II and IIe computers with DOS 3.3. *Roadsearch-Plus* can determine and print the shortest practical route between cities, helps you avoid toll or other specified kinds of roads, and find feature-specific routes. Its database of 406 cities/roads intersections and 70,000 miles of interstate and major through highways can be updated and revised with the user's shortcuts, local roads, and new destinations. Printouts will provide route, distances, travel times, and fuel usage. *Roadsearch-Plus* retails for $74.95.

Columbia Software
5461 Marsh Hawk
Columbia, MD 21045
(301) 997-3100

GAMES FOR THE ARMCHAIR TRAVELER & HOME SLEUTH

Available for the TI-99/4A and Commodore 64 from Briley Software are five new programs in BASIC on cassette. The Explorer Series includes two new variable text adventures, *High Seas* and *Fur Trapper*. The Detective Series offers three games of deduction and logic with maps and changing solutions: *Mansion, Pentagon,* and *Museum*. Each game is priced at $14.95

Briley Software
Box 2913
Livermore, CA 94550.

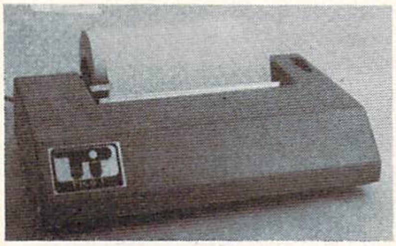
SAKATA MONITORS & STAND

SAKATA U.S.A. Corporation has introduced its SC-100 CRT Composite Color Display Monitor with controls including phone pin jack, sound, contrast, power switch, vertical hold, color, tint, and brightness. The 13" monitor, which weighs 30.8 lbs., can display up to 1,000 characters and retails for $329.00. It is compatible with the Apple II, Atari 800, Commodore 64 and VIC-20, IBM PC, TI-99/4A, and Osborne computers.

For those computers and the Apple III and NEC PC as well, SAKATA has introduced its SG-1000 Monochrome Monitor. The high-resolution CRT is phosphor green with a non-glare high contrast dark face plate. With a 12" screen, the unit weighs 16.5 lbs. and retails for $129.00.

SAKATA has also announced a CRT stand for its color display monitors. It tilts up and down and swivels left and right and to 90°. Made of polystyrene in a neutral color, the monitor stand retails for $49.00.

SAKATA U.S.A. Corporation
651 Bonnie Lane
Elk Grove Village, IL 60007
(800) 323-6647



PRINTER AND INTERFACE BRING DOWN THE COST OF EXPANSION

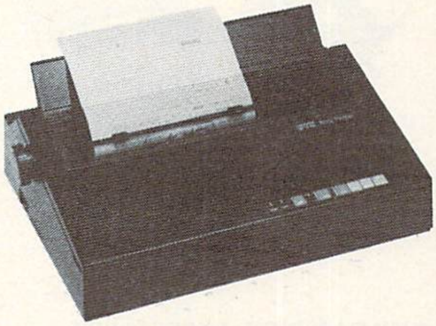
Pueblo Instruments has introduced the PICO-1(tm), a low-cost dot matrix printer compatible with the Apple, Atari, TRS-80, and TI-99/4A. The draft-quality, 7-pin, 80 cps printer is available in beige or grey, and can be ordered with an adapter and cable that allow direct hook-up. A unique interface board, the TIPI(tm) (Texas Instruments Printer Interface), gives TI users printing capabilities without the expense of the TI peripheral expansion system and RS232 card. The PICO-1 printer retails for $330, and the TIPI adapter costs $65.

Pueblo Instruments, Inc.
P.O. Box 3367
Pueblo, CO 81005
(303) 544-7700

"MISSING LINK" INTERFACE

Midwest Engineering Consultants has announced a 200 cps printer adapter called "The Missing Link" that uses the TI joystick port and requires only cassette tape and TI Mini Memory or Extended BASIC cartridge. The adapter package is priced at $36.95. The package includes 35 pages of documentation, fully assembled interface, and nine Mini Memory and Extended BASIC programs, including Mini Memory utilities and Extended BASIC engineering programs:*Ladder Network Analysis*, *Home Security Program*, and *Sprite Program*. Also included are four electrical engineering programs.

Midwest Engineering Consultants
P.O.Box 159
Hawthorn Center
Vernon Hills, IL 60061.



SELF-DIAGNOSTIC PRINTER

Data Terminals and Communications has just announced the DTC Style Writer, a daisy wheel printer with 35K buffer memory that permits simultaneous use of printer and other computer applications. The Style Writer, designed for use with most major personal computers, including IBM, Apple, and TRS-80, features multicopy capability, full bi-directional printing, automatic proportional spacing, standard Centronics parallel interface, graphics plotting, two-color printing, and a momentary pause for paper, print wheel, and ribbon changing. The print wheel is available in 17 different type fonts. A self-test diagnostic routine that operates independent of the computer will evaluate the printer's internal electronic circuits and print mechanism. Error conditions are indicated by LED lights. The DTC Style Writer retails for $899.00.

Data Terminals and Communications
590 Division Street
Campbell CA 95008
(800) 962-8185

NEW SOFTWARE FOR BUDDING ARTISTS

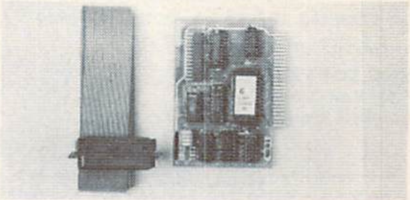
Scarborough Systems has announced a new line of software to encourage young children to experiment with art and music. *PICTUREWRITER* makes it easy to draw and "paint" on the screen using keyboard and joysticks. *SONGWRITER* brings composition skills to the novice and is versatile enough to accommodate the experienced musician. Both of these Art Series programs are available for Apple users for $39.95 each, with versions for the IBM PC, Commodore 64, and Atari in the making.

Scarborough Systems, Inc.
25 North Broadway
Tarrytown, NY 10591
(800) 882-8222

QUICK DISK COPIER

Quality Software has announced the *QUICK-COPYer*(tm), a program for backing up disks on the TI-99/4A. Designed to be faster than the Disk Manager cartridge, the *QUICK-COPYer* will back up a disk in three passes or less and back up a double-sided disk in six passes or less. The *QUICK COPYer* requires the TI-99/4A, disk drive, 32K Memory Expansion, and either the Extended BASIC, Mini Memory, or Editor-Assembler cartridge.The price is $39.95.

Quality Software
1884 Columbia Rd., #500
Washington, D.C. 20009
(202) 667-3574



WHAT YOU SEE IS WHAT YOU GET WITH PRINT-IT!

Texprint has announced a new printer interface card with 64 kilobytes of internal program ROM, 202 graphics modes, 25 text modes and 7 format modes. PRINT-IT! Model 2 is compatible with all Apple II computers and over 15 models of printers. The self-contained card can handle 40- or 80-column text and high and low resolution graphics in color and black & white, and automatically selects the print format to match the current screen display in the Apple IIe. PRINT-IT! Model 2 retails for $174.00.

Texprint, Inc.
8 Blanchard Rd.
Burlington, MA 01803
(800) 255-1510

HOME COMPUTER product news™

HI-RES GRAPHICS & SPREADSHEET FOR EXTENDED BASIC

VMC SOFTWARE has announced a series of new programs for the TI-99/4A. The *Hi Res Graphics Expander* in Extended BASIC includes six new commands and is available on cassette for $15.95. *Basi-Calc 99*, a mini spreadsheet in Extended BASIC requiring no additional memory or peripherals, retails for $16.95 on cassette and $18.95 on disk. Also available are *Mini-Mail 2* and *Checkbook Plus*, each in BASIC on cassette at $13.95, and *Disk Mail 99* in BASIC on disk at $17.95.

VMC SOFTWARE
P.O. Box 326
Cambria Heights, NY 11411.

NEW FANTASY WITH INTERLOGIC

Infocom, Inc., has just announced *ENCHANTER*(tm), a fantasy game co-authored by Mark Blank and Dave Lebling, writers of the *ZORK*(tm) trilogy. The prose adventure game is written for the IBM PC, TI PC, Apple II and Commodore 64 computers. Each package includes diskette, standard documentation, and an eight-page user's manual. Priced at $49.95, the game uses INTERLOGIC(tm), a programming system with a vocabulary of 600 words that enables players to communicate with the program in ordinary English.

Infocom, Inc.
55 Wheeler Street
Cambridge, MA 02138
(617) 492-1031.

AD PILOT PROGRAM & RESEARCH/ ANALYSIS SYSTEM GOES ON-LINE

CompuServe, the commercial consumer videotex/database service, and L.M. Berry and Company have announced a program to offer advertising and direct marketing via the CompuServe Information Service. The four-month pilot program is scheduled to start in January, 1984. A variety of products will be test-advertised for consumer responses, i.e., merchandise orders or requests for product information. Spot advertisements, databases of in-depth product descriptions, and electronic catalogs, as well as order placement and delivery and payment information will be offered.

CompuServe has also announced the availability of MarketScope, a computerized database aid to marketing media research and analysis. MarketScope's information sources include databases from Mediamark Research, Inc., Simmons Market Research Bureau, Broadcast Advertiser Reports, and Arbitron. Demographic data, sales potential data, and regional newspaper studies are also included. Capabilities offered by MarketScope include screening and reporting, consumer profiles, cross tabulation, competitive product analyses, media analyses, reach and frequency studies, sales analyses, statistical modeling, graphics, and mapping.

CompuServe, Inc.
5000 Arlington Centre Blvd.
Columbus, OH 43220
A.R. Char, (614) 457-8600.

DISK/CASSETTE ACCESSORIES

Discwasher has announced the DiscKeeper(tm), a magnetically shielded storage system for floppy disks designed to protect disks against stray magnetic fields from fans, printers, peripheral motors and CRTs, as well as from airport security systems, mail handling equipment, and telephone ringers. The DiscKeeper is available in three sizes: two 5 1/4" floppy disks at $12.95, four 5 1/4" floppy disks at $16.95, and two 8" floppy disks at $14.95.

Discwasher also offers a Disk Drive Cleaner with cleaning programs for both single- and double-sided disk drive heads. The cleaning programs are for the Apple II, IBM, CP/M and VIC-20 computers. Retail prices are $24.95 for 5 1/4" drives and $29.95 for 8" drives.

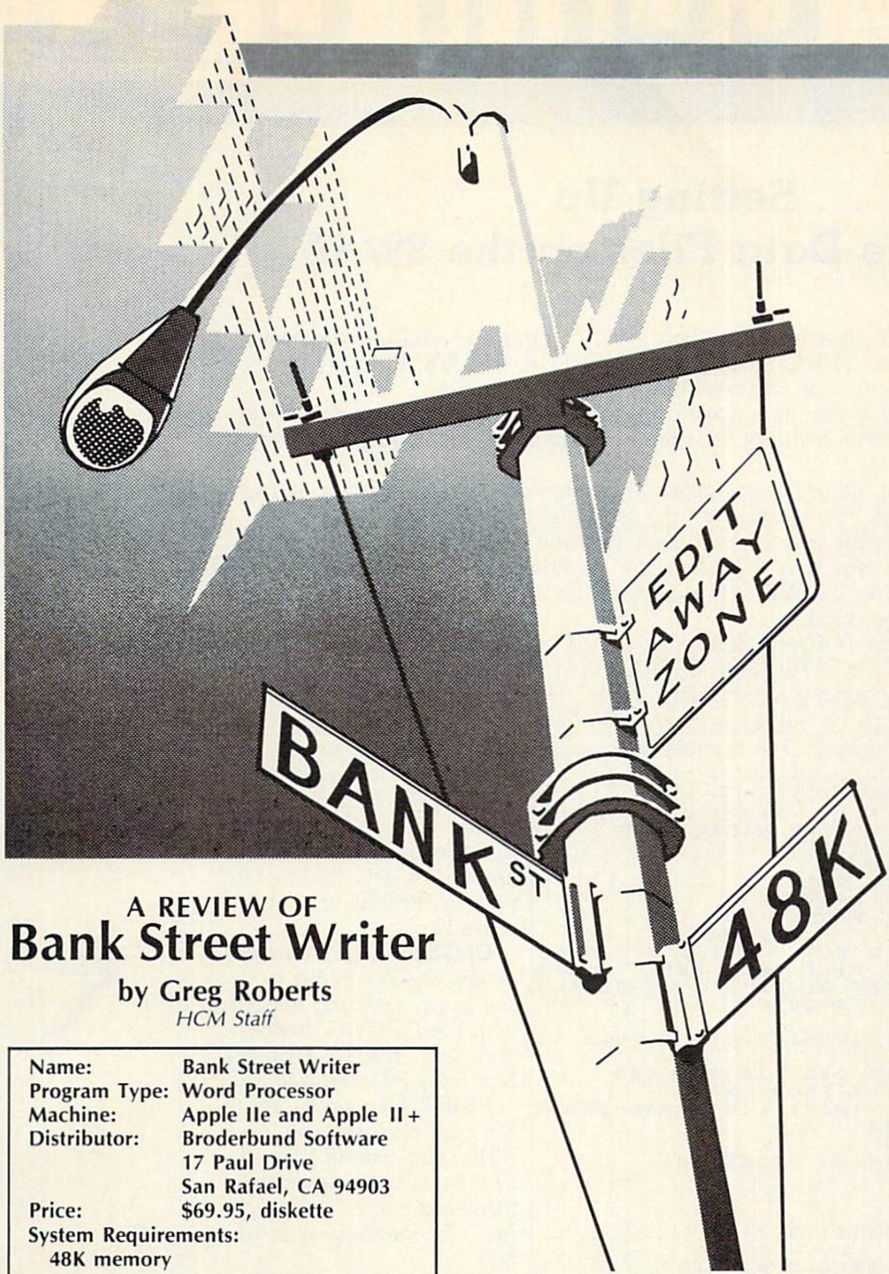
Discwasher's Computer Cassette Drive CareSet(tm) includes a computer cassette drive head cleaner and a computer cassette drive mechanism cleaner for $14.95.

Discwasher
1407 N. Providence Road
Columbia, MO 65205.

DIGITAL RECORDER FOR COMMODORE 64

Covox Co. has announced the Voice Master(tm), a digital recording device for the Commodore 64. The Voice Master permits the user to enter up to 150 words and phrases and compute responses in BASIC. The Voice Master package includes cassette software—transferable to diskette—with demonstration programs for a talking clock, calculator, and Blackjack, as well as instructions for defining keys for spoken phrases, song notes, or other sounds. The Voice Master with microphone, cassette software, manual, and newsletter updates retails for $119.95, and can be demonstrated over the telephone.

Covox Co.
675-D Conger St.
Eugene, OR 97402
(503) 342-1271
Telex 706017.



A REVIEW OF
Bank Street Writer

by Greg Roberts
HCM Staff

Name:	**Bank Street Writer**
Program Type:	**Word Processor**
Machine:	**Apple IIe and Apple II+**
Distributor:	**Broderbund Software 17 Paul Drive San Rafael, CA 94903**
Price:	**$69.95, diskette**
System Requirements:	
48K memory	

	Poor	Fair	Good	Excellent
Performance				
Documentation				
Ease of Use				

For sheer accessibility, *Bank Street Writer* is among the best word processing programs on the market. The person who has never written on a computer before has only to slide the diskette into the machine and throw on the switch. A rectangular "page" appears on the screen with the instructions: TYPE IN TEXT AT CURSOR. At this point, known as Write Mode, anyone who can type will be able to turn out a letter or article simply by entering text and letting it wrap within the 38-character format.

In Write Mode you cannot edit your text except to use the left arrow key to erase typos. You can carry out more complex editing features by pressing [ESC] for the Edit Mode menu: Erase, Unerase, Move, Moveback, Find, Replace, and Transfer Menu. Using the arrow key, take your cursor to the menu; the option you choose is highlighted on the screen. For example, once you choose Erase, you can use the arrow keys to move the cursor anywhere in the text, using the [RETURN] key to set the boundaries of the text to be eliminated—up to fifteen lines. After you've run your cursor over that part of the text, you are asked to make a Yes/No response to the editing operation before it will execute. In other words, the program walks you carefully through each step of the editing process, providing plenty of safeguards to keep you from losing your way or losing your work. A warning to speed-conscious writers: This procedure is necessary for every editing move; if you have to make many small changes on the page, it's going to take a lot of key-punching.

Accessible Editing

The program offers some very friendly features for switching paragraphs or otherwise moving blocks of text. The Move command asks you simply to position the cursor at the beginning of a block of text, press [RETURN], then move the cursor to the end of the text and press [RETURN] again. Now move your cursor to the place you'd like to see this text, hit [RETURN], and the task is carried out. For this operation the program highlights the text you wish to move; it also provides the Y/N safeguard. The program can carry no more than fifteen lines of type at a time, so very long blocks of writing would have to be handled the way ants carry bread crumbs up the drainpipe. For most writers, however, moving very large blocks of text is not a common problem.

A Welcome Transfer

To use the disk operating system, you have to go into the Transfer mode for the usual loading, saving, deleting, or printing of files. You can print your document just as it appears on the screen—in a 38-column format called "Draft"—or you can choose "Final," which lets you tailor a specific format to the printer. Here you can print out copy up to 126 characters wide, double or triple-space it, and automatically print an identification header on each page. Again, you put in your requirements in a simple straightforward way—by answering a series of formating questions appearing on the screen.

The program lets you write in upper and lower-case characters, and prints them out the same way. This feature is still not found in many word-processing programs for the Apple.

With this package comes a well-written and extremely detailed instruction booklet. Its straightforward style is all too rare in the spaghetti world of computer documentaion.

Broderbund should be applauded for making word-processing available to occasional writers who are intimidated by the complexity and cost of other programs. But it seems the designers of *Bank Street Writer* may have gone overboard with the Yes/No safeguard. Imagine having to respond to Y/N after each minor editing move. Unless you're an Oscar Wilde—who was said to have spoken in perfect sentences—you should be wary of this program's tedious editing features. They certainly slow down your writing. This drawback makes *Bank Street Writer* most suitable for dabbler writers who seldom turn out more than a page or so per week—for example, the average high-school student or the club secretary putting out a quarterly newsletter. For these lighter tasks, this word-processing program is ideal. HCM

Setting Up Cassette Data Files on the 99/4A

There is more to your TI cassette system than just SAVEing and OLDing programs. By learning to use four other TI BASIC commands (OPEN#, INPUT#, PRINT#, and CLOSE#), you will be able to save and recall data with the cassette.

Below is a short program to maintain a list of telephone numbers in an array (a contiguous block of the computer's memory for storage of similar items) whose data may be saved on tape.

There is a very important factor to remember when using a cassette tape for storage—always use a separate tape for your data, not the tape that contains your program!

OPEN # This statement prepares the computer to use a data file on an accessory device. With cassette tapes, you may "OPEN a file" in either input or output mode. See line 260 for an example using OPEN# with a cassette file in the input mode and see line 480 for an example in output mode.

Until you have studied the *TI User's Reference Guide* and mastered cassette data storage, always specify "CS1", INTERNAL, SEQUENTIAL, and FIXED as part of the OPEN# statement. You may specify any number from 1 to 255 for the device (after the # sign), but this number must be the same in all PRINT#, INPUT#, and CLOSE# statements that refer to the device. The number following the FIXED option should be 64, 128, or 192, specifying the number of bytes in each record. Select this number to be larger than the amount of data (characters) you are PRINTing to the cassette.

```
100 REM   ****************
110 REM   *    PHONE LIST   *
120 REM   ****************
130 REM   HOME COMPUTER MAGAZINE
140 REM   VERSION 4.1.1
150 REM   TI BASIC
160 REM   CASSETTE DATA
170 REM   STORAGE EXAMPLE
180 DIM NAMES(10),PHONES(10)
190 CALL CLEAR
200 PRINT "  P H O N E  L I S T"::::
210 PRINT "1-READ FILE FROM TAPE"::"2-R
    EVIEW AND ENTRY OF DATA"::"3-SAVE F
    ILE TO TAPE"::"4-QUIT"::::
220 INPUT CHOICE
230 IF (CHOICE>4)+(CHOICE<1)=-1 THEN 19
    0
240 ON CHOICE GOTO 250,320,470,540
250 REM   READ FILE FROM TAPE
260 OPEN #1:"CS1",INPUT ,INTERNAL,SEQUE
    NTIAL,FIXED 64
270 FOR N=1 TO 10
280 INPUT #1:NAMES(N),PHONES(N)
290 NEXT N
300 CLOSE #1
310 GOTO 190
320 REM   ENTER DATA IN FILE
330 CALL CLEAR
340 PRINT "WHICH RECORD NUMBER"
350 INPUT ENTRY
360 IF (ENTRY>10)+(ENTRY<1)=-1 THEN 190
370 CALL CLEAR
380 PRINT "ENTRY NUMBER ",ENTRY," IS:":
    NAMES(ENTRY)::"WHOSE PHONE # IS:":P
    HONES(ENTRY)::
390 PRINT "1-ENTER NEW NAME":"2-ENTER N
    EW PHONE NUMBER":"3-TRY ANOTHER ENT
    RY":"4-EXIT REVIEW AND ENTRY MODE"
400 INPUT CHOICE
410 IF (CHOICE>4)+(CHOICE<1)=-1 THEN 37
    0
420 ON CHOICE GOTO 430,450,330,190
430 INPUT "NAME? ":NAMES(ENTRY)
440 GOTO 370
450 INPUT "PHONE #? ":PHONES(ENTRY)
460 GOTO 370
470 REM   SAVE FILE TO TAPE
480 OPEN #1:"CS1",OUTPUT,INTERNAL,SEQUE
    NTIAL,FIXED 64
490 FOR N=1 TO 10
500 PRINT #1:NAMES(N),PHONES(N)
510 NEXT N
520 CLOSE #1
530 GOTO 190
540 END
```

CLOSE # This statement causes the computer to finish any pending action with the OPENed device and then "forget" about it. See lines 300 and 520.

PRINT # To save data to the cassette tape, the PRINT# statement is used. The items to be saved are formated into a "variable-list." An example is in line 500.

INPUT # To load data from the cassette tape, the INPUT# statement is used. An example of this is shown in line 280 of the program. Note that the format of the "variable-list" must match that used in the PRINT# statement that recorded the data.

Just entering this short program and seeing how it works with the cassette will help you think of other ways to use this technique for saving other types of data.

—David Brader

Mathematical Accuracy With the IBM PC and PCjr

The Intel 8088 microprocessor in the IBM PC and PCjr is the leading 16-bit chip in the world today. With features like built-in multiply and divide hardware acting on full word (16-bit) operands, it is one of the most accurate microprocessors on the market. As we at *HCM* were working on the *Saving* program in this issue ("Porsches and Other Pipedreams"), we were surprised to find that the IBM PC's answers to the problems used to check accuracy were different from those of the other machines. The answers were not drastically different (just a few pennies in most cases), but we knew that a penny saved is a penny earned, and the IBM wasn't saving as many pennies as the others. If you refer to "Microcomputer Accuracy" in this issue, you will see the source of the inaccuracy. The IBM PC is accurate to only 7 significant figures, while the Apple and Commodores are accurate to 9-10 places and the TI-99/4A is accurate to 13.

Why is the IBM with its 16-bit microprocessor less accurate than other machines (even 8-bit ones) in BASIC? After a little consultation with our BASIC manual we concluded that if we defined the variables in our program to be *double-precision*, we would attain accuracy to 16 places. Placing the following line:

```
200 DEFDBL A-W,Y
```

in our program before any math functions, we reasoned, would make all variables double-precision variables (except for variables starting with X and Z). By including this instruction, our program should attain 16-digit accuracy and be even more accurate than the other machines—or so we thought. . . To our dismay we found that the IBM PC *Saving* program was still pennies (and sometimes dollars) off because certain mathematical functions in BASIC—SIN, COS, LOG, EXP, and ^ (exponentiation), among others—give only single-precision (7-digit) accuracy.

We found a key to this problem on page 110 of the *IBM BASIC Compiler* manual:

> "If you use double-precision operands for any of the arithmetic functions, including the transcendental functions (SIN, COS, TAN, ATN, LOG, EXP, and SQR), then the BASIC Compiler returns double precision results. Only single precision results are returned by the interpreter." [Emphasis added.]

This means that if you own an IBM PC or PCjr, the resident BASIC Interpreter is unable to give results more accurate than single-precision when, say, the ^ operator is used in a function. Several functions use this operator in the *Saving* program:

```
220 DEF FNPV(1-1/(1+I^N)/I
230 DEF FNFV=((1+I)^N-1)/I
.
.
260 DEF FNC=PRE(X)*(I+IN/(100*K)) ^ (K*TY)
```

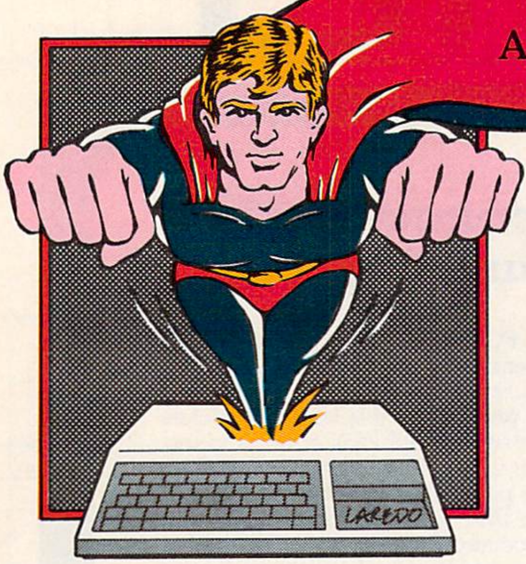
The IBM PCjr's Cartridge BASIC does return double-precision results from these functions, making its version of the *Saving* program the most accurate in the article. Also, if you enter BASIC or BASICA from DOS 2.0 on the IBM PC and type /D on the BASIC command line, you get double-precision results. But the interpreted BASIC available with Cassette BASIC on either IBM computer will, as the manual says, return only single-precision accuracy on the transcendental functions we've mentioned above.

However, if the program is compiled and RUN, it attains double-precision accuracy on either the PC or PCjr, and its results are more accurate than those of the Apple IIe, Commodore 64, or VIC-20. In order to compile a BASIC program you need at least one disk drive and the *IBM BASIC Compiler* package from Microsoft. If you don't have the compiler, you might check with your local IBM users group. One of the members who has the package might be willing to compile your *Saving* program for you. Once it has been compiled and linked, the program can be RUN directly from either DOS 2.0 on the PC or DOS 2.1 on the PCjr when either machine is equipped with a disk drive.

—Roger Wood

Eric A. Martin 83'

SUPER LANGUAGE
A Home Computer Assembly Language Series



Have No Fear:

Assembly Language Won't Byte!

Part III

by Peter Lottrup

In this segment of our tutorial on Assembly Language for the Mini Memory cartridge, we'll develop another short demonstration program and execute it. With this program, we'll learn how to enter a name in the Mini Memory's table of references and definitions (REF/DEF Table). An entry in this table allows us to run a program by *name*, rather than executing it using EASY BUG with a memory *address*.

In the following program, we will learn two new words of the Assembly Language vocabulary and how to use them. Later in this series, we'll provide a list of words with explanations and examples of the most important ones. To enter the program, load the *Line-by-Line Assembler*, using EASY BUG. Then select option 2 from the Mini Memory menu and type OLD in response to the prompt.

Shooting Asterisks

The object of the following program is to display an asterisk in the first screen position, erase it, print it again in the next screen positions, and so on. This will demonstrate the speed of Assembly Language. The following listing will display the asterisk:

```
7D00    AQ   LI R0,0
7D04    WD   LI R1,LB
7D08         LI R2,1
7D0C         BLWP @>6028
7D10         LI R1,LS
7D14         BLWP @>6028
7D18         INC R0
7D1A         CI R0,768
7D1E         JNE WD
7D20         JMP AQ
7D22    LB   TEXT '*'
7D24    LS   TEXT ' '
```

This program was written using the VMBW routine (VDP Multiple Byte Write) stored in memory location >6028. Note that it could be done just as well with the VSBW routine (VDP Single Byte Write) stored in Memory location >6024. Both of these routines are covered in the Mini Memory manual, starting on page 35. To give you the best understanding of how to use each routine, the first program listing uses the VMBW routine, and the second uses the VSBW routine.

Load, Skip and Jump

The first line loads the value of zero into register number zero, which is the position on the screen where we want the first asterisk printed. Next we load the label where our text is—in this case, LB—into register number one. In the third line, we load into register number two the length of our text (one byte long). In the fourth line we tell the computer to "GOSUB" (BLWP: Branch and Load Workspace Pointer) to the VMBW routine located in memory location >6028. That prints the first asterisk. In the fifth line, we load into register number one the label—or memory location—where the new text (a space) is and then repeat the printing procedure. In the seventh line, we INCrement the value in register number zero by one. INC always increases a value by one. Then, before returning to the printing routines, we must test to see if the value in register zero is equal to the last screen position (768). If it is, the value in the register must be reset to zero so that the printing will begin again in the first screen position. In the eighth line, we compare the value in register zero with 768, using the CI (Compare Immediate) instruction.

At this point we should clarify the use of the Compare instruction. When comparing a register to a number, we use CI (Compare Immediate), and when comparing two registers, we use C (Compare words). Both of the instructions alter bits in the CPU's Status Register. As your knowledge of Assembly Language increases, you will learn to differentiate among all these instructions and definitions with ease.

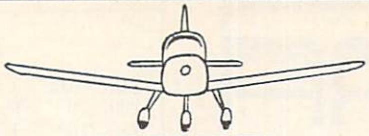
In the next line we write JNE (Jump if Not Equal) to the label WD. Otherwise, in the next line, we jump (JMP) to label AQ. The jump instructions test the bits in the Status Register and execute according to the conditions those bits indicate. Finally, we add the two labels with the text—one with the asterisk, the other with the blank.

Now END the program and execute it. (Remember: To execute, return to the master title screen, select EASY BUG, and use the Execute command with a starting address of 7D00.) You will see that the program works so quickly that you cannot see the full asterisk printed out before it is erased. To make the program execute better, we will have to slow it down a bit.

To do this, we will add two delay loops in the right places. Here is the new listing:

```
1.  7D00   AQ   LI R0,0
2.  7D04   WD   LI R1,>2A00
3.  7D08        BLWP @>6024
4.  7D0C        LI R2,300
5.  7D10        LI R3,1
6.  7D14   LO   INC R3
7.  7D16        C R2,R3
8.  7D18        JNE LO
9.  7D1A        LI R1,>2000
10. 7D1E        BLWP @>6024
11. 7D22        LI R3,1
12. 7D26   KL   INC R3
13. 7D28        C R2,R3
14. 7D2A        JNE KL
15. 7D2C        INC R0
16. 7D2E        CI R0,768
17. 7D32        JNE WD
18. 7D34        JMP AQ
```

This listing is the same as the first one, except that it has been written using the VSBW (VDP Single Byte Write) routine and has a couple of delay loops added to it. Please note that the line numbers (left column) have no relation whatsoever to the program listing, but were included to make it easier to understand the explanation of the program. Note also that the hexadecimal translations of the contents of each memory location (second column above) have not been included because they are unnecessary at this point.



Explanation of the Program

Line 1, with the label AQ, loads into register number zero the value of the first screen position (0). Line 2, with the label WD, loads into register number one the hexadecimal ASCII code of the asterisk (>2A). This hexadecimal value is entered, followed by two trailing zeros (>2A00). Line 3 writes the asterisk into VDP RAM. (This time we've switched to the VSBW routine in memory location >6024.) Lines 4 through 8 are the delay loop. First we load the value 300 (decimal) into register two and the value 1 into register number three. Then all we do is INCrement register three (line 6) and Compare it to register number two (line 7). Line 8 means Jump if Not Equal to label LO. The loop continues until both registers are equal (300 times).

At this point then, it is clear that lines 4 through 8 are similar to the following Extended BASIC statement:

```
FOR G = 1 TO 300 :: NEXT G
```

Just remember, though, that 300 times in Assembly Language is much faster than 300 in BASIC. We will experiment with the speed again further on.

Line 9 changes the hexadecimal ASCII value in register one to the value corresponding to a space (decimal value 32, hexadecimal 20) instead of an asterisk. Line 10 writes it onto the screen.

Lines 11 through 14 are the new delay loop. Note that the value in register number two is not included because it has not been changed from the first delay loop. Finally, the last four lines are the same as in the first program. Now END the program and execute it. Does the asterisk move better now? If you want to make it go slower or faster, just invoke the *Line-by-Line Assembler* and type:

```
7D00    AORG >7D0C
7D0C    LI   R2, number
```

Then press [ENTER], type END, and execute the program again.

> **"You will see that the program works so quickly that you cannot see the full asterisk printed out before it is erased."**

Checking for Space

If you want to save the program by name, you will first have to find out whether there is enough space to add the name and starting location of the program. The table where this information is stored is called the REF/DEF table. You need eight bytes in this table to store the name. Your next step—checking the remaining space—is really not necessary if your programs are short, but you should still know how to do it.

First, with the AORG Directive, you will check the value in address >701C (the location of the First Free Address of the Module) and the value in >701E (the Last Free Address of the Module). Then find the difference between the values to see if you have eight bytes or more. So right after you have copied the last line of the example program, type:

```
7D30 83A0 AORG >701C
701C 7FB2 AORG >701E
701E 7FE8 ■
```

(■ shows the position of the cursor.)

7FB2 is the value in 701C, and 7FE8 is the value in 701E. Subtracting the first from the second (7FB2 from 7FE8), we get hexadecimal 36 (or 54 decimal), enough to store the information about our program.

Now that we know there is enough space remaining, we will tell the computer which is the new First Free Address of the Module (the one right after our program finishes) by placing that value into >701C, thus updating the First Free Address. So type:

```
701E 7604 AORG >701C
701C 7D30 DATA >7D36
701E 7FE8 ■
```

Next, we must change the value in the Last Free Address of the Module (>701E) to a number eight bytes smaller so the name and starting point of our program can be added:

```
701E 7FE0 DATA >7FE0
7020 71A6 ■
```

The number we give the computer (7FE8 − 8 = 7FE0) is the place where the information about our program will be found.

Now we can give a name to the program. This name can be from one to six

Continued on p. 137

Meltdown . . . from p. 75

Meltdown

Explanation of the Program

Line Nos.	
100-170	Program Header.
180-210	Title screen.
220-330	Define characters.
340-380	Initialize variables and fill arrays.
390-440	Choose joystick or keyboard.
450-540	Draw playing area and hidden gamma particles and rays.
550-580	Change letters and numbers to write. Display scorekeeping.
590-630	Display level number.
640-660	Draw hero sprite. Check to see if enough particles have been created to move to higher level.
670-710	If no oxygen tank already present, randomly create and place one.
720-890	Randomly turn on hidden gamma rays and check to see if hero gets hit.
900-950	If level is greater than 1, then randomly place anti-matter particles (which appear more frequently at higher levels).
960-990	Choose particle or roach. (Roaches first appear at level 3 and appear more often at higher levels.)
1000-1050	Randomly choose position and color of particle or roach. Play appropriate sound. Move hero.
1060-1090	Shoot neutralizer. Check for various coincidences.
1100-1130	Move hero. Check for coincidences. Shoot.
1140-1180	Delete bullet or particle if off screen.
1190-1240	Update score.
1250-1270	Destroy particle if hit. Change roach to "super roach" if hit only once.
1280-1330	Destroy particle or roach. Update score.
1340-1390	Destroy hero. Update score.
1400-1420	Obtain oxygen. Update score.
1430-1750	Choose appropriate end of game.
1760-1830	Hero move subroutine.
1840-1940	Shoot subroutine.
1950-2010	Level color change.
2020-2110	Data.

TI-99/4A

```
100 REM ************
110 REM *  MELTDOWN  *
120 REM *           *
130 REM ************
140 REM BY STEVE LANGGUTH
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.1.1
170 REM TI EXTENDED BASIC
180 REM
190 REM**TITLE SCREEN**
200 REM
210 CALL CLEAR :: DISPLAY AT(11,11):"ME
    LTDOWN"
220 REM *DEFINE CHARACTERS*
230 CALL CHAR(43,"0000FF8181FF0000",37,
    "007E7E42427E7E00")
240 CALL CHAR(96,"03060603013F34353437 0
    303030303 0FC06060C080FC2CAC2CE06060
    6078 0000")
250 CALL CHAR(100,"0306060301 3F3435340 7
    0606061E0000C06060C080FC2CAC2CECC0C
    0C0C0C0F0")
260 CALL CHAR(104,"00000000000038EE0300
    000000000000000000000038EE830000 0
    0000000000")
270 CALL CHAR(108,"0103060C193366CCCC66
    33190C06030180C0603098CC66333366CC9
    83060C080")
280 CALL CHAR(132,"66",116,"66",124,"66
    ")
290 CALL CHAR(112,"001F204040 4E5B51515B
    4E4040201F0000F8040202020232 4A12227
    A0204F800")
300 CALL CHAR(120,"0000103B12021E10101E
    02123B1000000000008DC48407808087840 4
    8DC080000")
310 CALL CHAR(140,"C1E171381C0E07E3E307
    0E1C3871E1C183878E1C3870E0C7C7E0703
    81C8E8783")
320 CALL CHAR(128,"01211008020401 1BC309
    0411214000000000 02848820 90C3D880204
    01008848 0")
330 CALL CHAR(136,"0000121212 3F7FD5D57F
    3F1212120000000000484848FCFEABABFEFC4
    84848 0000")
340 REM **INITIALIZATION**
```

TI-99/4A

```
350 RESTORE :: OPTION BASE 1 :: DIM ROW(12):: DIM COL(12):: DIM VEL(12)::DIM MP(12):: DIM PPR(6):: DIM PPC(6)
360 FOR X=1 TO 12 :: READ ROW(X),COL(X),VEL(X):: NEXT X
370 FOR X=1 TO 12 :: READ MP(X):: NEXT X
380 FOR X=1 TO 6 :: READ PPR(X),PPC(X):: NEXT X
390 REM **KEYS OR JOYST**
400 CALL CLEAR :: DISPLAY AT(7,6):"DO YOU WISH TO USE" :: DISPLAY AT(9,10):"1) JOYSTICK "
410 DISPLAY AT(11,3):"-NOTE-RELEASE ALPHA LOCK-"
420 DISPLAY AT(14,10):"2) KEYBOARD ?" :: DISPLAY AT(16,6):"PRESS 1 OR 2"
430 CALL KEY(0,K,S):: IF K<>49 AND K<>50 THEN 430
440 IF K=49 THEN K1=1 ELSE K1=0
450 REM **DRAW PLAY FIELD**
460 CALL COLOR(1,2,2,2,2,2,11,2,2,12,2,2,13,2,2)
470 CALL CLEAR :: CALL SCREEN(2):: CALL MAGNIFY(3)
480 FOR T=1 TO 4 :: READ R1,C1 :: CALL HCHAR(R1,C1,37,14):: NEXT T
490 FOR T=1 TO 10 :: READ R2,C2 :: CALL HCHAR(R2,C2,43,14):: NEXT T
500 FOR T=1 TO 4 :: READ R3,C3 :: CALL VCHAR(R3,C3,37,2):: NEXT T
510 FOR T=1 TO 6 :: READ R4,C4,CH4 :: CALL HCHAR(R4,C4,CH4,15):: NEXT T
520 FOR T=1 TO 6 :: READ R5,C5 :: CALL SPRITE(#(T+5),140,2,R5,C5):: NEXT T
530 CALL COLOR(1,5,2,2,5,2)
540 FOR X=3 TO 8 :: CALL COLOR(X,16,2):: NEXT X
550 ARMOR=5 :: OXYGEN=100 :: SCORE=0 :: LEVEL=1 :: PN=0 :: OF=0 :: AM=0
560 DISPLAY AT(23,19)SIZE(5):"SCORE" :: DISPLAY AT(22,1)SIZE(6):"OXYGEN"
570 DISPLAY AT(24,2)SIZE(5):"ARMOR"
580 DISPLAY AT(22,7)SIZE(4):OXYGEN :: DISPLAY AT(24,8)SIZE(3):ARMOR
590 DISPLAY AT(12,14)SIZE(2):LEVEL :: CALL SOUND(100,(330*LEVEL),2,(392*LEVEL),2,(523*LEVEL),2)
600 GOSUB 1950
610 CALL SOUND(25,262,30):: CALL SOUND(500,(330*LEVEL),2,(392*LEVEL),2,(523*LEVEL),2)
620 FOR T=1 TO 250 :: NEXT T
630 DISPLAY AT(12,14)SIZE(2):"  "
640 REM **CREATE HERO**
650 CALL SPRITE(#1,96,3,165,121):: S=0 :: SH=0
660 PN=PN+1 :: IF PN>(15*LEVEL)THEN 1620
670 REM **CREATE OXYGEN**
680 IF OF=1 THEN 730
690 RANDOMIZE :: Q=INT(RND*3)+1 :: P=INT(RND*6)+1
700 IF Q<>2 THEN 730
710 CALL SPRITE(#4,112,12,PPR(P),PPC(P)):: OF=1 :: CALL SOUND(50,3000,3,4000,5)
720 REM **GAMMA RAYS**
730 RANDOMIZE :: X=INT(RND*2)+1 :: IF X<>2 THEN 910
740 RANDOMIZE :: Y=INT(RND*3)+1 :: ON Y GOTO 750,800,850
750 CALL COLOR(#6,16):: CALL SOUND(200,2500,2):: CALL COLOR(#7,16):: CALL SOUND(200,2600,2)
760 GOSUB 1770
770 CALL COLOR(11,16,2):: CALL SOUND(100,1000,2,1200,4):: CALL POSITION(#1,DR,DC)
780 IF (DR<20)+(DR>30)=0 OR(DR<92)+(DR>102)=0 THEN CALL COLOR(11,2,2):: CALL COLOR(#6,2,#7,2):: GOTO 1340
790 CALL COLOR(11,2,2):: CALL COLOR(#6,2,#7,2):: GOTO 910
800 CALL COLOR(#8,16):: CALL SOUND(200,2700,2):: CALL COLOR(#9,16):: CALL SOUND(200,2800,2)
810 GOSUB 1770
820 CALL COLOR(12,16,2):: CALL SOUND(100,1000,2,1200,4):: CALL POSITION(#1,DR,DC)
830 IF (DR<44)+(DR>54)=0 OR(DR<116)+(DR>126)=0 THEN CALL COLOR(12,2,2):: CALL COLOR(#8,2,#9,2):: GOTO 1340
840 CALL COLOR(12,2,2):: CALL COLOR(#8,2,#9,2):: GOTO 910
850 CALL COLOR(#10,16):: CALL SOUND(200,2900,2):: CALL COLOR(#11,16):: CALL SOUND(200,3000,2)
860 GOSUB 1770
870 CALL COLOR(13,16,2):: CALL SOUND(100,1000,2,1200,4):: CALL POSITION(#1,DR,DC)
880 IF (DR<68)+(DR>78)=0 OR(DR<140)+(DR>150)=0 THEN CALL COLOR(13,2,2):: CALL COLOR(#10,2,#11,2):: GOTO 1340
```

Continued on p. 112

Meltdown

TI-99/4A

```
890 CALL COLOR(13,2,2) :: CALL COLOR(#10,2,#11,2) :: GOTO 910
900 REM **ANTIMATTER**
910 IF LEVEL<2 THEN 970
920 RANDOMIZE :: X=INT(RND*2)+1 :: Y=INT(RND*(8-LEVEL))+1 :: Z=INT(RND*6)+1
930 IF AM<>1 THEN 950
940 IF X=2 THEN CALL DELSPRITE(#5) :: CALL SOUND(50,3333,2,4444,3) :: AM=0 :: GOTO 970 ELSE GOTO 970
950 IF Y=1 THEN CALL SPRITE(#5,120,6,PPR(Z),PPC(Z)) :: AM=1 :: CALL SOUND(50,2222,2,3333,3)
960 REM **PARTICLE CHOOSE**
970 RANDOMIZE :: X=INT(RND*(LEVEL*2))+1 :: ON X GOTO 980,980,980,980,990,980,990,990,990,990
980 CN=108 :: GOTO 1010
990 CN=136 :: GOTO 1010
1000 REM **MAIN GAME LOOP**
1010 GOSUB 1770
1020 RANDOMIZE :: X=INT(RND*12)+1 :: Y=INT(RND*13)+3 :: CALL SPRITE(#3,CN,Y,ROW(X),COL(X),0,VEL(X)*(5+LEVEL))
1030 IF CN=108 THEN 1040 ELSE 1050
1040 FOR A=0 TO 30 STEP 15 :: CALL SOUND(-99,250+Y*10,A,760+Y*10,A,1512+Y*10,A) :: NEXT A :: GOTO 1060
1050 FOR T=1 TO 3 :: CALL SOUND(25,5000,2,5555,3,6000,2) :: NEXT T
1060 IF SH<>0 THEN CALL COINC(#2,#3,15+LEVEL,HIT1) :: IF HIT1=-1 THEN 1250
1070 GOSUB 1850
1080 IF OF<>1 THEN 1090 ELSE CALL COINC(#1,#4,15,HIT3) :: IF HIT3=-1 THEN 1410
1090 IF AM<>1 THEN 1100 ELSE CALL COINC(#1,#5,10,HIT5) :: IF HIT5=-1 THEN AM=0 :: GOTO 1440
1100 GOSUB 1770
1110 CALL COINC(#1,#3,15,HIT2) :: IF HIT2=-1 THEN 1340
1120 IF SH<>0 THEN CALL COINC(#2,#3,15+LEVEL,HIT1) :: IF HIT1=-1 THEN 1250
1130 GOSUB 1850
1140 CALL POSITION(#2,DR2,DC2) :: IF DC2>40 AND DC2<210 THEN 1160
1150 CALL DELSPRITE(#2) :: SH=0
1160 GOSUB 1770
1170 CALL POSITION(#3,DR3,DC3) :: IF DC3>25 AND DC3<225 THEN 1220
1180 CALL DELSPRITE(#3) :: IF SCORE<5 THEN SCORE=0 ELSE SCORE=SCORE-5
1190 OXYGEN=OXYGEN-1 :: DISPLAY AT(23,24):SCORE
1200 DISPLAY AT(22,7)SIZE(4):OXYGEN :: RH=0
1210 IF OXYGEN<1 THEN 1440 ELSE 660
1220 OXYGEN=OXYGEN-1 :: DISPLAY AT(22,7)SIZE(4):OXYGEN :: IF OXYGEN<1 THEN 1440
1230 CALL SOUND(40-(LEVEL*3),1000+(LEVEL*200),2,500+(LEVEL*150),2)
1240 CALL SOUND(40-(LEVEL*3),500+(LEVEL*200),2,250+(LEVEL*150),2) :: GOTO 1060
1250 IF CN=108 THEN 1280
1260 IF RH=1 THEN 1280 ELSE CALL DELSPRITE(#2) :: RH=1 :: SCORE=SCORE+10 :: CALL COLOR(#3,16)
1270 CALL MOTION(#3,0,VEL(X)*(7+LEVEL)) :: GOTO 1060
1280 REM **DESTROY PARTICLE*
1290 CALL DELSPRITE(#2) :: CALL MOTION(#3,0,0,#1,0,0) :: RH=0
1300 FOR T=1 TO 5 :: CALL COLOR(#3,2) :: CALL SOUND(-25,1111,2)
1310 CALL COLOR(#3,10) :: CALL SOUND(-25,1122,4) :: NEXT T
1320 CALL PATTERN(#3,128) :: FOR T=1 TO 5 :: CALL SOUND(-99,500*T,2+T) :: NEXT T
1330 CALL SOUND(50,4000,2) :: CALL DELSPRITE(#3) :: SH=0 :: SCORE=SCORE+10 :: DISPLAY AT(23,24):SCORE :: GOTO 660
1340 REM**DESTROY HERO**
1350 CALL DELSPRITE(#3) :: CALL MOTION(#1,0,0)
1360 FOR T=1 TO 5 :: CALL COLOR(#1,2) :: CALL SOUND(-99,750,2)
1370 CALL COLOR(#1,16) :: CALL SOUND(-99,775,4) :: NEXT T
1380 CALL PATTERN(#1,128) :: FOR T=5 TO 1 STEP -1 :: CALL SOUND(-99,450*T,1+T) :: NEXT T
1390 CALL DELSPRITE(#1) :: ARMOR=ARMOR-1 :: DISPLAY AT(24,8)SIZE(3):ARMOR :: IF ARMOR<1 THEN 1440 ELSE 650
1400 REM **OBTAIN OXYGEN**
1410 CALL DELSPRITE(#4) :: CALL SOUND(100,200,3,300,4) :: OF=0 :: OXYGEN=OXYGEN+10
1420 DISPLAY AT(22,7)SIZE(4):OXYGEN :: GOTO 1060
1430 REM **END GAME**
1440 CALL DELSPRITE(ALL)
1450 FOR T=10 TO 2 STEP -1
1460 CALL SOUND(-1000,-7,T) :: CALL SCREEN(12) :: CALL COLOR(1,12,12,2,12,12,11,12,12,12,12,12,13,12,12)
1470 CALL SOUND(-1000,-7,T-1) :: CALL SCREEN(9) :: CALL COLOR(1,9,9,2,9,9,11,9,9,12,9,9,13,9,9)
1480 NEXT T
1490 CALL SCREEN(2) :: CALL COLOR(1,2,2,2,2,2,11,2,2,12,2,2,13,2,2)
1500 FOR T=1 TO 5 :: CALL SOUND(-1000,1000+(200*T),T) :: NEXT T
1510 CALL SOUND(100,125,1)
1520 IF OXYGEN<1 THEN DISPLAY AT(8,10)ERASE ALL:"OUT OF OXYGEN"
1530 IF ARMOR<1 THEN DISPLAY AT(8,10)ERASE ALL:"OUT OF ARMOR"
1540 IF HIT5=-1 THEN DISPLAY AT(8,10)ERASE ALL:"ANTIMATTER HIT!"
1550 DISPLAY AT(12,10)SIZE(5):"SCORE" :: DISPLAY AT(12,17):SCORE
1560 IF SCORE>HIGH THEN HIGH=SCORE
1570 DISPLAY AT(14,10)SIZE(4):"HIGH" :: DISPLAY AT(14,17):HIGH
1580 DISPLAY AT(19,10):"GAME OVER"
1590 DISPLAY AT(21,5):"PLAY AGAIN? Y OR N"
```

TI-99/4A

```
1600 CALL KEY(0,K,S):: IF K=89 OR K=121 THEN CALL CLEAR :: GOTO 350
1610 IF K=78 OR K=110 THEN 1750 ELSE 1600
1620 LEVEL=LEVEL+1 :: PN=0 :: IF LEVEL>5 THEN 1630 ELSE 590
1630 CALL CLEAR :: CALL SCREEN(2):: CALL DELSPRITE(ALL)
1640 CALL SOUND(200,262,4):: CALL SOUND(200,330,4):: CALL SOUND(200,392,4):: CALL SOUND(100,523,3)
1650 CALL SOUND(150,523,30):: CALL SOUND(200,392,3):: CALL SOUND(500,330,2,392,2,523,2)
1660 CALL SOUND(200,131,2,165,2,196,2)
1670 DISPLAY AT(8,7):"CONGRATULATIONS !" :: DISPLAY AT(11,6):"YOU HAVE JUST SAVED"
1680 DISPLAY AT(12,6):"THE REACTOR AND THE" :: DISPLAY AT(13,7):"ENTIRE WORLD FROM"
1690 DISPLAY AT(14,6):"COMPLETE DESTRUCTION."
1700 DISPLAY AT(16,6):"YOUR MOTHER WOULD BE" :: DISPLAY AT(17,11):"PROUD !!!" :: DISPLAY AT(20,3):"PLAY AGAIN?"
1710 DISPLAY AT(21,4):"PRESS Y OR N"
1720 DISPLAY AT(5,3):"SCORE ";SCORE
1730 CALL KEY(0,K,S):: IF K=89 THEN CALL CLEAR :: GOTO 350
1740 IF K=78 THEN 1750 ELSE 1730
1750 END
1760 REM **HERO MOVE SUB**
1770 IF K1<>1 THEN 1800
1780 CALL JOYST(1,XR,YR):: IF XR=0 AND YR<>0 THEN CALL MOTION(#1,-YR*4,0):: GOTO 1820
1790 CALL MOTION(#1,0,0):: RETURN
1800 CALL KEY(0,K,S):: IF K=69 THEN CALL MOTION(#1,-16,0):: GOTO 1820
1810 IF K=88 THEN CALL MOTION(#1,16,0):: GOTO 1820 ELSE CALL MOTION(#1,0,0):: RETURN
1820 IF HP=0 THEN CALL PATTERN(#1,100):: CALL SOUND(3,-3,2):: CALL SOUND(20,-7,10):: HP=1 :: RETURN
1830 CALL PATTERN(#1,96):: CALL SOUND(3,-3,2):: CALL SOUND(20,-6,10):: HP=0 :: RETURN
1840 REM *SHOOT SUB*
1850 IF SH=1 THEN RETURN
1860 IF K1<>1 THEN 1880
1870 CALL JOYST(1,XR,YR):: IF XR=0 THEN RETURN ELSE 1890
1880 CALL KEY(0,K,S):: IF K<>68 AND K<>83 THEN RETURN
1890 CALL POSITION(#1,DR,DC):: Z=INT(((DR-1)/12)-.05):: IF Z<1 OR Z>12 THEN RETURN
1900 IF K1<>1 THEN 1920
1910 IF XR=4 THEN 1930 ELSE 1940
1920 IF K=83 THEN 1940
1930 CALL SPRITE(#2,104,16,MP(Z),122,0,16):: CALL SOUND(100,500,2):: SH=1 :: RETURN
1940 CALL SPRITE(#2,104,16,MP(Z),107,0,-16):: CALL SOUND(100,500,2):: SH=1 :: RETURN
1950 REM *CALL SCREEN GOSUB*
1960 ON LEVEL GOTO 1970,1980,1990,2000,2010
1970 RETURN
1980 CALL SCREEN(13):: RETURN
1990 CALL SCREEN(5):: RETURN
2000 CALL SCREEN(14):: RETURN
2010 CALL SCREEN(10):: RETURN
2020 REM **DATA**
2030 DATA 25,14,1,49,14,1,73,14,1,97,14,1,121,14,1,145,14,1
2040 DATA 25,237,-1,49,237,-1,73,237,-1,97,237,-1,121,237,-1,145,237,-1
2050 DATA 25,25,49,49,73,73,97,97,121,121,145,145
2060 DATA 25,120,49,120,73,120,97,120,121,120,145,120
2070 DATA 3,1,3,19,21,1,21,19
2080 DATA 6,1,6,19,9,1,9,19,12,1,12,19,15,1,15,19,18,1,18,19
2090 DATA 1,14,1,19,22,14,22,19
2100 DATA 5,3,116,14,16,116,8,16,124,17,3,124,11,3,132,20,16,132
2110 DATA 25,14,97,237,49,237,121,14,73,14,145,237
```

HCM

Restoring DATA Pointers

To read DATA statements embedded in the body of a program, the BASIC interpreter uses pointers to keep its place. When the program starts up, the pointers point to the first data item of the first DATA statement. After each READ statement, the pointers move to the next data item available, either in the same DATA statement or in the next one encountered. After a program READs through all the data items available to it, the pointers no longer point to a piece of data, but sit after the last data item.

You must always keep track of the data item you're reading so you don't try to READ past the last item. That will cause an error and stop your program. You can keep track either by counting how many items there are, or by using a flag—a special value, like "999" that indicates the end of data.

In the Commodore 64 version of *Larry's Ten Fiddle Tunes* it's necessary to READ the same blocks of data more than once as the program continues to run. Most systems provide a statement that places the pointers at the beginning of any DATA statement in the program, either before or after its current position. It usually looks like this:

RESTORE *line number*

The TI version of *Larry's Ten Fiddle Tunes* uses this method to reread the data for the notes of any given fiddle tune. When it needs to play a tune, it places the pointers at the first data item of the first DATA statement for that tune.

On the Commodore 64 system, the RESTORE statement always places the pointers at the very first DATA item of the very first data statement of the program. In order to play the tunes as they are played on the TI-99/4A, we developed a subroutine that allows us to move the pointers to any DATA line, rather than to just the first. This subroutine uses an array to store the pointer values of any DATA lines that will be used more than once. This allows for a RESTORE to any DATA line number.

To construct a RESTORE command for any data line desired, we must first know the current data line and current data address pointers. We can find the current data line with a PEEK(63) and a PEEK(64). The 63 contains the low byte, and 64 contains the high byte. We find the current data item address with a PEEK(65) (low byte) and a PEEK(66) (high byte). With this information, we can develop a way to store these pointers for all the line numbers we want to reuse. We do this by storing these pointers in four arrays called D1, D2, D3, and D4. Each of these arrays is dimensioned to 25 items.(The number you use in your program depends on the number of data statements you will want to reuse.) Now let's look at the program lines that restore the pointers to any DATA line number.

For the following explanation, refer to the Commodore 64 program listing of *Uncle Larry's Ten Fiddle Tunes*. First, take a look at the subroutine that runs from lines 8200 through 8270. Next take a quick look at lines 200 through 260. Then examine one of the blocks of melody data items in lines 600 through 1360 (for example, the "Soldier's Joy" is played by lines 610 through 640). These three sets of lines all work together to RESTORE to a specified line.

In order to implement this method in our own programs, we must decide which DATA statements we want to restore. Whenever we find one, we enter the same flag (a number that will not be used as data in any data statement) as the first element in that data line. (If you look through the data statements of *Larry's Ten Fiddle Tunes*, you'll see that we used the number 100 as the flag for the data lines we want to restore.)

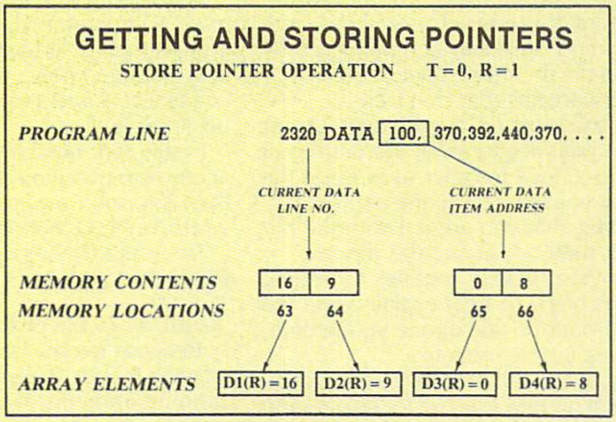
Once this is done, all we have to do is read through all the DATA statements and store the pointer values when we come across the flag. We do this at the beginning of the program in lines 200-260. In line 200, T is set equal to 0. Line 8200 of the subroutine shows a conditional jump based upon the value of T. This is how the program knows whether we are storing pointers or restoring pointers.

R is also set equal to 0 in line 200. This counter is incremented every time the Store Pointer portion of the subroutine is entered, and this is how the array subscript is incremented.

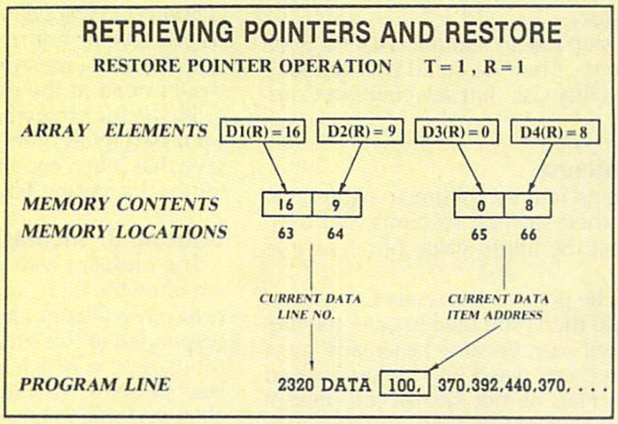
To restore a particular DATA line number, simply keep track of the value of the array subscript that corresponds to that line. Then set R equal to that value, set T equal to 1, and GOSUB 8200. Lines 220 and 230 read through data items that will not be restored in the program. Line 240 is the initial workhorse for the RESTORE. It checks for the flag in the data items and GOSUBs to 8200 if it comes across 100 at the beginning of a DATA line. Line 250 tells the program it has reached the end of the DATA statements, and line 260 does the Commodore RESTORE for entry into *Larry's Ten Fiddle Tunes*.

We need to attend to one more detail. We placed a flag into our DATA statement which we wanted the program to regard as an indicator, rather than as part of our data. We must make sure that the program doesn't try to "play" the flag value when it READS the DATA statements. Line 1920 instructs the program to disregard the number 100 when it appears in the data stream and go on to the next data item.

—*John Thrasher*

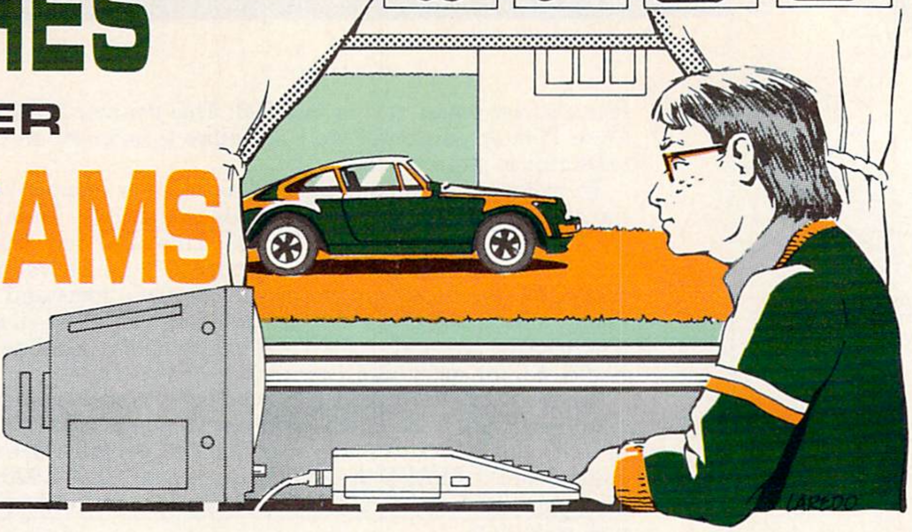


*Addresses 63 and 64 will contain the current DATA line number. The value of the high order byte (64), which is 9, is multiplied by 256: 256*9=2304. Add to this result the value of the low order byte (63), which is 16: 2304+16=2320. Thus 2320 is the value of the current DATA Line number.*



The current DATA item address value is not the value of the data item. It is the value of the address of the data item. The address of the data item is calculated the same way as the current DATA line number. Multiply the value of the high order byte (66), which is 8, by 256, to equal 2048. Add to this the value of the low order byte (65), which is 0: 2048 +0=2048. So 2048 is the address where the number 100, the flag, will be found.

PORSCHES
AND OTHER
PIPE DREAMS



Computer Assisted Savings Planning

By Joel S. Moskowitz, J.D.
and HCM Staff

When it comes time to make an expensive purchase—a car, stereo, refrigerator—those of us without large cash reserves often find ourselves comparing costly payment plans, trying to wheedle the money out of friends, or searching endlessly for super-bargains that don't exist.

If you lack the optimism to count on the Readers Digest Sweepstakes, or the ghoulish patience to await the demise of your rich uncle, you will have to look to other sources for the filthy stuff. You may find, after you exclude all the improbable, immoral and illegal alternatives, that you are left with the old fashioned, character-building method: *saving* the money.

Here is a program that provides helpful routines for setting up your own personal savings plan. To help explain what the program does, I will tell a sad-but-true tale of one young man, and how he used the routines in this program.

Jim had never been able to save much money. To encourage him, his father offered to sell him his new Porsche for only $4,000 if Jim could save the money in three years. The deal was too good to pass up, but Jim had only $500 in the bank, and his new job didn't pay much yet.

Anyone who anticipates a future expense (such as a balloon payment on a mortgage, or a trip to Hawaii next year) will see something familiar in Jim's story.

Could Jim make it? His first step was to find out what his $500 would amount to in three years. Then he could start thinking about saving the rest. For this first task, Jim selected the Compound Interest routine.

Routine 1: Compound Interest

Just as banks use a compound interest routine to figure how much interest they owe on their savings accounts, Jim used his home computer to project the future value of his savings in approximately 3 years.

When Jim ran the program, he pressed 1 to select Compound Interest from the menu. He was then prompted to enter the *present month* and then the *present year*. Because he already knew that the value of his savings account was $500, he just entered the current month and year. Had he not known the present value of his account, he could have entered the date on which he opened it and the original amount. Then the program would have computed his interest and brought his balance up to date.

Next, Jim entered his account's interest rate and number of compounding periods per year (in this case, 10% compounded 360 times per year). Both of these items required some reading of the fine print on the passbook.

The interest rate may be stated as the *effective rate* (the rate after compounding) or the *nominal rate* (the rate before compounding—a lower figure). Many institutions which offer *daily compounding* use a 360-day year. If only the effective rate of interest is known, the number of compounding periods should be set at 1.

Finally, Jim entered the *present amount* (the amount in his account as of the *present date*), which was $500. The program then informed him that in three years his account would be worth $674.90, noting that he would earn $174.90 in interest.

This was a big help in Jim's search for the $4000. He had *only* $3325.10 to go. Let's see how he could get it.

Routine 2: Level Payments

Because people rarely save the money *left over* at the end of each month, it is usually more effective to set aside a certain amount immediately upon receiving a paycheck. Jim needed to know how much to set aside each month in order to get his $3325.10 at the end of three years. The Level Payments routine gave him that answer.

As in the case of the Compound Interest routine, Jim entered the present and future dates, the rate of interest, and the number of compounding periods per year. Next he was asked how many payments per year he planned to make. Because Jim was paid monthly, he entered 12. Finally, he entered the amount he would need at the end of the three year period: $3325.10.

Jim ran the program and found that he had to set aside $78.87 each payday in order to reach his goal. Jim knew he couldn't save that much each month. Was he sunk? Not yet. He could try the Increasing Payments routine.

Routine 3: Increasing Payments

The problem with level payment plans is that the payments are often too high, and they do not take into account any ability to pay a greater amount later on. The mortgage industry has responded to this situation by introducing Graduated Payment Mortgages, in which the payments rise a fixed percentage each year, as one's salary (hopefully) rises. The same technique which allows people to buy houses can be applied to other large purchases they otherwise could not afford.

Under this plan, the payments start off lower, remain level for a year, and then rise by a set percentage for each subsequent year until the future date arrives. The initial payments are most profoundly reduced when the savings period extends over many years, as in the case of a 30-year mortgage. In such a case, the first year's payments are much lower than those of later decades.

There is a price to pay for the luxury of lower initial payments. Under the Level Payments plan, more money is saved during the early years, and this extra money goes right to work earn-

Note: TI users will recognize this article from the April, 1983 issue of 99'er Home Computer Magazine. *The HCM staff has developed versions of the* Saving *program for the Apple, IBM PCjr, C-64 and VIC-20 machines.*

ing interest. The Level Payments plan is therefore always cheaper than the Increasing Payments plan. And yet, this should stop no one from using the Increasing Payments plan if it would help. After all, it would be cheaper still to forget about payments altogether and put away a lump sum which will grow into the amount needed. It would not be practical, however, and that is what savings plans are all about.

Let's get back to Jim. He selected the Increasing Payments plan from the menu. The program asked him the same questions it asked in the Level Payments routine, as well as by what percentage he would like his payments to rise each year.

Jim figured that his salary would rise by 5% each year, so he entered 5. To his dismay he found that his payments were only lowered to $75.30 each month. Jim shouldn't have been surprised, though. After all, his payments at the end of the three-year period would be only a little more than 10% higher than they were at the beginning. In any case, he needed to give his project more thought.

Routine 4: Future Value, Fixed Payments

After figuring his expenses, Jim realized that the most he could save was $50 per month. How much would that give him at the end of three years?

The Future Value, Fixed Payments routine asked Jim the same questions as the Level Payments routine, except that instead of asking him how much money he needed, it wanted to know how much he was prepared to save at the beginning of each period. Jim entered $50. He was told that his payments plus the interest would add up to $2107.86 at the end of three years. This amount, plus the future value of his savings account, was more than $1200 short.

That left Jim with two basic options. He could either negotiate a better deal with his dad, or find a way to afford more savings. So what did Jim do? He was last seen driving around in a 1973 Toyota Corolla with a wrinkled rear end. Obviously not *everyone's* money worries can be solved by this program. But these routines will help define problems, and thereby aid home computer users in charting a path through the financial wilderness.

The IBM PCjr locates the cursor on the screen with a convenient command called LOCATE. This BASIC instruction allows you to position the cursor at any point on the screen simply by supplying the X and Y coordinates. Once the cursor has been repositioned, any statements which rely on cursor positioning, such as PRINT and INPUT, will be affected. [**Note:** This program will not give accurate results if you're using Cassette BASIC or BASICA on the IBM PC (jr's big brother) unless the BASIC program is compiled rather than run with the interpreter. See the IBM PC Home Computer Tech Note in this issue for details.—Ed.]

Continued on p. 119

The *Saving* program for the Commodore systems performs the same four routines as the other machines, with a fifth option that ends the program. The algorithms, defined functions and general format are identical to those used in the other versions. The major differences between the TI and Commodore programs stem from the Commodore BASIC's lack of statements like DISPLAY and ACCEPT. We therefore used the INPUT statement instead, and approximated the screen format by including error-checking (e.g., lines 410-430 on the VIC-20 and lines 440-460 on the C-64) coupled with the unique cursor movement within a Commodore PRINT statement. For example, if you answer 13 in response to the prompt for the present month, the prompt will repeat, and the cursor will flash in the same spot on the screen.

When trying to substitute the INPUT statement on the VIC-20, however, we discovered another problem. When the prompt accompanying the INPUT statement exceeded the length of the line (i.e., caused the cursor to flash on the following line), the computer returned the REDO FROM START message. Apparently, when the prompt appears on a different line from the INPUT, the prompt itself is viewed as part of the INPUT. We get a REDO FROM START message because only numeric characters are allowed when we specify a numeric variable. We solved the problem by simply making our prompts PRINT statements and following them up with INPUT statements.

The Commodore version lacks the option to change just one variable in a particular routine. TI BASIC with its ACCEPT and DISPLAY statements implemented this easily—but on the C-64, this would have been very difficult without resorting to machine code routines. On the VIC-20, it would have been impossible, due to the limited memory space.

Apart from changing the INPUT prompts to PRINT statements, the only other difference between the C-64 and the VIC-20 versions is the elimination of the REM documentation lines. Without this change, the program would not fit in the VIC-20.

C-64 continued on p. 144, VIC-20 continued on p. 145

The most distinctive feature in the Apple version of *Saving* is the way in which the cursor is moved to the desired position for the PRINT statement. The Apple has two commands that position text anywhere on the screen when you're using a 40-column format. (The 80-column format requires a POKE to memory location 36 with the column number.) The HTAB statement is capable of moving the cursor either left or right to any of the 40 columns on the screen. The VTAB function can place the cursor on any of the 24 lines on the screen without affecting its horizontal position. If, for example, the cursor were in row 3 and column 17, and you gave the command VTAB 15, then the cursor would move to row 15, column 17. Likewise, a subsequent command of HTAB 5 would move the cursor to column 5 on row 15. Once you have repositioned the cursor, you can PRINT from that position, or use the INPUT or GET statements. Any statement which is dependent on the position of the cursor will be affected by using the VTAB and HTAB functions.

SAVING (APPLE II Series)
Explanation of the Program

Line Nos.	
100-170	Program Header.
180-200	Title screen, Branch to subroutine to dimension arrays and define functions.
210-320	Display menu screen, input choice, branch to subroutine to get input, then branch to appropriate subroutine.
330-360	Upon return from subroutines, wait for return key to be pressed before continuing.
370-420	Compound interest subroutine.
430-490	Level payments subroutine.
500-660	Increasing Payments subroutine.
670-740	Future value fixed payments subroutine.
750-790	Subroutine for input to option 1.
800-840	Subroutine for input to options 2, 3, and 4.
850-890	Subroutine for input to options 2 and 3.
900-940	Subroutine for input to option 3.
950-990	Subroutine for input to option 4.
1000-1100	Subroutine for input to all options.
1110-1180	Subroutine to dimension arrays and define functions.
1190	End program.

Continued on p. 118

SAVING

APPLE II Series

```
100 REM **********
110 REM * SAVING *
120 REM **********
130 REM BY JOEL S.MOSKOWITZ
140 REM AND THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.1.1
170 REM APPLE II SERIES APPLESOFT
180 HOME : HTAB 16: VTAB 12: PRINT "SAV
    ING"
190 GOSUB 1110
200 INVERSE : VTAB 24: HTAB 4: PRINT "P
    RESS RETURN TO START THE PROGRAM":
    GOTO 340
210 NORMAL : HOME
220 HTAB 12: PRINT "SAVING": HTAB 1: VT
    AB 4: PRINT "CHOOSE:";: VTAB 6: PRI
    NT "1. COMPOUND INTEREST"
230 HTAB 8: VTAB 8: PRINT "2. LEVEL PAY
    MENTS"
240 HTAB 8: VTAB 10: PRINT "3. INCREASI
    NG PAYMENTS"
250 HTAB 8: VTAB 12: PRINT "4. FUTURE V
    ALUE"
260 HTAB 11: VTAB 13: PRINT "FIXED PAYM
    ENTS"
270 HTAB 8: VTAB 16: PRINT "5. END PROG
    RAM"
280 HTAB 1: VTAB 23
290 INPUT "YOUR CHOICE:";A: IF A < 1 OR
     A > 5 THEN GOTO 280
300 IF A = 5 THEN GOTO 1190
310 GOSUB 1000
320 ON A GOSUB 400,460,530,700
330 PRINT : PRINT "PRESS RETURN TO CONT
    INUE"
340 GET A$: IF A$ = "" THEN 340
350 IF A$ < > CHR$ (13) THEN 340
360 GOTO 210
370 REM *****************
380 REM COMPOUND INTEREST
390 REM *****************
400 GOSUB 780
410 HTAB 1: VTAB 18: PRINT "FUTURE SAVI
    NGS = $"; FN RD( FN COM(IN)): VTAB
    20: PRINT "TOTAL INTEREST = $"; FN
    RD( FN COM(IN) - PRE(X))
420 RETURN
430 REM **************
440 REM LEVEL PAYMENTS
450 REM **************
460 GOSUB 830: GOSUB 880:I = R / K:TY =
     Y + M / 12:TP = INT (NP * TY): IF
     TP = 0 THEN TP = 1
470 N = K / NP * TP:F = FN FV(N):N = K
     / NP:P = FN PV(N):TY = (M / (12 /
     NP) - INT (M / (12 / NP))) / NP
480 IF TY < > 0 THEN ND = ND / (1 + IN
     / (100 * K)) ^ (K * TY)
490 HTAB 1: VTAB 20: PRINT "LEVEL PAYME
    NTS = $"; FN RD( FN CP(P)): RETURN
500 REM *******************
510 REM INCREASING PAYMENTS
520 REM *******************
530 GOSUB 830: GOSUB 880: GOSUB 930:I =
     R / K:TY = Y + M / 12
540 IF TY > = 1 AND TY < = 25 THEN 57
    0
550 HTAB 1: VTAB 22: PRINT "TIME SPAN M
    UST BE FROM 1 TO 25 YEARS"
560 GET A$: GOSUB 1000: GOTO 530
570 X = 0:A = 0:CA(1) = 1:N = K / NP:P
    = FN PV(N):N = K:F = FN FV(N):Z =
     1:PRE(1) = FN G(F)
580 FOR Z = 2 TO Y + 1:CA(Z) = CA(Z - 1
    ) * (1 + RA / 100):PRE(Z) = FN G(F
    ): NEXT Z
590 IF Y = 0 THEN 640
600 FOR X = 1 TO Y:TY = TY - 1: IF TY =
     0 THEN 620
610 A = A + FN COM(IN): GOTO 630
620 A = A + PRE(X)
630 NEXT X
640 IF M > = 12 / NP THEN Z = X:N = K
    / NP:P = FN PV(N):N = K / NP * IN
    T (M / (12 / NP)):F = FN FV(N):A =
     A + FN G(F)
650 TY = (M / (12 / NP) - INT (M / (12
     / NP))) / NP: IF TY < > 0 THEN PR
    E(X) = CA(X):A = A + FN COM(IN)
660 PA = ND / A: HTAB 1: VTAB 22: PRINT
     "1st PERIOD PAYMENT =$"; FN RD(PA)
    : RETURN
670 REM *************************
680 REM FUTURE VALUE FIXED PAYMENT
690 REM *************************
700 GOSUB 830: GOSUB 980:I = R / K:Z =
    1:TY = Y + M / 12:TP = INT (NP * T
    Y): IF TP = 0 THEN TP = 1
710 N = K / NP:P = FN PV(N):N = K / NP
     * TP:F = FN FV(N):A = FN G(F)
720 TY = (M / (12 / NP) - INT (M / (12
     / NP))) / NP: IF TY < > 0 THEN PR
    E(X) = A:A = FN COM(IN)
730 HTAB 1: VTAB 20: PRINT "SAVINGS WIL
    L = $"; FN RD(A): RETURN
740 HOME : HTAB 1: VTAB 3: INPUT "PRESE
    NT MONTH (1-12):";PM: IF PM < 1 OR
    PM > 12 THEN GOTO 1000
750 REM *****************************
760 REM INPUT SUBROUTINE FOR OPTION 1
770 REM *****************************
780 HTAB 1: VTAB 16: INPUT "PRESENT AMO
    UNT:";PRE(X): IF PRE(X) < = 0 THEN
     GOTO 780
790 RETURN
800 REM ********************************
810 REM INPUT SUBROUTINE FOR OPTIONS 2
    , 3, AND 4
820 REM ********************************
830 HTAB 1: VTAB 16: INPUT "PAYMENTS PE
    R YEAR:";NP: IF NP < = 0 THEN GOT
    O 830
840 RETURN
850 REM ********************************
860 REM INPUT SUBROUTINE FOR OPTIONS 2
    , AND 3
870 REM ********************************
880 HTAB 1: VTAB 18: INPUT "AMOUNT NEED
    ED:";ND: IF ND < = 0 THEN GOTO 88
    0
890 RETURN
900 REM *******************************
910 REM INPUT SUBROUTINE FOR OPTION 3
920 REM *******************************
930 HTAB 1: VTAB 20: INPUT "% INCREASE
    PER YEAR:";RA: IF RA < 0 THEN GOTO
     930
```

APPLE II Series

```
940 RETURN
950 REM  ****************************
960 REM  INPUT SUBROUTINE FOR OPTION 4
970 REM  ****************************
980 HTAB 1: VTAB 18: INPUT "PAYMENT AMOUNTS:";CA(1): IF CA(1) < 0 THEN GOTO 980
990 RETURN
1000 HOME : HTAB 1: VTAB 3: INPUT "PRESENT MONTH (1-12):";PM: IF PM < 1 OR PM > 12 THEN 1000
1010 HTAB 1: VTAB 4: INPUT "PRESENT YEAR:";PY: IF PY < 1 OR PY > 9999 THEN GOTO 1010
1020 HTAB 1: VTAB 6: INPUT "FUTURE MONTH (1-12):";FM: IF FM < 1 OR FM > 12 THEN GOTO 1020
1030 HTAB 1: VTAB 7: INPUT "FUTURE YEAR:";FY: IF FY < 1 OR FY > 9999 OR FY < PY THEN GOTO 1030
1040 IF FY < PY OR FY = PY AND FM < = PM THEN VTAB 9: HTAB 1: PRINT "FUTURE IS SMALLER THAN OR EQUAL TO THE PRESENT YEAR. PLEASE REENTER.": GOTO 1020
```

APPLE II Series

```
1050 Y = FY - PY:M = FM - PM: IF M < 0 THEN M = M + 12:Y = Y - 1
1060 TY = Y + M / 12
1070 HTAB 1: VTAB 12: INPUT "PERCENT INTEREST:";IN: IF IN < = 0 THEN GOTO 1070
1080 R = IN / 100
1090 HTAB 1: VTAB 14: INPUT "COMPOUNDS PER YEAR:";K: IF K < = 0 THEN GOTO 1090
1100 RETURN
1110 DIM CA(25),PRE(25)
1120 DEF FN PV(N) = (1 - 1 / (1 + I) ^ N) / I
1130 DEF FN FV(N) = ((1 + I) ^ N - 1) / I
1140 DEF FN G(F) = F * CA(Z) / P
1150 DEF FN CP(P) = P * ND / F
1160 DEF FN COM(IN) = PRE(X) * (1 + IN / (100 * K)) ^ (K * TY)
1170 DEF FN RD(A) = INT (A * 100 + .5) / 100
1180 RETURN
1190 END
```

HCM

***SAVING . . .* from p. 117**

SAVING (IBM PCjr) Explanation of the Program

Line Nos.	
100-190	Program Header
200-270	DEFDBL variables, dimension arrays and define functions
280-330	Title screen.
340-420	Display menu screen, input choice.
430-550	Get input of DATA common to all routines; branch to appropriate subroutine.
560-580	Upon return from subroutines, wait for return key to be pressed before continuing.
590-640	Compound interest subroutine.
650-730	Level payments subroutine.
740-870	Increasing Payments subroutine.
880-1050	Future value fixed payments subroutine.
1060-1090	End Program.

IBM PCjr

```
100 REM **********
110 REM * SAVING *
120 REM **********
130 REM BY JOEL S. MOSKOWITZ
140 REM AND THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.1.1
170 REM PC CASSETTE BASIC, PC BASICA
180 REM PCjr CASSETTE BASIC
190 REM PCjr CARTRIDGE BASIC
200 DEFDBL A-W,Y
210 DIM CA(25),PRE(25)
220 DEF FNPV=(1-1/(1+I)^N)/I
230 DEF FNFV=((1+I)^N-1)/I
```

IBM PC & PCjr

```
240 DEF FNG=F*CA(Z)/P
250 DEF FNCP=P*ND/F
260 DEF FNC=PRE(X)*(1+IN/(100*K))^(K*TY)
270 DEF FNRD(A)=INT(A*100+.5)/100
280 REM ************
290 REM TITLE SCREEN
300 REM ************
310 KEY OFF:CLS:LOCATE 12,16:PRINT "SAVING":LOCATE 24,1:PRINT "PRESS ENTER TO BEGIN"
320 A$=INKEY$:IF A$="" THEN 320
330 IF A$<>CHR$(13) THEN 320
340 REM ***********
350 REM MENU SCREEN
360 REM ***********
370 CLS:LOCATE 1,39:PRINT "MENU SCREEN":LOCATE 4,1:PRINT "CHOOSE :":LOCATE 6,9:PRINT "1. COMPOUND INTEREST":LOCATE 8,9:PRINT "2. LEVEL PAYMENTS":LOCATE 10,9:PRINT "3. INCREASING PAYMENTS":LOCATE 12,9:PRINT "4. FUTURE VALUE"
380 LOCATE 13,12:PRINT "FIXED PAYMENTS":LOCATE 15,9:PRINT "5. END THE PROGRAM"
390 LOCATE 24,1:PRINT "YOUR CHOICE?";
400 A$=INKEY$:IF A$="" THEN 400
410 IF ASC(A$)<49 OR ASC(A$)>53 THEN 400
420 CHOICE=VAL(A$):IF CHOICE=5 THEN 1060
430 CLS:LOCATE 1,1:INPUT "CURRENT MONTH (1-12):";PM:IF PM<1 OR PM>12 THEN GOTO 430
440 LOCATE 3,1:INPUT "CURRENT YEAR :";PY:IF PY<0 THEN 440
450 LOCATE 5,1:INPUT "FUTURE MONTH (1-12):";FM:IF FM<1 OR FM>12 THEN 450
460 LOCATE 7,1:INPUT "FUTURE YEAR:";FY:IF FY<0 THEN 460
```

Continued on p. 144

WHAT IS LOGO

by Robert Ackerman
Managing Editor

As much as anything else, LOGO is a way of thinking. It is, of course, primarily known as a programming language and a part of a computer system. But it came into being specifically from attempts to aid beginners in computing to think about problems. As a result, it developed into a language that helps you attack problems in a structured way.

In its beginnings, LOGO used a robot that rolled across the floor on command from a computer console. On command, it changed direction, backed up, turned. On command, it also lowered a pen to a sheet of paper on the floor, and the pen traced out a pattern as the robot moved. Because of its domed shape, this robot became known as the *turtle*. Even after its evolution from a robot on the floor to a graphics character on a video display screen, the name stuck.

The form stuck as well. *Turtle graphics*, as it's sometimes called, gives an immediate and visual response to commands typed at the keyboard. Beginners can see instantaneously the result of a command. A beginner can build on that command, follow it with others and create shapes on the screen. After some experimentation, you might find, for example, that the following sequence of commands builds a square:

```
TELL TURTLE
PENDOWN
RIGHT 180
FORWARD 20
RIGHT 90
FORWARD 20
RIGHT 90
FORWARD 20
RIGHT 90
FORWARD 20
```

In some versions of LOGO, you may have to type TELL 0 or SHOWTURTLE to get the turtle on the screen initially.

If you now use the following commands to make two legs of a triangle, you have a house—a bit simple, but a house:

```
PENDOWN
LEFT 120
FORWARD 20
LEFT 120
FORWARD 20
```

Introduction

LOGO Times is an information resource for users who want to create their own *personal* languages—languages that will easily allow them to communicate with the computer in a totally new audiovisual realm of applied imagination, exploration, and self-discovery. The articles on these pages concern the use of the LOGO language, but readers do *not* need any additional software or equipment (or even a computer) to understand and learn from the material presented here.

If readers want to actually *experience* a LOGO environment, they will need a computer, the requisite software and/or cartridges, and any additional hardware required for a particular implementation. A disk drive is required for some LOGO implementations, but in other cases, a user's work may be saved on cassette tape, or copied into a notebook (for later re-keyboarding).

The varieties of LOGO we'll consider include—but are not limited to—Terrapin LOGO for the Apple II, II+ or IIe and the Commodore 64, TI LOGO for the TI-99/4A, and LOGO Computer Systems LOGO for the IBM PC and PCjr.

- **Apple:** Terrapin LOGO requires an Apple II, II+ or IIe with 64K of RAM, one disk drive with controller, and a blank, initialized disk.
- **Commodore 64:** Terrapin LOGO requires a Commodore 64 with a VIC-1541 Disk Drive and a blank, initialized disk.
- **TI-99/4A:** TI LOGO requires the TI LOGO or TI LOGO II cartridge and a compatible 32K memory expansion unit. A cassette recorder may be used for storage, but a compatible disk system is recommended for convenience.
- **IBM PC or PCjr:** LOGO Computer Systems LOGO requires the PC or PCjr with 128 bytes of RAM, one disk drive, and a blank, initialized disk.

In each issue, one or more of the articles may refer to or build upon the topics discussed in a previous article. It is therefore recommended that for maximum benefit and understanding, new readers obtain the appropriate back issues of *Home Computer Magazine* containing *LOGO Times* articles.

LOGO Listings

As you enter LOGO statements, the last thing you do at the end of every statement is to press [ENTER] on the TI and IBM (the key with the (↵) symbol), or [RETURN] on the Commodore 64 and Apple. This signals the system to begin a new line. In our typeset listings, single LOGO statements may carry over from one line to the next without ending. The end of a LOGO statement is marked with a curved arrow (↩) to indicate that you press [ENTER] or [RETURN] at that point.

Notice

LOGO Times is actively soliciting articles. Manuscripts should be typed double-spaced, and accompanied by a cassette tape or disk if containing any lengthy procedures or graphics.

Send all materials to:

LOGO Times Editorial Dept.
Home Computer Magazine
1500 Valley River Dr., Suite 250
Eugene, OR 97401

All mail directed to the Letters-to-the-Editor column (*Letters on LOGO*) will be published in accordance with the conditions set forth on *Home Computer Magazine's* Masthead page.

Our Contributing Editors

Henry Gorman, Jr. Roger B. Kirchner William M. Goodman Rich Haller

Logo Contents

Volume 4, Number 1

What is LOGO?

LOGO allows this exploration of shapes from the bottom up, starting from single commands typed in at the keyboard. Because of this immediate response to commands at the lowest level, LOGO enthusiasts say it has *no threshold*.

In addition to giving you an immediate response, LOGO also allows you to gather your discoveries together and name them. This lets you think of them as units. For instance, if you type:

TO SQUARE

you'll enter LOGO's edit mode. When you follow that with the commands for a square and then close off that unit by typing END, you have created a structure that will draw a square. Now, to get LOGO to draw that shape, you only have to return to the main LOGO screen and type SQUARE. This structure, beginning with TO and finishing with END, is called a *procedure*. Procedures are the basic programming structure in LOGO. If you now type TO TRIANGLE, follow that with the above commands for the two legs of a triangle and END, you have another procedure. Type SQUARE and then TRIANGLE. The turtle will draw the simple house on the screen.

But instead of merely typing these procedure names one after the other, you can organize them into yet another procedure. If you type TO HOUSE, followed by SQUARE, TRIANGLE and END, you have a procedure (HOUSE) that uses, or *calls*, other procedures. After you type HOUSE, LOGO first draws the square according to the commands in SQUARE, and then it makes the triangle from the commands in TRIANGLE.

Other languages are oriented towards program lines and line numbers; BASIC is a good example. Programs in those languages transfer from one section of the program to another according to line numbers, and program segments run from line number *xxx* to line number *yyy*. That tends to obscure the structure of the program and makes programs harder to read and understand. LOGO does away with BASIC statements such as a GOSUB 2000 (what's 2000?) and instead calls procedures by their names—NEWHOUSE or SKYSCRAPER or MOONRISING or whatever.

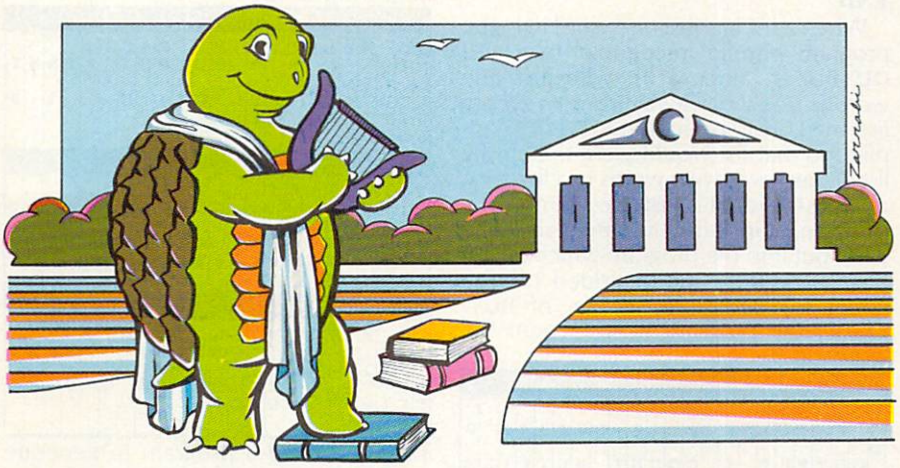
In addition to building from the bottom up, LOGO helps you build from the top down. Suppose, for instance, that you wanted to draw a cityscape. You could type in TO CITY. You know that to draw the city on the video display, it would probably be easiest to start at the edge. So you type in EDGE after TO CITY. Your city will have houses, so you can type in HOUSES. When you follow that with END, you've defined a procedure that calls two other procedures. But wait—aren't those two other procedures undefined? Yes, and if you try to run CITY before you define them, LOGO won't know what to do. But you now know what you want to do and what the general structure of your program will be.

Continued on p. 128

LYRICAL LOGO

Recursion Lets Loose the Poet in Your Console

by Henry Gorman Jr.
and the HCM staff



Because LOGO's graphics capabilities are so many and so easy to use, there is a tendency to overlook its other features. List handling is a case in point: By combining some of LOGO's list primitives—operations such as FIRST, BUTFIRST, (or the converse LAST, BUTLAST)—with recursive OUTPUT lines, we can easily write programs to reverse a list, alphabetize a list, or even compose poetry. (See the adjoining piece, "A Primer: Recursion and List Primitives.") The examples that follow have been adapted for Terrapin LOGO on the Apple IIe, Terrapin LOGO on the Commodore C-64 and LOGO Systems LOGO on the IBM PCjr. [*Program segments with no computer specified at the beginning are for all machines.—Ed.*] They will, I hope, demonstrate the powerful simplicity and list-manipulation potential of the language.

Verifying the presence or absence of a word in a list is a problem commonly encountered in list processing. The MIT LOGO group refers to this as the "MEMBER?" problem because the program answers the question, "Is a specified word a member of a specified list?" Some aspects of the program are obvious. For example, once the answer is obtained (whether TRUE or FALSE), it should OUTPUT to the user or to the program which called for the answer. It is also obvious that if the list is empty, the word is not in the list. Given just this much information, it is possible to frame a MEMBER? program for the Apple version of Terrapin LOGO. MEMBER? is a primitive in Terrapin LOGO for the Commodore and LOGO Systems LOGO for the IBM PCjr.

TERRAPIN LOGO — APPLE II Series

```
TO MEMBER? :WORD :LIST ↵
 IF :LIST = [ ] OUTPUT "FALSE ↵
 . ↵
 . ↵
 . ↵
END ↵
```

Papert notes that one way of solving a complex problem is to ignore the complex whole and focus on those parts which can easily be solved. [See *Mindstorms: Children, Computers, and Powerful Ideas* by Seymour Papert, Basic Books.] In the "MEMBER?" problem, if the *first* word in the list were the target word, then it would be easy to detect it and solve the problem using the LOGO primitive FIRST*, which returns the first word in a list:

TERRAPIN LOGO — APPLE II Series

```
TO MEMBER? :WORD :LIST ↵
 IF :LIST = [ ] OUTPUT "FALSE ↵
 IF FIRST :LIST = :WORD OUTPUT "TRUE ↵
 . ↵
 . ↵
 . ↵
END ↵
```

Now all that remains is finding a solution for those cases in which the word is in an *interior* position or is *absent* from the list. Were the word *second* in the list, the problem would be solved by adding a line using the LOGO primitive BUTFIRST*, which returns all but the first word in a list of words:

IF FIRST BUTFIRST :LIST = :WORD OUTPUT TRUE

since the *second* word in the list is the *first* word in a list which excludes the first word. Similarly, the third word will become the FIRST of the BUTFIRST of the BUTFIRST of the list, and the fourth word is the FIRST of the BUTFIRST of the BUTFIRST of the BUTFIRST of the list. It would be possible to write a separate line for each of those positions as well as the fifth, sixth, seventh or any other potential word position. However, a program that did this would quickly grow ponderous. Fortunately, in LOGO this is unnecessary. Notice that for each position an additonal BUTFIRST is all that is needed. The problem therefore requires only a *single* recursion line to complete the program:

TERRAPIN LOGO — APPLE II Series

```
TO MEMBER? :WORD :LIST ↵
 IF :LIST = [ ] OUTPUT "FALSE ↵
 IF FIRST :LIST = :WORD OUTPUT "TRUE ↵
 OUTPUT MEMBER? BUTFFIRST :LIST :WORD ↵
END ↵
```

Now when we run the program by typing MEMBER? "FOX, ? [A QUICK BROWN FOX] , the first stack checks to see if the list is empty or if the first word in the list matches the target word, FOX. Then it awaits the results of a second stack, which runs MEMBER? with the truncated list and the target word. The second stack then awaits the results of a third stack, which runs MEMBER? on BROWN FOX and "FOX. That stack then awaits the results of MEMBER? "FOX, FOX which returns "TRUE (from the match in the second line). "TRUE is

***FIRST returns the first word in a list of words, or the first letter in a list of words, or the first letter in a word. LAST returns the last letter in a list of words, or the last letter in a word. BUTFIRST returns all but the first word in a list of words, or all but the first letter in a word. BUTLAST returns all but the last word in a list of words, or all but the last letter in a word.**

The TI version of this article originally ran as LOGO Poet in two parts in Volume 1, Numbers 3 and 4, of 99'er Magazine.

returned to the second stack, which outputs "TRUE to the first stack, which outputs "TRUE to the program that first called it (or to top level). In the event that there were no matches, one of the stacks would eventually run MEMBER? on an empty list and would output "FALSE.

Another common problem is to count the number of words in a list of words. As before, this problem is solved by outlining the obvious elements of the solution and the simplest case.

TO COUNT :LIST
OUTPUT *some number*
END

The simplest case occurs when the list is empty.

TO COUNT :LIST
IF :LIST = [ ] OUTPUT 0
OUTPUT *some number*
END

When a list has just *one* word in it, the program should recognize that and OUTPUT 1. Since a list with just one word is one word away from an empty list, the LOGO operation BUTFIRST applied to that list would yield the empty list. If there were *two* words in a list, then obviously the list is just *two* words away from an empty list. If a recursive line were put into the program which (a) applied BUTFIRST and (b) added 1 to the count for every application of BUTFIRST, the program would count the words in the list.

TERRAPIN LOGO — APPLE II Series

```
TO COUNT :LIST↵
 IF :LIST = [] OUTPUT 0
↵
 OUTPUT ( COUNT BUTFIRS
T :LIST ) + 1↵
END↵
```

For another example, consider a program which will reverse a list. The simplest case would be a list with no words.

```
TO REVERSE↵
 IF :LIST = [] OUTPUT [
]↵
END↵
```

The next simplest case would be a list with just one word. For such a list we could have the program OUTPUT the SENTENCE or the word and an empty list.

```
TO REVERSE↵
 IF :LIST = [] OUTPUT [
]↵
 OUTPUT SENTENCE ( LAST
 :LIST ) ( REVERSE BUTL
AST :LIST )↵
END↵
```

This solution can be applied to longer lists as well!

For a final example, let's use LOGO to "write" random poetry. As a first effort at LOGO poetry, we'll attempt some "free verse" by instructing a poet to string words together randomly from a list we select. First, we will need a program like SELECT to output a selected item from a list.

TERRAPIN LOGO — C-64

```
TO SELECT :N :LIST↵
 IF :N = 1 OUTPUT FIRST
 :LIST↵
 OUTPUT SELECT :N - 1 B
UTFIRST :LIST↵
END↵
```

TERRAPIN LOGO — APPLE II Series

```
TO SELECT :N :LIST↵
 IF :N = 1 OUTPUT FIRST
 :LIST↵
 OUTPUT SELECT :N - 1 B
UTFIRST :LIST↵
END↵
```

LOGO SYSTEMS LOGO — IBM PCjr

```
TO SELECT :N :LIST↵
IF :N = 1 [OUTPUT FIRST
 :LIST]↵
OUTPUT SELECT :N - 1 BU
TFIRST :LIST↵
END↵
```

Then we need a program to generate random numbers for SELECT. Because LOGO's RANDOM primitive provides the integers through nine, if our list is less than ten, we can get a COUNT of it and use that COUNT.

TERRAPIN LOGO — C-64

```
TO NUMB :LENGTH↵
 MAKE "R RANDOM :LENGTH
↵
 OUTPUT :R + 1↵
END↵
```

A PRIMER: RECURSION AND LIST PRIMITIVES

It is easier to understand recursion in LOGO if one imagines that each LOGO program is a job for a contractor to perform. Each contractor is a specialist and can do only *one* job. Every contractor follows strict working rules; these rules say that when the contractor sees STOP, he must stop; when he sees OUTPUT, he must pass back some information and then stop. Of course, when a contractor reaches an END, he also stops. When a contractor sees the name of any LOGO program inside of the program he is completing, he subcontracts that job out to another contractor. Thus, in COUNT [A B C], the first contractor reads the first line of the program, but the condition isn't met, so he moves to line two. There he is told to OUTPUT 1+ the COUNT of [B C]. Since he can't do another program, he subcontracts the job. The subcontractor reads line 1 of COUNT, but since it doesn't apply, he reads line 2. He is told to OUTPUT 1+ the COUNT of [C]. He can't do that, so he also subcontracts the job. The third contractor notes that line 1 doesn't apply, and line 2 tells him to OUTPUT 1+ the COUNT of [ ]. He also must subcontract the job out, and so the fourth contractor reads line 1 of COUNT. Since the list is empty, he OUTPUTs 0 and passes the job back to the third contractor; he, in turn, adds 1 and then OUTPUTs 1. In a similar way, the second contractor adds 1 to that and OUTPUTs 2. Then, the first contractor adds 1 to that and OUTPUTs 3, which is the correct answer. With this explanation, you should now be able to analyze a program which gives you the answer to a number X raised to N power.

```
TO EXPONENT :X :N
.
.
.
END
```

```
TO EXPONENT :X :N
IF :N = 0 OUTPUT 1
.
.
.
END
```

```
TO EXPONENT :X :N
IF :N = 0 OUTPUT 1
IF :N = 1 OUTPUT :X
OUTPUT (EXPONENT :X :N - 1) * :X
END
```

TERRAPIN LOGO — APPLE II Series

```
TO NUMB :LENGTH
  MAKE "N ( RANDOM 8 )
  TEST ALLOF ( :N > 0 ) ( :N < ( :LENGTH + 1 ) )
  IFT OUTPUT :N
  IFF OUTPUT NUMB :LENGTH
END
```

LOGO SYSTEMS LOGO — IBM PCjr

```
TO NUMB :LENGTH
OUTPUT 1 + RANDOM :LENGTH
END
```

By first typing:

MAKE "LENGTH COUNT :LIST

we can then use NUMB for the value of LENGTH. If we then type:

TYPE SELECT (NUMB :LENGTH)
[*a list of words*]
PRINT1 SELECT (NUMB :LENGTH)
[*a list of words*]

the computer types one of the words in the list. We can write that as a program:

TERRAPIN LOGO — C-64

```
TO VERSE :LIST
  PRINT1 SELECT ( NUMB :LENGTH ) :LIST
END
```

TERRAPIN LOGO — APPLE II Series

```
TO VERSE :LIST
  PRINT1 SELECT ( NUMB :LENGTH ) :LIST
END
```

LOGO SYSTEMS LOGO — IBM PCjr

```
TO VERSE :LIST
TYPE SELECT ( NUMB :LENGTH ) :LIST
END
```

To turn this into a line of poetry, we should have a random number of such randomly selected words with a random number of spaces between words (E. E. Cummings's style), and then a carriage return:

TERRAPIN LOGO — C-64

```
TO SPACE
  REPEAT RANDOM 6 [PRINT1 CHAR 32]
END

TO LINE :LIST
  REPEAT RANDOM 6 [SPACE VERSE :LIST]
  PRINT SELECT ( NUMB :LENGTH ) :LIST
END
```

TERRAPIN LOGO — APPLE II Series

```
TO SPACE
  REPEAT ( RANDOM 8 ) [PRINT1 CHAR 32]
END
TO LINE :LIST
  REPEAT ( RANDOM 8 ) [SPACE VERSE :LIST]
  PRINT SELECT ( NUMB :LENGTH ) :LIST
END
```

LOGO SYSTEMS LOGO — IBM PCjr

```
TO SPACE
REPEAT RANDOM 6 [TYPE CHAR 32]
END
TO LINE :LIST
REPEAT 1 + RANDOM 6 [SPACE VERSE :LIST]
SPACE
PRINT SELECT ( NUMB :LENGTH ) :LIST
END
```

Note: PRINT1 or TYPE CHAR 32 puts the character with ASCII code 32, a *space*, on the screen.

If we want continuing lines of poetry, we can write a recursive program:

```
TO LINES :LIST
  LINE :LIST
  LINES :LIST
END
```

Now, putting this all together we get:

```
TO POET :LIST
  MAKE "LENGTH COUNT :LIST
  LINES :LIST
END
```

Now we can try converting POET into a program that produces either rhyming verse, blank verse, or a finite number of lines of verse. One way to modify POET to produce rhymed verse is to give it two different lists—one of words for the interior words of each line of verse, and another of rhyming words for the last word in each line. Then the program can be changed so that only rhyming words are placed in end positions.

TERRAPIN LOGO — C-64

```
TO SELECT :N :LIST
  IF :N = 1 OUTPUT FIRST :LIST
  OUTPUT SELECT :N - 1 BUTFIRST :LIST
END

TO GET :ARG
  OUTPUT SELECT ( NUMB ( COUNT :ARG ) ) :ARG
END

TO VERSE :LIST
  PRINT1 SELECT ( NUMB :LENGTH ) :LIST
END

TO LINE :LIST :RHYMES
 ;
 ; LINE MUST PUT RHYMES ONLY AT
 ; THE END OF EACH LINE
 ;
  REPEAT RANDOM 6 [PRINT1 CHAR 32 VERSE :LIST]
  PRINT1 CHAR 32
  PRINT SELECT ( NUMB :LENGTHR ) :RHYMES
 ;
 ; PUTS A RHYME AT THE END
 ;
END

TO LINES :LIST :RHYMES
 ;
 ; LINES MUST ACCOMMODATE TWO
 ; LISTS
 ;
  LINE :LIST :RHYMES
  LINES :LIST :RHYMES
END

TO POET :LIST :RHYMES
 ;
 ; THE SECOND LIST MUST NOW BE
 ; GIVEN THE PROGRAM
 ;
 MAKE "LENGTH COUNT :LIST
 MAKE "LENGTHR COUNT :RHYMES
 ;
 ; NECESSARY TO FIND OUT HOW
 ; MANY RHYMING WORDS THERE
 ; ARE
 ;
 LINES :LIST :RHYMES
END
```

TERRAPIN LOGO — APPLE II Series

```
TO POET :LIST :RHYMES
 ;
 ; THE SECOND LIST MUST NOW BE
 ; GIVEN THE PROGRAM
 ;
 MAKE "LENGTH ( COUNT :LIST )
```

Continued on p. 126

Lyrical LOGO ...from p. 125

TERRAPIN LOGO — APPLE II Series

```
MAKE "LENGTHR ( COUNT :RHYMES )
;
; NECESSARY TO FIND OUT HOW
; MANY RHYMING WORDS THERE
; ARE
;
LINES :LIST :RHYMES
END

TO LINE :LIST :RHYMES
;
; LINE MUST PUT RHYMES ONLY
; AT THE END OF EACH LINE
;
REPEAT ( RANDOM 8 ) [SPACE VERSE :LIST]
PRINT SELECT ( NUMB :LENGTHR ) :RHYMES
;
; PUTS A RHYME AT THE END
;
END

TO LINES :LIST :RHYMES
;
; LINES TAKES TWO LISTS
; AS INPUTS
;
LINE :LIST :RHYMES
LINES :LIST :RHYMES
END
```

LOGO SYSTEMS LOGO — IBM PCjr

```
TO LINE :LIST :RHYMES
;[]
;[LINE MUST PUT RHYMES ONLY AT]
;[THE END OF EACH LINE]
;[]
REPEAT 1 + RANDOM 6 [TYPE CHAR 32 VERSE :LIST]
TYPE CHAR 32
PRINT SELECT (NUMB :LENGTHR) :RHYMES
;[]
;[PUTS A RHYME AT THE END]
;[]
END

TO LINES :LIST :RHYMES
;[]
;[LINES MUST ACCOMODATE TWO]
;[LISTS]
;[]
LINE :LIST :RHYMES
TO ; :COMMENTS
END
LINES :LIST :RHYMES
END

TO POET :LIST :RHYMES
;[]
;[THE SECOND LIST MUST NOW BE]
;[GIVEN THE PROGRAM]
;[]
MAKE "LENGTH COUNT :LIST
MAKE "LENGTHR COUNT :RHYMES
;[]
;[NECESSARY TO FIND OUT HOW]
;[MANY RHYMING WORDS THERE]
;[ARE]
;[]
LINES :LIST :RHYMES
END
```

The problem of generating rhyming verse is one form of the problem of teaching the computer to write text which follows a specified rule (in this case each line must rhyme). The more general application of rules to text is nothing less than grammar. One of the grade school pupils in the Brookline project wrote a textbook rule program like POET which generated random sentences. After she saw the effects of changing parts of speech she exclaimed enthusiastically that she now understood what a noun was.

POET can also be quickly adapted to a sentence generator which young people can play with to make grammar meaningful.

TERRAPIN LOGO — C-64

```
TO SENTENCES
  PRINT [TYPE A LIST OF ARTICLES AND THEN PRESS RETURN.]
  MAKE "ART REQUEST
  PRINT [TYPE A LIST OF NOUNS AND THEN PRESS RETURN.]
  MAKE "NOUNS REQUEST
  PRINT [TYPE A LIST OF ADJECTIVES AND PRESS RETURN.]
  MAKE "ADJ REQUEST
  PRINT [TYPE A LIST OF VERBS AND THEN PRESS RETURN. NOW WATCH.]
  MAKE "VERBS REQUEST
  GRAMMAR :ART :NOUNS :ADJ :VERBS
END

TO GRAMMAR :ART :NOUNS :ADJ :VERBS
  PRINT1 GET :ART
  PRINT1 CHAR 32
  PRINT1 GET :NOUNS
  PRINT1 CHAR 32
  PRINT1 GET :VERBS
  PRINT1 CHAR 32
  PRINT1 GET :ART
  PRINT1 CHAR 32
  PRINT1 GET :ADJ
  PRINT1 CHAR 32
  PRINT1 GET :NOUNS
  PRINT ".
  REPEAT 300 [ ]
  GRAMMAR :ART :NOUNS :ADJ :VERBS
END

TO GET :ARG
  OUTPUT SELECT ( NUMB ( COUNT :ARG ) ) :ARG
END
```

TERRAPIN LOGO — APPLE II Series

```
TO SENTENCES
  PRINT [TYPE A LIST OF ARTICLES AND THEN PRESS RETURN.]
  MAKE "ART REQUEST
  PRINT [TYPE A LIST OF NOUNS AND THEN PRESS RETURN.]
  MAKE "NOUNS REQUEST
  PRINT [TYPE A LIST OF ADJECTIVES AND PRESS RETURN.]
  MAKE "ADJ REQUEST
  PRINT [TYPE A LIST OF VERBS AND THEN PRESS RETURN. NOW WATCH.]
  MAKE "VERBS REQUEST
  GRAMMAR :ART :NOUNS :ADJ :VERBS
END

TO GRAMMAR :ART :NOUNS :ADJ :VERBS
  PRINT1 GET :ART
  SPACE
  PRINT1 GET :NOUNS
  SPACE
  PRINT1 GET :VERBS
  SPACE
  PRINT1 GET :ART
  SPACE
  PRINT1 GET :ADJ
  SPACE
  PRINT1 GET :NOUNS
  PRINT ".
  WAIT 20
  GRAMMAR :ART :NOUNS :ADJ :VERBS
END

TO GET :ARG
  OUTPUT SELECT ( NUMB ( COUNT :ARG ) ) :ARG
END

TO WAIT :LIM
  MAKE "I 1
  A:
  MAKE "I ( :I + 1 )
  IF :I = :LIM THEN STOP
  ELSE GO "A
END
```

LOGO SYSTEMS LOGO — IBM PCjr

```
TO SENTENCES
PRINT [TYPE A LIST OF ARTICLES AND THEN PRESS ENTER.]
MAKE "ART READLIST
PRINT [TYPE A LIST OF NOUNS AND THEN PRESS ENTER.]
MAKE "NOUNS READLIST
PRINT [TYPE A LIST OF ADJECTIVES AND PRESS ENTER.]
MAKE "ADJ READLIST
PRINT [TYPE A LIST OF VERBS AND PRESS ENTER. NOW WATCH.]
MAKE "VERBS READLIST
GRAMMAR :ART :NOUNS :ADJ :VERBS
END

TO GRAMMAR :ART :NOUNS :ADJ :VERBS
TYPE GET :ART
TYPE CHAR 32
TYPE GET :NOUNS
TYPE CHAR 32
TYPE GET :VERBS
TYPE CHAR 32
TYPE GET :ART
TYPE CHAR 32
TYPE GET :ADJ
TYPE CHAR 32
PRINT GET :NOUNS
WAIT 20
GRAMMAR :ART :NOUNS :ADJ :VERBS
END

TO GET :ARG
OUTPUT SELECT (NUMB (COUNT :ARG)) :ARG
END
```

SENTENCES can be made a better grammarian by adding distinctions of number and gender where appropriate; it can be made a more sophisticated language generator if GRAMMAR is altered to allow for conjunctions and subordinate clauses. All of these changes and more can be programmed by students as they learn both the specifics of grammar[1] and the mathematics of LOGO. HCM

[1] *Papert would probably argue that most students know the grammar which schools attempt to teach, but that the students do not have verbal labels for syntactical rules and parts of speech and do not see the relevance of the labels once they are told them. A sentence generator program can make grammar "speech syntonic."*

What is LOGO? . . . **from p. 122**

You might want to tackle EDGE first. One solution is:

```
TO EDGE
PENUP
CLEARSCREEN
TELL TURTLE
RIGHT 90
FORWARD 120
RIGHT 90
END
```

Then to get your houses, you could define HOUSES as follows:

```
TO HOUSES
HOUSE
HOUSE
HOUSE
HOUSE
HOUSE
END
```

Let's modify the original SQUARE procedure as well, by removing the third line, RIGHT 180. (The two commands of RIGHT 90 in EDGE accomplish the same thing.) That will give you five houses—enough for a small city. Or will it? When you type HOUSES, you get part of a complex polygon: a semicircle of five interconnected houses. Between houses, you want to move the turtle to the location for the next house. You can do this with a procedure called, say, MOVE. Now HOUSES reads:

```
TO HOUSES
HOUSE
MOVE
HOUSE
MOVE
HOUSE
MOVE
HOUSE
MOVE
HOUSE
END
```

And what should MOVE look like? When the turtle moves, you don't want it to draw—the houses should be separate. So you have it pull its pen up. Next you want it to turn right 60 degrees. Now comes the actual move: FORWARD 5. To get the house right side up, you have to turn the turtle in the right direction: LEFT 90. MOVE has the following structure:

```
TO MOVE
PENUP
RIGHT 60
FORWARD 5
LEFT 90
END
```

The turtle is finally ready to draw the next house on the block.

This process of breaking the larger problem of CITY into EDGE, HOUSES and MOVE allows you to work on smaller, clearly-identified portions of the big problem. One of the developers of LOGO, Seymour Papert, calls these small portions "mind-sized bites." Each is a structure with a name; complete in itself, yet small enough that the whole of it can be grasped readily.

If you look back on the procedure CITY, one thing stands out: The procedure HOUSES is awkward, with its long string of repetitive commands. But what's worse, it gives you only five houses. What if you want four? Or six? Or two? Do you have to change it every time? Or is there an easier way?

There is an easier way. LOGO procedures allow you to give the information they need within the procedures themselves. If, for example, you wanted a city with six houses, you could alter the procedure to start its definition with this line:

```
TO CITY :N
```

N is a variable name; the colon indicates to CITY that it is to use the value stored in the variable. The name of the procedure now includes the information that the procedure needs more data before it can begin work. If you type:

```
CITY 6
```

LOGO recognizes this as the procedure with its value. In this case, you don't have to give N a value and then call the procedure using N. The procedure expects its name to be followed by a value, and it recognizes 6 as a value it can use. These bits of information that procedures need in order to work are called their *inputs*.

The next step should have the procedure draw the six houses, but to do this, you'll have to redefine HOUSES. Each time it runs, it should draw a house, move, draw a house, move, and so on, until it has put six of them on the screen. You can get the procedure to do this by redefining it as:

```
TO HOUSES :N
IF ( :N = 0 ) THEN STOP ELSE
   HOUSES ( :N - 1 )
HOUSE
MOVE
END
```

The second line of the procedure first checks to see if the value of N is zero; if it is, then it stops. IBM LOGO does not recognize THEN and ELSE. Instead you simply place brackets around the instructions like:

```
IF ( :N = 0 ) [STOP] [HOUSES ( :N - 1 )]
```

If N isn't zero, then HOUSES does something interesting: It calls itself. This process, called *recursion*, is one of the most powerful features of LOGO. In essence, LOGO makes a copy of the procedure it is presently in, and then runs that copy. It continues until it is told to leave the copy—in this case, by STOP or END. Then it runs the previous copy, until it is told to leave. It then runs the previous copy, until. . . , and so on. In this case, HOUSES calls itself with a value one less than the original value of N. It will continue to subtract 1 from N and call itself until N equals zero. Then it will begin to draw houses and move. You'll have to alter CITY to read:

```
TO CITY :N
EDGE
HOUSES :N
END
```

and then run it with a number—a 6, for instance.

By adding some PRINT statements to the procedure, you can see part of the recursion take place:

```
TO HOUSES :N
PRINT :N
IF ( :N = 0 ) THEN STOP ELSE
   HOUSES ( :N - 1 )
HOUSE
MOVE
PRINT :N
END
```

The first PRINT statement prints how many more times HOUSES is going to call itself. The second prints the number of houses the procedure has finished drawing.

If you want to see it run slowly, you can insert a delay after the first PRINT statement by having the program repeat an action—say raising and lowering its pen—50 times.

```
REPEAT 50 [PENUP PENDOWN]
```

Recursion finds many applications in many areas. Factorials, for example, or the Fibonacci series—two common mathematical functions—are readily calculated with recursive procedures.

If LOGO has a serious weakness, it is that individual statements can quickly become difficult to read if they become complex. At one point, the Texas Instruments LOGO manual notes that "parentheses help considerably in enabling the human eye to see the pattern [of a statement, and] unless you are very practiced, you should not write a complex expression without parentheses for fear of not being able to read it the next day." It's always helpful to group portions of a statement in parentheses, as in the example above.

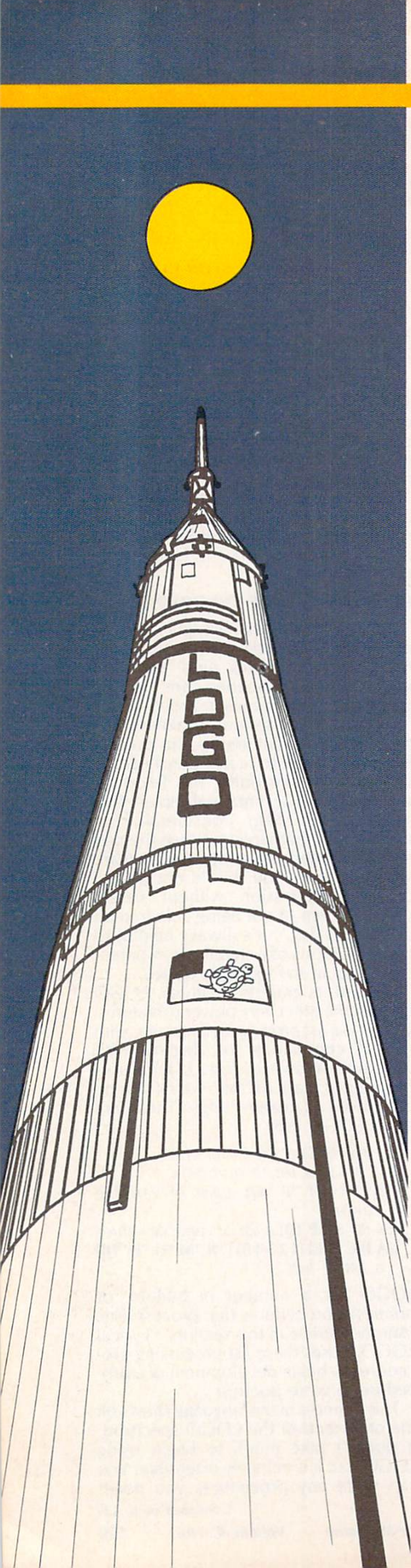
Beyond its graphics features, LOGO offers some other very powerful features. LOGO is a *list processing* language, with special commands for dealing with structures known as *lists*. LOGO uses brackets to mark off lists, which can be comprised of many things. These are some LOGO lists:

[A B C D E] *a list of letters*
[1 2 3 4 5] *a list of numbers*
["N "M "O "P "Q] *a list of variable names*
[:N :M :O :P :Q] *a list of variable values*
[[A B C D E] [1 2 3 4 5] ["N "M "O "P "Q]] *a list of lists*

LOGO has a number of built-in, or *primitive*, procedures that process lists. (Another article in this section, "Lyrical LOGO," uses these list processing procedures to begin development of a procedure to write poetry.)

This richness of the language shows off the other end of the LOGO spectrum: It doesn't take much to begin using LOGO, but it is infinitely extensible. You can write any procedures you need;

Continued on p. 137



LOGO SHOOTS FOR THE MOON

by Henry Gorman, Jr.
and the HCM staff

An important element of the LOGO philosophy is the idea that an intricate problem is composed of a hierarchy of smaller problems. Finding a problem's solution is, therefore, merely a matter of solving the smaller component problems. It is apparently easier to manage a large amount of information by subdividing it into smaller units than it is to attack the whole block *en masse*. For example, it is easier and faster to shop for groceries with a shopping list arranged hierarchically by type of food (dairy products, produce, cereals, beverages) than with a random list of items.

It is important that this structured approach to programming be developed through actual problem-solving experience. Students who have only textbook knowledge of the concept may develop unclear structures, which are worse than none at all. LOGO programming is a task which must be learned by doing. Often students can write structured programs without being able to verbalize what they are doing because their knowledge is active and procedural rather than passive and formal.

TI LOGO is tailor-made for the structured approach. Its procedures hide most of the program's details, revealing only the bare bones—the logical structure—of the program. The language also allows problems to be solved by programming with very little structure. Students can quickly compare programs with varying levels of structure and see the advantages of the more highly structured programs as they work on their own projects.

The Apollo Project

APOLLO is a student-written program depicting the lift-off, voyage, and touchdown of the Apollo space mission. One of its procedures, shown below, demonstrates a common beginning programmer's mistake: the tendency to strive for dense, short code. In LOGO procedures, clarity is more important than conciseness. I have, therefore, reworked the student's procedure to add a bit more structure and to make it more modular.

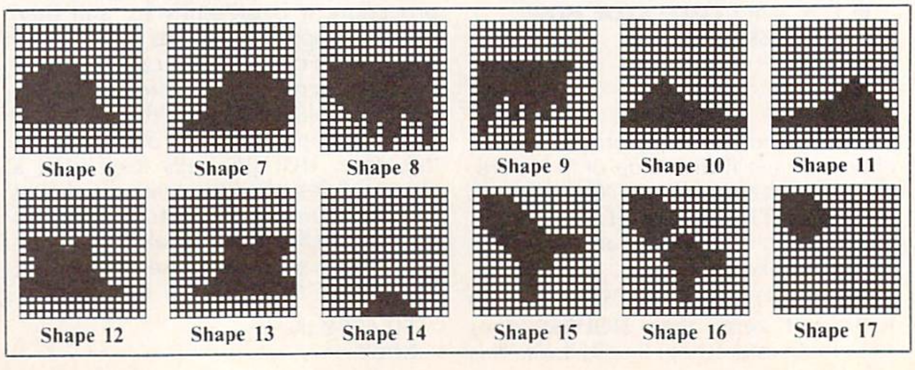
Student Version

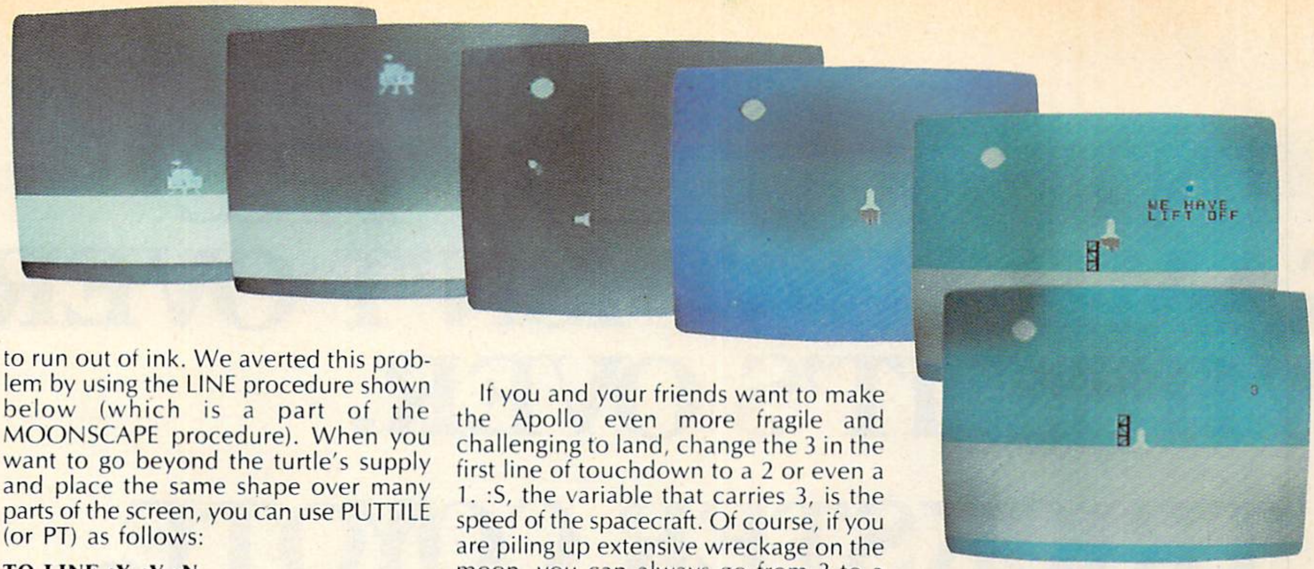
```
TO PREP
VANISH
EARTH
PT 101 15 11
PT 101 15 13
PT 102 15 12
TELL 0
CARRY 3
SC 1
SS 0
SH 0
HOME
TELL 1
CARRY 4
SC 15
SXY -80 80
END
```

Revised Version

```
TO PREP
VANISH
EARTH
MOON
GANTRY
ROCKET
END

TO MOON
TELL 1
CARRY 4
SC 15
SXY -80 80
END

TO GANTRY
TELL TILE 101
SC 1
PT 101 15 11
PT 101 15 13
PT 102 15 12
END

TO ROCKET
TELL 0
CARRY 3
SC 15
SS 0
SH 0
HOME
END
```

To run the program, you key in APOLLO. If you want to clear the sprites and characters from the screen, type in VANISH. The APOLLO program presents a common LOGO problem: The LOGO turtle has a limited number of tiles to use for drawing, so it's easy





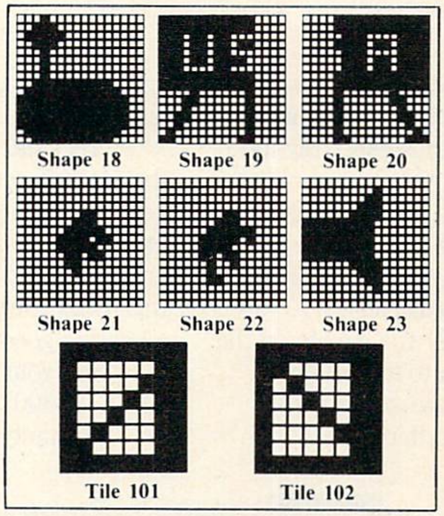
to run out of ink. We averted this problem by using the LINE procedure shown below (which is a part of the MOONSCAPE procedure). When you want to go beyond the turtle's supply and place the same shape over many parts of the screen, you can use PUTTILE (or PT) as follows:

```
TO LINE :X :Y :N
IF :X > 31 LINE 0 :Y + 1 :N STOP
IF :Y > :N STOP PT 1 :X :Y LINE :X
  + 1 :Y :N END
```

This procedure fills the screen from one side to the other, from row :Y to row :N (rows run down the screen 0-23). :N is the number of the last row filled. The procedure places the first tile in column :X of the first row (:Y). The 1 after PT stands for tile 1, but any tile number—or even a variable—could be substituted, as long as it's designed as a tile. It's also possible to rewrite the procedure to fill different blocks of the screen—from top to bottom, for example.

When the Apollo spacecraft settles toward the lunar surface, you will need to take over control from the computer to make the soft landing. Each time you press and release the space bar, you will trigger a burst from your braking rockets.

If you and your friends want to make the Apollo even more fragile and challenging to land, change the 3 in the first line of touchdown to a 2 or even a 1. :S, the variable that carries 3, is the speed of the spacecraft. Of course, if you are piling up extensive wreckage on the moon, you can always go from 3 to a larger number.



If you want to make additional landings, you can just type in the command LAND, and you will quickly find yourself hovering over the lunar surface.

Stepping Through a Program

When you want to double-check these modifications, you may run into a slight problem. Stepping a program (running it line by line) can be a helpful debugging device, but this facility is not built into the LOGO system. Although TI LOGO doesn't have a STEP primitive, you can write one:

```
TO STEP :A
PRINT [PRESS ENTER TO RUN THE
NEXT LINE.]
PRINT SE [NOW STEPPING THE PRO-
GRAM] :A
CHECKOFF BUTFIRST TEXT :A
END
```

Continued on p. 134

C64

Here are some suggestions for making a working version of APOLLO.

1. To start, you will need your LOGO disks and a formated disk with enough space on it (32 blocks free) to hold your completed program. The first step is to load the LOGO language.

2. Next, load the program SPRED from your utility disk (type READ "SPRED). SPRED will enable you to enter the graphics shapes that APOLLO needs. These shapes are divided into two packages for APOLLO, each of which is stored on disk and called up when it's needed.

When SPRED is loaded, you are ready to enter the shapes in the first sprite package, which will be called PREP. Make sure you enter the first package first, because we will carry over shapes 1 and 4 from the first package to the second package. . . that way you won't have to enter them twice. (With some modification the clouds of smoke from the takeoff serve well as the dust clouds of the lunar landing.)

In Table 1 you will find the first package of shapes, numbered 1 through

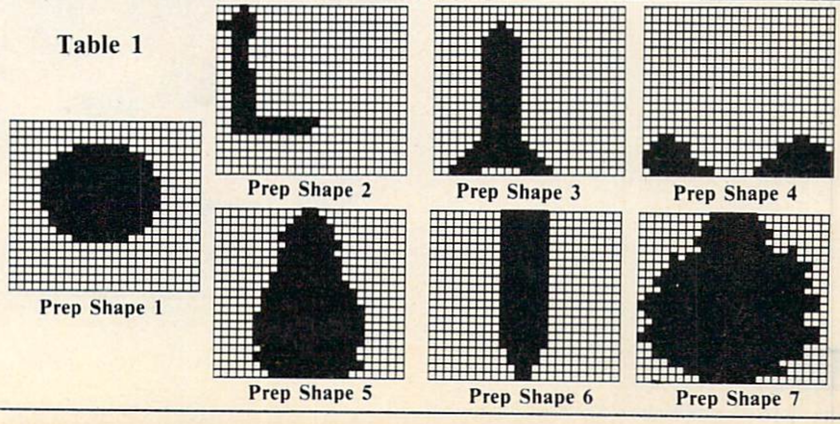
7. The first sprite you'll "talk to" is sprite 1 (sprite 0 is the turtle and can't be changed), so type TELL 1, and then type EDSH (edit shape). This will put you in the Sprite Edit mode. You will see the large box that will hold sprite shape number 1. Hold down the [SHIFT] key, press the [CLR] key, and you will have a clean sprite box to fill. (Look in your LOGO manual at pages S-10 to S-13 if you need any help entering the sprite shape.) After you've entered the shape, press the [STOP] key and you will return to TOPLEVEL, ready to enter the next shape.

When you have entered all of the shapes, put the disk that you intend to use for your APOLLO program into your drive, type SAVESHAPES "PREP, and the package will be saved.

The second package of sprite shapes is called ORBIT and can be found in

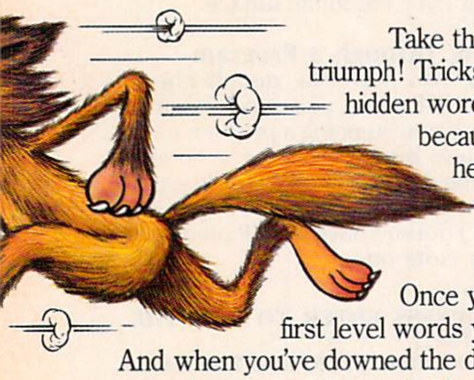
Continued on p. 133

Table 1



INTRODUCING THE GAME THAT ISN'T OVER WHEN IT'S OVER.

TRICKSTER COYOTE.



Take this clever critter to task and triumph! Trickster Coyote™ is a challenging hidden word game that's especially fun because the coyote's so cunning he can beat you if you blink!

It's fast, colorful and a natural to play with your friends.

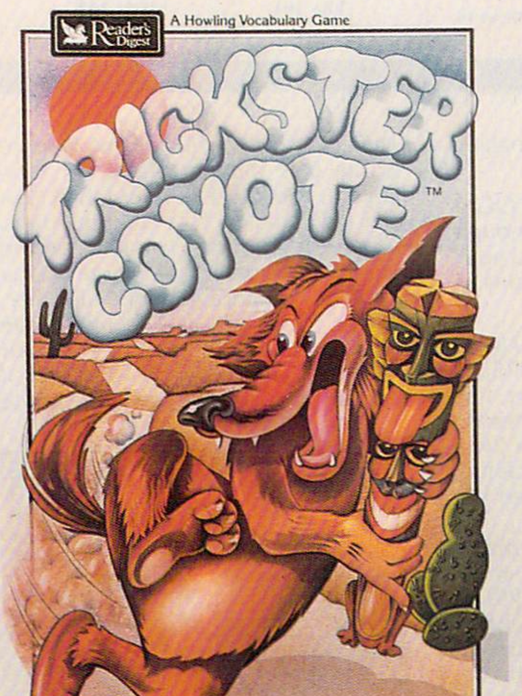
Once you clobber the coyote on the first level words you go on to the second. And when you've downed the dirty dog the second time around you're ready to trip him up on the third.

When all three battles are won, and you've used up the 300 words on the disk, you can add your own words to see if the coyote can beat you at your own game. This mutt loves punishment!

Trickster Coyote is a brain game that doesn't get boring because you keep on making new games. It grows as you do. Long after other games start to look old and tired, this rotten varmint will still give you a howling good game.

See your local dealer, or Call Toll-Free: 800/431-8800. (NY, AK, HI: 914/241-5727).

Warranty information available upon request by writing to: Reader's Digest Services, Inc., Microcomputer Software Division, Pleasantville, New York 10570.



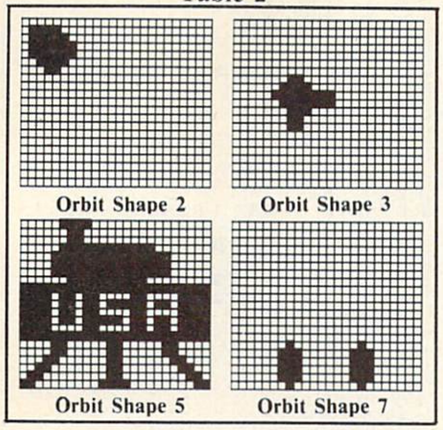
Reader's Digest®

Trickster Coyote is a trademark of The Reader's Digest Association, Inc. Apple is a registered trademark of Apple Computer, Inc. Apple® II, II Plus, IIe: 48K and disk drive.

LOGO Apollo . . .from p. 131

Table 2. Without clearing your workspace, enter the second group of sprite shapes (this will carry over some shapes from package 1 to 2.) When you've entered all the shapes, type

Table 2



Orbit Shape 2 — Orbit Shape 3 — Orbit Shape 5 — Orbit Shape 7

SAVESHAPES "ORBIT, and then type GOODBYE. Now you have a clean workspace for your procedures.

3.Now carefully enter the procedures from the listing. You should stop to save your workspace several times under the name APOLLO so that if you are interrupted or the machine is disturbed, you won't lose all of your work. After you have entered all of the procedures, merge the procedures named SPRITES from the utility disk with APOLLO in your workspace: First, load the utilities disk and type READ "SPRITES; then save the whole workspace as APOLLO. SPRITES contains the procedures that recall your saved sprites from the disk and put them to work in your program.

Just as in the TI version, if you want to make repeated attempts to land on the moon after your initial try, just type LAND, and you will soon find yourself over the lunar surface. You must start with a regular launch, though, before you can make repeated landing attempts. In the procedure LANDED you will find the line

IF :VELOCITY>4 THEN CRASHED

If you use a number smaller than 4, the spacecraft will be trickier to land; larger than 4 and you can make a bone-cruncher and still be congratulated by Mission Control. You can also change the amount of fuel in the lander by changing this line in the procedure LAND:

MAKE "FUEL 15

You're almost ready. Remember, at the appropriate time in APOLLO the shapes in PREP and ORBIT will be called from the disk, so make sure you have them in the drive while you're running the program. Enter APOLLO to begin. Goodbye, Major Tom, and good luck. HCM

TERRAPIN LOGO — C-64

```
TO APOLLO
  RESET 7
  PREP
  BLAST
  LIFT
  ATMOSPHERE
  STRATOSPHERE
  ORBIT
  LAND
END

TO PREP
  CLEARTEXT
  READSHAPES "PREP
  EARTH
  MOON
  GANTRY
  ROCKET
  MESSAGE.LAUNCH
  LAUNCHKEY
  CLEARTEXT
END

TO EARTH
  TEXTBG 0 BG 11
  CS
END

TO GANTRY
  TELL 2
  HOME
  RT 90 FD 13 LT 90
  BK 65
  PC 10
  ST
END

TO ROCKET
  TELL 3
  HOME
  BACK 59
  PC 1
  ST
END

TO MESSAGE.LAUNCH
  TEXTCOLOR 10
  CURSOR 9 22
  PRINT [WE ARE READY TO LAUNCH]
  REPEAT 900 [ ]
  CURSOR 10 23
  PRINT [PRESS THE SPACE BAR]
  REPEAT 500 [ ]
  CURSOR 20 21
END

TO LAUNCHKEY
  IF RC = CHAR 32 THEN STOP
  TELL 7 BIGX PC 1
  LAUNCHPAD
  TELL 3 HT
  IGNITION.EXPLOSION
  CLEARTEXT
  CURSOR 8 22
  PRINT [YOU ABORTED THE MISSION!]
  REPEAT 500 [ ]
  TELL 7 HT
  TELL 4 HT
  TOPLEVEL
END
```

TERRAPIN LOGO — C-64

```
TO CHECK :CHTR
  TELL 7 ST REPEAT 100 [ ] HT
  IF :POSITION > 90 THEN TELL 4 ST
  MAKE "FUEL :FUEL - 1
  CURSOR 19 22
  PRINT :FUEL
  MAKE "VELOCITY :VELOCITY - 2
  TELL 4 HT
  MOVE :VELOCITY
END

TO DUST.3
  TELL 4 PU
  HT FD 28 BIGY ST
END

TO DUST.2
  TELL 4 PU
  LT 90 HT FD 10 ST RT 90
  BIGX
  REPEAT 300 [ ]
END

TO CRASHED
  DUST.2
  DUST.3
  RESET 7
  PRINT [YOU HAVE CRASHED!]
  TOPLEVEL
END

TO LAND
  RESET 7
  CLEARTEXT
  MOONSCAPE
  TELL 5 FD 120 PC 6 ST RT 180
  TELL 7 FD 120 PC 1 RT 180
  MAKE "FUEL 15
  CURSOR 3 22 TEXTCOLOR 10
  PRINT [FUEL REMAINING: ]
  CURSOR 19 22
  PRINT :FUEL
  MAKE "VELOCITY 1
  MAKE "POSITION 0
  DUST.1
  MOVE :VELOCITY
END

TO MOONSCAPE
  TEXTBG 1 BG 0
  SPLITSCREEN
  CS
END

TO ATMOSPHERE
  RESET 7 FULLSCREEN
  BG 6
  MOON
  TELL 6 LT 90 FD 3 RT 90
  BK 98 PC 10 ST TELL 3 BACK 70 ST
  MOTION.6
END
```

TERRAPIN LOGO — C-64

```
TO MOTION.7
  REPEAT 30 [TELL 3 FD 3 TELL 2 FD 3]
END

TO MOTION.8
  TELL 3 PC 1 TELL 2 PC 1 PC 10 TELL 3
  PC 10
  REPEAT 15 [TELL 3 FD 3 TELL 2 FD 4]
END

TO MOTION.9
  REPEAT 20 [TELL 3 FD 3 TELL 2 FD 4 LT 1]
END

TO MOTION.6
  REPEAT 40 [TELL 6 FD 5 PC 2 PC 10 TELL 3 FD 5 ]
END

TO MOON
  TELL 1
  LT 55 FD 170
  PC 1
  ST
END

TO STRATOSPHERE
  RESET 7 FULLSCREEN
  BG 0
  MOON
  TELL 6 LT 90 FD 3 RT 90
  BK 90 TELL 3 BACK 70
  LT 2 ST TELL 6 LT 2 ST
  MOTION.6
END

TO MOVE :VELOCITY
  IF :POSITION > 177 THEN LANDED
  IF ALLOF ( :FUEL > 0 ) ( RC? ) THEN CHECK RC
  IF :POSITION > - 20 THEN TELL 5 FD :VELOCITY
  IF :POSITION > - 20 THEN TELL 7 FD :VELOCITY
  MAKE "POSITION :POSITION + :VELOCITY
  MAKE "VELOCITY :VELOCITY + 0.5
  MOVE :VELOCITY
END

TO DUST.1
  TELL 4 PU
  BACK 59
  PC 1
END

TO RESET :N
  TELL :N
  PU HT HOME
  SMALLX SMALLY
  IF :N = 0 THEN STOP
  RESET :N - 1
END
```

Continued on p. 134

LOGO Apollo . . . from p. 131

```
TO CHECKOFF :STUFF
CALL RC "B
IF FIRST :STUFF = [ ] STOP
IF FIRST :STUFF = SE :A [ ] STEP :A
PRINT SE [NOW RUNNING ] FIRST
:STUFF
RUN FIRST :STUFF
CHECKOFF BUTFIRST :STUFF
END
```

To STEP a program, first load the program, and then key in (or load from cassette or diskette) the two procedures above, STEP and CHECKOFF. The STEP and CHECKOFF procedures will be a part of your program even while you SAVE and RECALL. To use the STEP procedure, assume your program is called READ. Give the command STEP "READ. Note that CHECKOFF is called from within STEP, so the one command makes both procedures run. STEP can be modified to allow for the STEPPING of programs inside of STEPPED programs, or to allow for programs with variable input.

Now that we have explored the steps for structured programming as well as a structure for stepping programs, you are ready to run your modified APOLLO. Just keep in mind that all those cliches of the "divide and conquer" ilk apply when programming in LOGO. You must take many tiny steps before you can take that "giant step for mankind" from your APOLLO craft. HCM

TI-LOGO — TI-99/4A

```
TO APOLLO
PREP
BLAST
LIFT
ATMOSPHERE
STRATOSPHERE
ORBIT
LAND
END
```

TI-LOGO — TI-99/4A

```
TO ATMOSPHERE
TELL 5
REPEAT 48 [FLAME ]
TELL 5
SXY 0 (-72)
TELL 0
SC 15
SXY 0 (-60)
TELL [0 5 ]
SH 0 SS 3
END

TO STRATOSPHERE
CS
CB :BLUE
TELL 1
CARRY 4
SC 15
SXY -80 80
TELL 5
REPEAT 103 [FLAME ]
TELL 0
SXY 0 (-60)
TELL 5
SXY 0 (-72)
TELL [0 5 ]
SH 0
SS 3
CB :BLACK
END

TO ORBIT
SECTION
EJECT
END

TO LAND
TELL [0 1 2 3 4 5 ]
VANISH
CB 1
MOONSCAPE
TELL 0
CARRY 18
SC 4
SXY -20 95
TELL 1
CARRY 19
SC 4
SXY -23 80
TELL 2
CARRY 20
SC 4
SXY -16 80
TELL [0 1 2 ]
SH 180
CALL 3 "S
SS :S
CHECK
END
```

LOGO Apollo . . . from p. 133

TERRAPIN LOGO — C-64

```
TO DOWN :N
  IF :N < 48 THEN STOP
  CURSOR 20 22
  PRINT CHAR :N
  REPEAT 900 [ ]
  DOWN :N - 1
END

TO IGNITION
  TELL 4 PU
  LT 90 FD 4 RT 90
  BACK 59
  PC 1
  ST
  REPEAT 900 [ ]
  LT 90 HT FD 10 ST RT 9
0
  BIGX
  REPEAT 300 [ ]
  MESSAGE
  HT FD 28 BIGY ST
END

TO MESSAGE
  TEXTCOLOR 1
  CURSOR 16 22
  PRINT [IGNITION]
END

TO COUNTDOWN
  REPEAT 900 [ ]
  CURSOR 19 22
  TEXTCOLOR 1
  PRINT "10
  REPEAT 900 [ ]
  CURSOR 19 22
  PRINT CHAR 0
  DOWN 57
  CURSOR 20 22
  PRINT CHAR 0
  TEXTCOLOR 6
END
```

TERRAPIN LOGO — C-64

```
TO LIFT
  MOTION.1
  MOTION.2
  MOTION.3
  MOTION.4
END

TO MOTION.1
  REPEAT 29 [TELL 3 FD 1
  TELL 6 FD 1 PC 8 PC 2]

END

TO MOTION.2
  TELL 5 HT RT 90 FD 12
LT 90 SMALLX ST
  REPEAT 13 [TELL 5 FD 2
  TELL 3 FD 2 TELL 6 FD
2 PC 8 PC 2]
END

TO MOTION.3
  TELL 7 HT RT 90 FD 12
LT 90 SMALLX ST
  REPEAT 13 [TELL 3 FD 2
  TELL 6 FD 2 PC 8 PC 2]

END

TO MOTION.4
  REPEAT 20 [TELL 3 FD 4
  TELL 6 FD 4 PC 8 PC 2]

END

TO MOTION
  TOPLEVEL
END

TO ORBIT.START
  PC 10
  BK 120 RT 90 FD 140
  LT 135 ST
END
```

TI-LOGO — TI-99/4A

```
TO EARTH↵
TELL TILE 1↵
SC 14↵
LINE 0 14 23↵
END↵

TO MOON↵
TELL 1↵
CARRY 4↵
SC 15↵
SXY - 80 80↵
END↵

TO GANTRY↵
TELL TILE 101 SC 1↵
PT 101 15 11↵
PT 101 15 13↵
PT 102 15 12↵
END↵

TO COUNTDOWN↵
WAIT 60↵
PT 49 25 8↵
PT 48 26 8↵
DOWN 57↵
END↵

TO IGNITION↵
TELL [2 3 ]↵
SC 15↵
SXY - 13 ( - 3 )↵
TELL 2↵
CARRY 6↵
TELL 3↵
CARRY 7↵
SXY 13 ( - 3 )↵
WAIT 30↵
END↵

TO MESSAGE↵
PT 73 20 8↵
PT 71 21 8↵
PT 78 22 8↵
PT 73 23 8↵
PT 84 24 8↵
PT 73 25 8↵
PT 79 26 8↵
PT 78 27 8↵
END↵

TO SECTION↵
TELL [0 5 ]↵
SS 3↵
SH 315↵
TELL 5↵
CARRY 0↵
TELL 0↵
CARRY 15↵
SXY 0 ( -65 )↵
TELL 5↵
SXY 3 ( -68 ) WAIT 400↵
END↵
```

TI-LOGO — TI-99/4A

```
TO BURST↵
REPEAT 10 [WAIT 5 TELL
2 CARRY 12 TELL 3 CARRY
 13 WAIT 5 TELL 2 CARRY
 6 TELL 3 CARRY 7 ]↵
END↵

TO FLAMES↵
TELL [2 3 ]↵
SC :WHITE↵
TELL 2↵
CARRY 10↵
TELL 3↵
CARRY 11↵
TELL 4↵
CARRY 14↵
SC :RED↵
SY - 3↵
REPEAT 3 [BURN :WHITE :
YELLOW ]↵
TELL 4↵
REPEAT 5 [BURN :WHITE :
RED ]↵
END↵

TO MOTION↵
TELL [2 3 4 ]↵
CARRY 0↵
TELL 5↵
CARRY 8↵
SC :RED↵
HOME↵
SY - 11↵
TELL [0 5 ]↵
SH 0↵
SS 2↵
TELL 5↵
REPEAT 3 [PICKUP ]↵
END↵

TO MOONSCAPE↵
TELL TILE 1↵
SC 15↵
LINE 0 17 23↵
END↵

TO CHECK↵
WAIT 15↵
TEST RC?↵
IFT CALL :S - 2 "S CALL
 RC "B↵
CALL :S + 1 "S↵
TELL [0 1 2 ]↵
SS :S↵
TELL 1↵
TEST YCOR > 90↵
IFT OUTOFFUEL↵
IFF TEST YCOR < - 23↵
IFT TOUCHDOWN↵
IFF CHECK↵
END↵
```

Continued on p. 136

TERRAPIN LOGO — C-64

```
TO BLAST↵
 COUNTDOWN↵
 MESSAGE↵
 IGNITION↵
 CLEARTEXT↵
 BURST↵
END↵

TO LAUNCHPAD↵
 BK 58 LT 90↵
 FD 17 RT 90 ST↵
END↵

TO BURST↵
 TELL 3↵
 FD 1 REPEAT 300 [ ]↵
 TELL 6↵
 PC 2↵
 BK 63 LT 90 FD 4 RT 90
 ST↵
 TELL 5↵
 BIGX PC 1↵
 LAUNCHPAD↵
 TELL 7↵
 BIGX PC 1↵
 LAUNCHPAD↵
 TELL 3↵
 REPEAT 6 [FD 1 REPEAT
300 [ ] ]↵
 LIFT.MESSAGE↵
 TELL 3↵
 REPEAT 16 [FD 1 REPEAT
 200 [ ] ]↵
 CLEARTEXT↵
END↵

TO LIFT.MESSAGE↵
 TEXTCOLOR 1↵
 CURSOR 12 22↵
 PRINT [WE HAVE LIFTOFF
]↵
 REPEAT 800 [ ]↵
END↵
```

TERRAPIN LOGO — C-64

```
TO LANDED↵
 IF :VELOCITY > 4 THEN
CRASHED↵
 DUST.2 ST DUST.3 ST↵
 PRINT [CONGRATULATIONS
, COMMANDER]↵
 HT↵
 REPEAT 1800 [ ]↵
 TELL 5 HT BK 20 BIGX B
IGY ST↵
 TOPLEVEL↵
END↵

TO IGNITION.EXPLOSION↵
 TELL 4 PU↵
 LT 90 FD 4 RT 90↵
 BACK 59↵
 PC 1↵
 ST↵
 REPEAT 90 [ ]↵
 LT 90 HT FD 10 ST RT 9
0↵
 BIGX↵
 REPEAT 30 [ ]↵
 HT FD 28 BIGY ST↵
END↵

TO ORBIT↵
 RESET 7 FULLSCREEN↵
 READSHAPES "ORBIT↵
 TELL 1 I EFT 75 FD 120
ST PC 1↵
 BIGX BIGY ST↵
 TELL 3 FD 3↵
 ORBIT.START↵
 TELL 2↵
 ORBIT.START↵
 MOTION.7↵
 MOTION.8↵
 MOTION.9↵
END↵
```

HCM

LOGO Apollo . . . from p. 135

TI-LOGO — TI-99/4A

```
TO LIFT.MESSAGE ↵
PT 87 20 7 ↵
PT 69 21 7 ↵
PT 32 22 7 ↵
PT 72 23 7 ↵
PT 65 24 7 ↵
PT 86 25 7 ↵
PT 69 26 7 ↵
PT 76 20 8 ↵
PT 73 21 8 ↵
PT 70 22 8 ↵
PT 84 23 8 ↵
PT 32 24 8 ↵
PT 79 25 8 ↵
PT 70 26 8 ↵
PT 70 27 8 ↵
END ↵

TO FLAME ↵
WAIT 5 ↵
CARRY 9 ↵
WAIT 5 ↵
CARRY 8 ↵
END ↵

TO TOUCHDOWN ↵
IF :S > 3 CRASH ↵
TELL [0 1 2 ] ↵
SS 0 ↵
END ↵

TO VANISH ↵
TELL :ALL ↵
CARRY 0 ↵
SXY 126 96 ↵
SH 0 ↵
SS 0 ↵
SC 0 ↵
CB 7 ↵
CS ↵
NOTURTLE ↵
END ↵

TO OUTOFFUEL ↵
VANISH ↵
PRINT [YOU ARE OUT OF C
ONTROL ] ↵
PRINT [AND OUT OF FUEL
] ↵
PRINT [I'M SORRY ] ↵
END ↵
```

TI-LOGO — TI-99/4A

```
TO DOWN :N ↵
WAIT 60 ↵
PT 32 25 8 ↵
PT :N 26 8 ↵
IF :N < 49 STOP ↵
DOWN :N - 1 ↵
END ↵

TO BURN :FIRST :SECOND ↵
WAIT 5 ↵
SC :FIRST ↵
WAIT 5 ↵
SC :SECOND ↵
END ↵

TO PICKUP ↵
WAIT 5 ↵
CARRY 9 ↵
WAIT 5 ↵
CARRY 8 ↵
END ↵

TO FIRE ↵
WAIT 5 ↵
CARRY 22 ↵
WAIT 5 ↵
CARRY 21 ↵
END ↵

TO CRASH ↵
VANISH ↵
PRINT [YOU HAVE CRASHED
] ↵
PRINT [A BILLION DOLLAR
 CRAFT ] ↵
END ↵

TO LINE :X :Y :N ↵
IF :X > 31 LINE 0 :Y +
1 :N STOP ↵
IF :Y > :N STOP ↵
PT 1 :X :Y ↵
LINE :X + 1 :Y :N ↵
END ↵
TO ROCKET ↵
TELL 0 ↵
CARRY 3 ↵
SC 15 ↵
SS 0 ↵
SH 0 ↵
HOME ↵
END ↵
```

TI-LOGO — TI-99/4A

```
TO EJECT ↵
TELL [0 5 ] ↵
SS 3 ↵
SXY - 50 ( - 15 ) ↵
TELL 0 ↵
SC 15 ↵
WAIT 100 ↵
CARRY 16 ↵
WAIT 100 ↵
CARRY 17 ↵
TELL 5 ↵
CARRY 21 ↵
SC :RED ↵
TELL 3 ↵
SXY - 50 ( - 20 ) ↵
CARRY 23 ↵
SC 15 ↵
SH 180 ↵
SS 1 ↵
TELL 5 ↵
REPEAT 25 [FIRE ] ↵
END ↵

TO PREP ↵
VANISH ↵
EARTH ↵
MOON ↵
GANTRY ↵
ROCKET ↵
END ↵

TO BLAST ↵
COUNTDOWN ↵
IGNITION ↵
MESSAGE ↵
BURST ↵
FLAMES ↵
END ↵

TO LIFT ↵
MOTION ↵
LIFT.MESSAGE ↵
END ↵
```

HCM

What is LOGO? . . . from p. 129

after they're written, they're treated by the system exactly like the LOGO primitives. Neither has precedence. LOGO advocates call it a system with no threshold and no horizon.

In addition, LOGO's error messages are not cryptic and curt. They may not always be entirely clear, but they point you to your problem more closely than those in some other languages. The LOGO system was developed with the understanding that we would make programming errors, and that the system should automatically help us find the causes of our errors. For instance, if you type CITU instead of CITY, LOGO will respond:

TELL ME HOW TO CITU

If you forget to give HOUSES an input, a number that it can use in its processing, LOGO responds:

TELL ME MORE

In LOGO, you can look at CITY, fix your error and run the procedure again immediately. This is the *interactive* nature of LOGO.

LOGO is a very clear language to work with because of these features: its low threshold; its use of procedures that divide a program into comprehensible, self-contained segments; its friendly, informative error messages; its unlimited horizon; and its interactive system.

HCM

Have No Fear . . . from p. 109

characters in length, but if it is less than six, it must be filled out to six with blanks. The name is given using the TEXT Directive, and must be added in the location where we told the computer the information about our program would be (location >7FE0).

```
7020 71A6 AORG >7FE0
7FE0      TEXT 'TEST  '
7FE6 0000 ■
```

And finally, we add the starting point of the program (in our case, >7D00):

```
7FE6 7D00 DATA >7D00
7FE8      END
```

Then type END and exit the Assembler. Choose the Run option of the Mini Memory menu, type TEST and press [ENTER]. The asterisk will start moving across the screen.

Just one other thing: If, in the first line of our program there is a label—for example, 7D00____ NT LI R0,1—then these two lines would be equivalent:

```
7FE6      DATA >7D00
7FE6      DATA NT
```

It will be best to read the procedure step-by-step several times to understand it thoroughly, and work on the computer as you go along. Ultimately, you will see that it is a logical procedure.

Here is a summary of the steps to follow in making a REF/DEF table entry:

a) Use AORG to get to the First Free Address of the Module. Check the value there.
b) Check the value in the Last Free Address of the Module in the same way.
c) Subtract the value in >701C from the value in >701E. If the number is eight or larger, proceed.
d) Change the value in >701C to the value at the end of your program using the DATA Directive.
e) In the same way, change the value in >701E to a value eight bytes smaller than its present value.
f) Jump to the value you added in part (e) and write the program name (six characters long, including blanks) using the TEXT Directive.
g) At your ending location, use DATA to tell the computer the starting place of your program (either a memory location or a label).
h) Exit the Assembler using the END Directive and run the program.

In this third part of the series, I have tried to give you some more background information about this new language you are learning, to help you understand the advantages and limitations of the *Line-by-Line Assembler* more thoroughly. I have also shown how to save your program by name, using a very simple program as an example. You will notice that all the programs we have seen so far have been quite simple and straightforward, with a minimum of instructions. I selected these programs on purpose to keep everything as clear as possible until you had more knowledge before launching into more complicated ones.

HCM

Editing Apple Programs

What's a person to do? You've written a 62-line program to calculate the possible number of quasars in the universe, but when you RUN it to test your programming, the computer comes back with a nasty little message. Instead of the number of quasars, it says:

?TYPE MISMATCH ERROR IN 47

So you LIST line 47, a multiple-statement line that fills three lines on the screen. After a tedious and painstaking search, you see that you've inadvertently typed I$ instead of I%, and as a result, the program tries to put an integer into a string variable.

You can fix this error by typing in the line number at the present cursor position and then retyping the entire line. If the line is short—say, 47 I% = I% + 1—that's the quickest solution. But when you're looking at a long, multiple-statement line—like you might find in a 62-line program to calculate the number of quasars in the universe—you stand a good chance of making some other error while correcting the first.

MODIFY EXISTING TEXT

Fortunately, the Apple IIe system provides another way. When you move the cursor across the screen to the right with the right arrow key, the cursor is, in essence, reading the screen. The trick is to get it to read the line you want to fix, along with the corrections you want to make. To do this in the easiest way, first LIST the line you want to fix. The cursor will reappear on the screen after the prompt, below the line you've just LISTed. If you now press the [ESC] key, you can move it anywhere on the screen with the arrow keys, and while it's in this ESCape mode, the cursor doesn't read the screen. (If you have an active 80-column card, the cursor will have a little cross in it).

Stay in the ESCape mode, use the arrow keys to move the cursor over the leftmost digit of the line number of the line you've just LISTed, and then press [ESC]. (If the 80-column card is active, the cross in the cursor disappears). Now when you move the cursor to the right, it reads—or copies—the characters it crosses on the screen, exactly as if you'd typed them in from the keyboard. When you get to the offending character—the $—simply type in the % over it, and run the cursor across the rest of the line with the right arrow key. After it passes the last character, hit [RETURN]. If you LIST the line again, you'll see that your correction has been made.

INSERTION

Inserting is a bit trickier. To insert in a program line, LIST the line, hit [ESC], move the cursor over the leftmost digit of the line number and hit [ESC] again. Move the cursor (solid in the 80-column mode) to the point where you want to insert. Hit [ESC]. Move the cursor (with cross in the 80-column mode) up one line (or two or three lines), to a blank area of the screen. Hit [ESC]—now you've got the solid cursor again with the 80-column card active. Type the characters you want to insert. Hit [ESC] again and move the cursor (with cross in the 80-column mode) back to exactly the same position where you left the program line. Hit [ESC]—you've got the solid cursor again. Now move the cursor past the end of the line and press [RETURN]. When you LIST the line, the new characters will appear where you wanted to insert them.

DELETION

Deletion is much simpler: When you get to the characters you want to "erase," hit [ESC] (with an active 80-column card, you now have the cursor with the cross in it), move the cursor across those characters and hit [ESC] again. After the end of the line, hit [RETURN]. When you LIST the line, the characters will be gone.

TECH NOTES

Quoted strings can trip you up if you're not careful. If a program line wraps from one screen line onto the next, it will automatically indent on the next screen line. If you are typing a long quoted string and you let the cursor wrap from one screen line to the next, it will copy those blanks where the line indented as part of your quoted string. To avoid this, hit [ESC] when you pass the last character on the first line and hit [ESC] again when the cursor is over the first character in the next line. That way it skips the blanks. The cursor won't copy the blanks if they're not in a quoted string.

Now that you know how to edit your program and have corrected all the typos, back to the quasar computations. Let's see: RUN and. . . what's that? ?OVERFLOW ERROR IN 59. Hmmm. . .

—*Robert Ackerman*

APPLE EDITING

In the example below, we've initially set I equal to 757, but what we really want is to set I equal to the absolute value of I plus J. The following steps make that change. (If you have on Apple 80-column Text Card, you'll get the cross in the cursor. Otherwise you have to remember whether you're in the ESCape mode or not.)

• *STEP 1: list the line to be edited*

```
] LIST 40
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = 757: FLAG = 0
] █
```

• *STEP 2: push ESC and move cursor up to the beginning of the line with the arrow keys*

```
] LIST 40
[+]0 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = 757: FLAG = 0
```

• *STEP 3: push ESC and move cursor, reading the line to the location of change*

```
] LIST 40
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = █57: FLAG = 0
```

• *STEP 4: push ESC and move cursor up to the location above intended change*

```
] LIST 40                                          [+]
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = 757: FLAG = 0
```

• *STEP 5: push ESC and type required change*

```
] LIST 40                                          ABS(I)+J █
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = 757: FLAG = 0
```

• *STEP 6: push ESC and move cursor back to the beginning of the change*

```
] LIST 40                                          [+]BS(I)+J
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = 757: FLAG = 0
```

• *STEP 7: move cursor down on to the original line*

```
] LIST 40                                          ABS(I)+J
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = [+]57: FLAG = 0
```

• *STEP 8: move cursor over and past the area to be changed*

```
] LIST 40                                          ABS(I)+J
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = 757[+]FLAG = 0
```

• *STEP 9: push ESC when cursor is over the next correct entry and read to the end of the line*

```
] LIST 40                                          ABS(I)+J
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = 757: FLAG = 0█
```

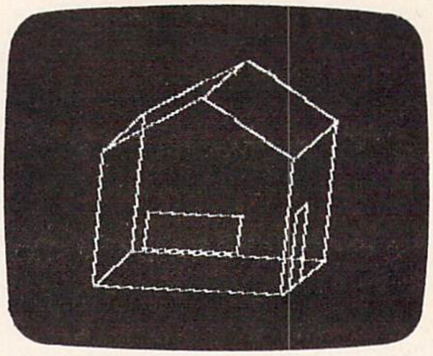
• *STEP 10: push RETURN and relist the line to verify the change*

```
] LIST 40                                          ABS(I)+J
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = 757: FLAG = 0
] LIST 40
40 FOR J = I TO (6-I): A$ = B$+A$: NEXT J: I = ABS(I)+J: FLAG = 0
] █
```

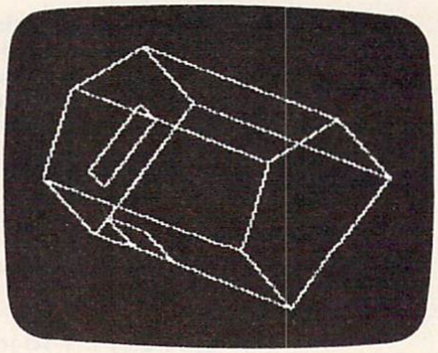
Eric A. Martin '83

3D-IIe:
Apple Graphics in Three Dimensions

by Michael D. Brownsworth



Line drawings can be represented in three dimensions with the *Applesoft 3-D Graphics* program.



Three dimensional figures can be rotated for a view from any angle.

Three-dimensional graphics is one of the most fascinating areas for beginning Apple programmers to explore. Several excellent three-dimensional graphics software packages are already available on the market, but these products are expensive and often difficult to use. This article provides an easy and inexpensive alternative—a BASIC program which demonstrates the principles of three-dimensional rotation, and even lets you design your own objects to rotate. Using this high-resolution graphics program, you can create the 3-D image of an object and then rotate it for a view from any angle.

3-D Realism

The representation of a three-dimensional object on a two-dimensional surface like your Apple's display screen is known as three-dimensional projection. This idea is simple. If we were to suspend a cube-shaped wire frame between a light source and a flat plane, the shadow cast would be a two-dimensional representation of the three-dimensional cube (see Fig. 1). Our program simulates this physical process, although the object exists only as a mathematical description within the computer. You tell the computer which angles to change, and it calculates a new projection and displays it on the screen.

In this program, all three-dimensional objects are like the wire-frame cube, made up of lines and showing all surfaces, including rear surfaces. Although methods do exist for eliminating surfaces that would be hidden from view, for our purposes, a wire-frame representation will be fine.

We must first establish a coordinate system representing three-dimensional space for the computer's internal representation of the object. The Cartesian (or rectangular) coordinate system consists of three perpendicular axes, traditionally labeled X, Y, and Z. The X and Y axes form a two-dimensional plane parallel to the display screen, in which the X axis is horizontal and the Y axis vertical. The Z axis, perpendicular to the XY plane, provides the dimension of depth. Any point in the coordinate system may be identified by using ordered triplets of the form (X,Y,Z). The origin, the place where the three axes intersect, is identified by (0,0,0).

Apple's high-resolution graphics screen layout is a two-dimensional XY coordinate system, and we can make use of it with one minor adjustment. The screen's two dimensional origin (0,0) is at the top left corner, rather than the bottom left. We can easily convert the inverted axis to the conventional direction within the program. Thus, with the Apple's bottom left corner (0,160) converted to (0,0), we can think of the screen as a window into the top right quadrant of our three-dimensional coordinate system, and all X and Y values become positive.

Crop Rotation

This program is designed specifically to rotate an object about the X, Y, and Z axes. You cannot see the object move during this rotation. Nor can the program "translate" the object, i.e., move the object along the X, Y or Z axis. Translation along the Z axis would result in a zoom effect, in which the object would appear to change distance from the viewer. With translation, it would even be possible to "enter" an object.

But since translation may place some lines of an object out of the screen boundaries, a program with this capacity must have some means of determining the portions of the object that should be displayed. This is known as "line-clipping." Our program, which is intended as an introduction to three-dimensional graphics, has no line-clipping algorithm. Our concern will be confined to three-dimensional rotation, which is intriguing in itself.

The program will display and rotate three predefined objects: a cube, a pyramid, and a house. The information describing each object (called its database) is stored in data statements in the program. You may also easily create and display objects of your own—we'll show you how to do it later.

The program is modular; that is, it assigns each specific task to a separate subroutine. The subroutines that do most of the work are placed first to increase execution speed. Let's take a brief look at the program flow.

After an object is selected in the subroutine 960, the program feeds the database for that object into arrays X, Y, and Z, using the subroutine beginning at line 1190, 1320, or 1480, depending on the object chosen. The arrays represent the endpoints of each line that forms the object. The lines are named by the pairs of points they connect and are placed into another array, PC( ).

In the subroutine starting at line 790, the user is asked to enter the amount of rotation (in degrees) for each axis. Then control is passed to the main driver in the subroutine in line 680, which calls the subroutines that perform the calculations.

First, the degrees entered are converted to radians to be used in the calculations. Then the centerpoint of the object is found (lines 560-670). The subroutines starting in lines 350, 430, and 490 each rotate the object on one of the three axes. Each point's new position is found, and the array containing its coordinates is updated. The subroutine at line 250 is called to translate the three-dimensional representation to a two-dimensional projection of line endpoints, and to store them in arrays XP( ) and YP( ). Finally, the subroutine at line 190 plots the object on the screen, using the PC( ) array to determine which points to connect.

Running The Program

After you type in the program, or before you reload it, you must reset LOMEM (the starting address of the program in memory) from 2048 to 16384, where the program's large array area will not conflict with the high-resolution screen display. To do this from direct mode, type: POKE 103,1: POKE 104,64: POKE 16384,0: POKE 16385,0: POKE 16386,0. Then you may run the program. Next month we will show you an EXEC file that sets LOMEM, then runs the program.

Actually, the program will run fine without resetting LOMEM if you change line 1130 from SIZE = 100 to SIZE = 20. This decreases the number of points that can be stored in the arrays, and therefore, the complexity of the objects that can be displayed. The cube, pyramid and house are rather simple, and SIZE = 20 is more than they require. But later you might want to create objects that are fairly complex (such as a house with furniture) and the added size will come in handy

When you run the program, the first thing you will probably notice is that it is s-l-o-w. This is because Applesoft, like most interpreted BASICs, is rather slow, especially when calculating expressions that use the SIN and COS functions. The program speed is dependent primarily upon the number of axes rotated and the complexity of the figure. Most executions will take about 3 or 4 seconds. There are several ways to speed things up. For example, you could use a utility that "crunches" programs—removes remarks, renumbers with low sequential line numbers, and replaces long variable names with single-letter variables. Or you could use one of the several Applesoft compilers, such as Microsoft's TASC, that give you about a ten-fold increase in speed.

Next, you may notice that it is often difficult to identify an object's orientation. Since the objects are wire-frame, with all lines visible, the viewer often does not have enough information to determine which surface is closest. This can lead to some surprising optical illusions. For example, the object may seem to turn inside-out before your eyes. To make the display less ambiguous, a short line has been added to the front faces of both the cube and the pyramid.

Create Your Own Objects

If you get bored with the cube, pyramid, and house, you will be happy to learn that next month's article will feature an easy-to-use editor that you can use to create your own 3-D objects. Instead of laboriously entering the X,Y,Z coordinates for each point of every line segment, you will be able to create the object on screen simply by moving the cursor around. Adventurous souls who just can't wait for next month's editor can create new objects using DATA statements, as we did with the cube, pyramid and house. It's easy—much like a child's connect-the-dots drawing, only in three dimensions.

It is helpful to use graph paper to plot your object's coordinates. You might plot first on the object's XY plane, then draw the YZ or XZ planes to add the third dimension. If you want to center the object on your screen, use (140,80,0) as the object's centerpoint. Keep in mind that the lower left corner of the screen (just above the four text lines at the bottom) is (0,0) on the XY axes.

No matter how you plot the object, you'll need to determine the X, Y, and Z coordinates for each endpoint of a single line or intersection of two or more lines. Starting at line 1650, copy the object's coordinates into a DATA statement. Count the number of points and the number of lines connecting the points in your object. Enter these values into the variables PNTCOUNT and LINECOUNT.

Although there are a number of ways to draw the object, some are more logical and efficient than others. For example, it is better to continue a line than to stop and start again with separate points, although sometimes this cannot be avoided. Next, enter the pairs of end points to be connected into the second DATA statement. If point 1 is connected to 2, and 2 is connected to 3, 3 to 4, 4 to 1, then 5 to 3, etc., you would enter: DATA 1,2,2,3,3,4,4,1,5,3 etc. Set the DATA list pointer to the first element in your DATA statement, which you can do by counting all the elements in all the preceding DATA statements (not including your new ones) and setting the ADVANCE variable equal to this number (to make it easy the first time, use 193). Update the object menu subroutine at 960 so that the program recognizes your database. Do this by inserting your new choice between line numbers 1030 and 1040. Change the message in line 1040 to include your object as one of the choices. In line 1060 change the number after the greater-than sign (>) to the new number of choices. There is sufficient array space to create an object with up to 100 points, enough for a fair degree of complexity. By way of comparison, the database for the house uses only 18 points.

This program, along with the editor to be featured in next month's article, will give you a chance to play with the exciting possibilities of three dimensional graphics.

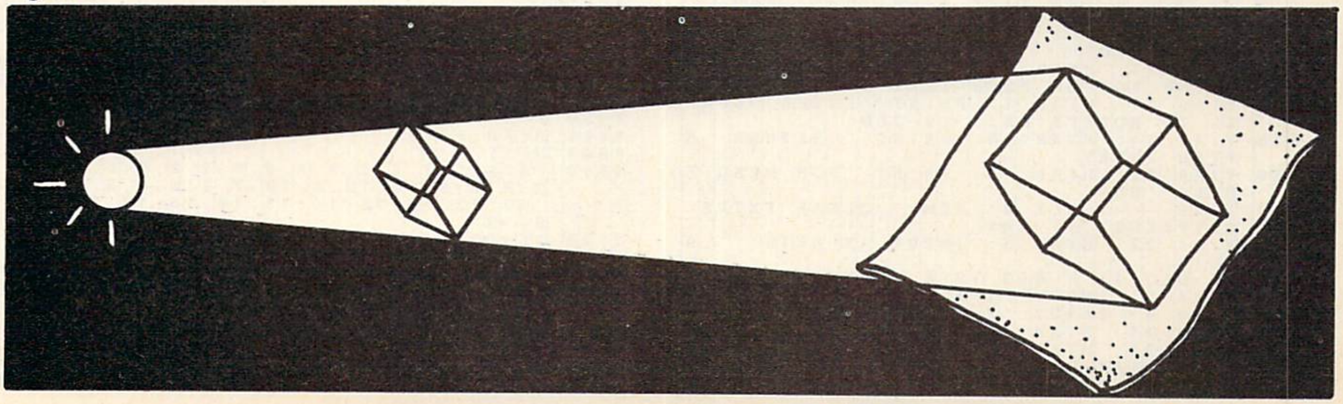
APPLESOFT 3-D GRAPHICS

Explanation of the Program

Line nos.	
100-170	Program Header
180	Branch to Initialization
190-240	Subroutine to plot Object on Screen
250-340	Subroutine to Translate 3-D to 2-D
350-420	Subroutine for X-axis rotation
430-480	Subroutine for Y-axis rotation
490-550	Subroutine for Z-axis rotation
560-670	Subroutine to find objects center
680-780	Main driver subroutine
790-950	Input subroutine
960-1110	Subroutine to display object menu
1120-1180	Initialization routine
1190-1310	Cube database subroutine
1320-1470	Pyramid database subroutine
1480-1630	House for routines to be added
1640-1990	Space for routines to be added
2000-2020	Routine to exit program

Continued on p. 142

Figure 1



APPLE II Series

```
100 REM ********************
110 REM * APPLESOFT 3-D *
120 REM * GRAPHICS *
130 REM ********************
140 REM BY M. D. BROWNSWORTH
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.1.1
170 REM APPLE II SERIES APPLESOFT
180 GOTO 1120
190 REM PLOT OBJECT ON SCREEN
200 HGR : HCOLOR= 3
210 FOR N = 1 TO LINECOUNT * 2 STEP 2
220 HPLOT XP(PC(N)),YP(PC(N)) TO XP(PC(
N + 1)),YP(PC(N + 1))
230 NEXT N
240 RETURN
250 REM TRANSLATE 3-D TO 2-D
260 FOR N = 1 TO PNTCOUNT
270 XP(N) = (X(N) * VZ) / (Z(N) + VZ)
280 YP(N) = (Y(N) * VZ) / (Z(N) + VZ)
290 NEXT N
300 FOR N = 1 TO PNTCOUNT
310 XP(N) = ((VZ - Z(N)) * XP(N)) / VZ:
 IF XP(N) < 0 THEN XP(N) = 0
320 YP(N) = ((VZ - Z(N)) * YP(N)) / VZ:
 IF YP(N) < 0 THEN YP(N) = 0
330 NEXT N
340 RETURN
350 REM X-AXIS ROTATION
360 FOR N = 1 TO PNTCOUNT
370 YR = (Y(N) - YCTR) * COS (RAD) - (
Z(N) - ZCTR) * SIN (RAD) + YCTR
380 ZR = (Z(N) - ZCTR) * COS (RAD) + (
Y(N) - YCTR) * SIN (RAD) + ZCTR
390 Y(N) = YR:Z(N) = ZR
400 NEXT N
410 RETURN
420 REM Y-AXIS ROTATION
430 FOR N = 1 TO PNTCOUNT
440 XR = (Z(N) - ZCTR) * SIN (RAD) + (
X(N) - XCTR) * COS (RAD) + XCTR
450 ZR = (Z(N) - ZCTR) * COS (RAD) - (
X(N) - XCTR) * SIN (RAD) + ZCTR
460 X(N) = XR:Z(N) = ZR
470 NEXT N
480 RETURN
490 REM Z-AXIS ROTATION
500 FOR N = 1 TO PNTCOUNT
510 XR = (X(N) - XCTR) * COS (RAD) - (
Y(N) - YCTR) * SIN (RAD) + XCTR
520 YR = (X(N) - XCTR) * SIN (RAD) + (
Y(N) - YCTR) * COS (RAD) + YCTR
530 X(N) = XR:Y(N) = YR
540 NEXT N
550 RETURN
560 REM FIND OBJECT'S CENTER
570 XMAX = 0:YMAX = 0:ZMAX = 0:XSML = 2
80:YSML = 160:ZSML = 300
580 FOR N = 1 TO PNTCOUNT
590 IF X(N) < XSML THEN XSML = X(N)
600 IF X(N) > XMAX THEN XMAX = X(N)
610 IF Y(N) < YSML THEN YSML = Y(N)
620 IF Y(N) > YMAX THEN YMAX = Y(N)
630 IF Z(N) < ZSML THEN ZSML = Z(N)
640 IF Z(N) > ZMAX THEN ZMAX = Z(N)
650 NEXT N
660 XCTR = (XSML + XMAX) / 2:YCTR = (YS
ML + YMAX) / 2:ZCTR = (ZSML + ZMAX)
 / 2
670 RETURN
680 REM MAIN DRIVER
690 GOSUB 560: REM FIND CENTER OF OBJ
ECT
700 FOR AXIS = 3 TO 1 STEP - 1
710 IF DEG(AXIS) = 0 OR DEG(AXIS) = 360
 THEN GOTO 740
720 RAD = DEG(AXIS) / 57.2958: REM R
ADIANS TO USE FOR CALCULATIONS
730 ON AXIS GOSUB 350,420,490: REM CA
LCULATE ROTATION FOR APPROPRIATE AX
IS
740 NEXT AXIS
750 GOSUB 250: REM CONVERT 3-D TO 2-D
760 GOSUB 190: REM PLOT OBJECT ON SCR
EEN
770 ON OBJ GOSUB 1190,1320,1480: REM
RE-INITIALIZE ARRAY
780 RETURN
790 REM INPUT
800 VTAB 21: HTAB 1: CALL CLR: REM
CLEAR BOTTOM FOUR TEXT LINES
810 VTAB 21: HTAB 1: PRINT "ENTER DEGRE
ES TO ROTATE AXIS (0-360):"
820 VTAB 24: HTAB 18: PRINT "<RETURN> A
FTER ENTRY";
830 VTAB 22: HTAB 18: PRINT "FOR MENU E
NTER '999'";
840 POKE - 16368,0: REM CLEAR EXTRA
KEYPRESS, IF ANY
850 VTAB 22: HTAB 3: INPUT "X-AXIS: ";D
EG(1)
860 IF DEG(1) = 999 THEN RETURN
870 HTAB 3: INPUT "Y-AXIS: ";DEG(2)
880 HTAB 3: INPUT "Z-AXIS: ";DEG(3)
890 VTAB 21: HTAB 1: CALL CLR
900 VTAB 21: HTAB 1: PRINT "ANGLES OF A
XIS ROTATION:"
910 HTAB 3: PRINT "X-AXIS: ";DEG(1)
```

APPLE II Series

```
920 HTAB 3: PRINT "Y-AXIS: ";DEG(2)
930 HTAB 3: PRINT "Z-AXIS: ";DEG(3);
940 GOSUB 680
950 GOTO 790: REM ANOTHER ROTATION SE
RIES
960 REM OBJECT MENU
970 TEXT : HOME : HTAB 11: PRINT "3-D O
BJECT DISPLAY"
980 HTAB 11: FOR N = 1 TO 18: PRINT "-"
;: NEXT N
990 VTAB 6: HTAB 1: PRINT "OPTIONS:"
1000 VTAB 9: HTAB 5: PRINT "1) CUBE"
1010 VTAB 11: HTAB 5: PRINT "2) PYRAMID"
1020 VTAB 13: HTAB 5: PRINT "3) HOUSE"
1030 VTAB 15: HTAB 5: PRINT "4) LOAD OBJ
ECT FROM DISK": PRINT
1040 VTAB 17: HTAB 5: PRINT "5) QUIT": P
RINT : PRINT
1050 VTAB 20: HTAB 1: PRINT "CHOOSE 1-5:
";
1060 GET K$:OBJ = VAL (K$): IF OBJ < 1
OR OBJ > 5 THEN 1060
1070 PRINT K$;
1080 ON OBJ GOSUB 1190,1320,1480,1640,20
00
1090 GOSUB 250: GOSUB 190: REM DRAW INI
TIAL IMAGE OF OBJECT
1100 GOSUB 790: REM INPUT
1110 GOTO 960
1120 REM INITIALIZATION
1130 D$ = CHR$ (4):SIZE = 100:VZ = - 1
5000:CLR = - 958
1140 DIM X(SIZE),Y(SIZE),Z(SIZE)
1150 DIM XP(SIZE),YP(SIZE)
1160 DIM PC(SIZE * 3)
1170 DIM DEG(3)
1180 GOTO 960
1190 REM CUBE DATABASE
1200 PNTCOUNT = 10:LINECOUNT = 13
1210 FOR N = 1 TO PNTCOUNT
1220 READ X(N),Y(N),Z(N)
1230 Y(N) = ABS (Y(N) - 160): REM INVE
RT Y-AXIS
1240 NEXT N
1250 DATA 100,120,0,180,120,0,180,40,0,1
00,40,0,100,40,80,100,120,80,180,12
0,80,180,40,80,140,40,0,140,50,0
1260 FOR N = 1 TO LINECOUNT * 2
1270 READ PC(N)
1280 NEXT N
1290 DATA 1,2,2,3,3,4,4,1,5,6,6,7,7,8,8,
5,1,6,2,7,3,8,4,5,9,10
1300 RESTORE
1310 RETURN
1320 REM PYRAMID DATABASE
1330 FOR ADVANCE = 1 TO 56
1340 READ DP: REM ADVANCES DATA LIST
POINTER TO PYRAMID
1350 NEXT ADVANCE
1360 PNTCOUNT = 7:LINECOUNT = 9
1370 FOR N = 1 TO PNTCOUNT
1380 READ X(N),Y(N),Z(N)
1390 Y(N) = ABS (Y(N) - 160): REM INVE
RT Y-AXIS
1400 NEXT N
1410 DATA 90,40,0,190,40,0,190,40,100,90
,40,100,140,120,50,140,40,0,140,50,
5
1420 FOR N = 1 TO LINECOUNT * 2
1430 READ PC(N)
1440 NEXT N
1450 DATA 1,2,2,3,3,4,4,1,1,5,5,4,2,5,5,
3,6,7
1460 RESTORE
1470 RETURN
1480 REM HOUSE DATABASE
1490 FOR ADVANCE = 1 TO 95
1500 READ DP: REM ADVANCES DATA LIS
T POINTER TO HOUSE
1510 NEXT ADVANCE
1520 PNTCOUNT = 18:LINECOUNT = 22
1530 FOR N = 1 TO PNTCOUNT
1540 READ X(N),Y(N),Z(N)
1550 Y(N) = ABS (Y(N) - 160): REM INVE
RT Y-AXIS
1560 NEXT N
1570 DATA 100,40,0,180,40,0,180,40,100,1
00,40,100,100,110,0,100,110,100,140
,140,100,180,110,100,180,110,0,140,
140,0,180,40,20,180,80,20,180,80,40
,180,40,40,120,60,0,160,60,0,160,80
,0,120,80,0
1580 FOR N = 1 TO LINECOUNT * 2
1590 READ PC(N)
1600 NEXT N
1610 DATA 1,2,2,3,3,4,4,1,1,5,5,6,6,7,7,
8,8,9,9,10,10,5,10,7,3,8,4,6,2,9,11
,12,12,13,13,14,15,16,16,17,17,18,1
8,15
1620 RESTORE
1630 RETURN
1640 REM ROUTINES TO BE ADDED
1650 REM
2000 REM EXIT PROGRAM
2010 TEXT : HOME
2020 END
```

HCM

SAVING . . . from p. 119

IBM PCjr

```
470 Y=FY-PY:M=FM-PM:IF M<=0 THEN M=M+12:Y=Y-1
480 TY=Y+M/12
490 IF FY*100+FM<=PY*100+PM OR TY<=0 THEN LOCATE 9,1:PRINT "THE FUTURE DATE IS SMALLER THAN THE PRESENT DATE.  PLEASE RE-ENTER THE DATES.";CHR$(7);CHR$(7):PRINT "PRESS ENTER TO CONTINUE" ELSE 520
500 A$=INKEY$:IF A$="" THEN 500
510 IF A$<>CHR$(13) THEN 500 ELSE 430
520 LOCATE 9,1:INPUT "PERCENT INTEREST:";IN:IF IN<0 THEN 520
530 R=IN/100
540 LOCATE 11,1:INPUT "COMPOUNDS PER YEAR:";K:IF K<1 THEN 540
550 ON CHOICE GOSUB 620,680,770,910
560 LOCATE 24,1:PRINT "PRESS ENTER TO RETURN TO THE MENU SCREEN";
570 A$=INKEY$:IF A$="" THEN 570
580 GOTO 370
590 REM ****************************
600 REM COMPOUND INTEREST SUBROUTINE
610 REM ****************************
620 GOSUB 960
630 LOCATE 15,1:PRINT "FUTURE SAVINGS=$";FNRD(FNC):LOCATE 17,1:PRINT "TOTAL INTEREST= $";FNRD(FNC-PRE(X))
640 RETURN
650 REM *************************
660 REM LEVEL PAYMENTS SUBROUTINE
670 REM *************************
680 GOSUB 980:GOSUB 1000
690 I=R/K:TY=Y+M/12:TP=INT(NP*TY):IF TP=0 THEN TP=1
700 N=K/NP*TP:F=FNFV:N=K/NP:P=FNPV:TY=(M/(12/NP)-INT(M/(12/NP)))/NP
710 IF TY<>0 THEN ND=ND/(1+IN/(100*K))^(K*TY)
720 LOCATE 17,1:PRINT "LEVEL PAYMENTS=$";FNRD(FNCP)
730 RETURN
740 REM ******************************
750 REM INCREASING PAYMENTS SUBROUTINE
760 REM ******************************
770 GOSUB 980:GOSUB 1000:GOSUB 1020
780 I=R/K:TY=Y+M/12:IF Y<1 OR TY>25 THEN LOCATE 19,1:PRINT "TIME PERIOD MUST BE FROM 1 TO 25 YEARS.  PLEASE RE-ENTER THE DATES.";CHR$(7);CHR$(7);CHR$(7);CHR$(7):PRINT "PRESS ENTER TO CONTINUE" ELSE 810
790 A$=INKEY$:IF A$="" THEN 790
800 IF A$<>CHR$(13) THEN 790 ELSE 430
810 X=0:A=0:CA(1)=1:N=K/NP:P=FNPV:N=K:F=FNFV:Z=1:PRE(1)=FNG
820 FOR Z=2 TO Y+1:CA(Z)=CA(Z-1)*(1+RA/100):PRE(Z)=FNG:NEXT Z
830 IF Y<>0 THEN FOR X=1 TO Y:TY=TY-1:IF TY=0 THEN A=A+PRE(X) ELSE A=A+FNC:NEXT X
840 IF M>=12/NP THEN Z=X:N=K/NP:P=FNPV:N=K/NP*INT(M/(12/NP)):F=FNFV:A=A+FNG:TY=(M/(12/NP)-INT(M/(12/NP)))/NP:IF TY<>0 THEN PRE(X)=CA(X):A=A+FNC
850 PA=ND/A
860 LOCATE 19,1:PRINT "1st PERIOD PAYMENT = $";FNRD(PA)
870 RETURN
880 REM ******************************
890 REM FUTURE VALUE FIXED PAYMENTS SUBROUTINE
900 REM ******************************
910 GOSUB 980:GOSUB 1040
920 I=R/K:Z=1:TY=Y+M/12:TP=INT(NP*TY):IF TP=0 THEN TP=1
930 N=K/NP:P=FNPV:N=K/NP*TP:F=FNFV:A=FNG:TY=(M/(12/NP)-INT(M/(12/NP)))/NP:IF TY<>0 THEN PRE(X)=A:A=FNC
940 LOCATE 17,1:PRINT "SAVINGS WILL = $";FNRD(A)
950 RETURN
960 LOCATE 13,1:INPUT "PRESENT AMOUNT IN SAVINGS:";PRE(X):IF PRE(X)<0 THEN GOTO 960
970 RETURN
980 LOCATE 13,1:INPUT "PAYMENTS PER YEAR:";NP:IF NP<1 OR NP>365 THEN GOTO 980
990 RETURN
1000 LOCATE 15,1:INPUT "AMOUNT NEEDED AT END OF PERIOD:";ND:IF ND<0 THEN GOTO 1000
1010 RETURN
1020 LOCATE 17,1:INPUT "% INCREASE PER YEAR IN PAYMENTS:";RA:IF RA<0 THEN GOTO 1020
1030 RETURN
1040 LOCATE 15,1:INPUT "AMOUNT OF PAYMENTS TO SAVINGS:";CA(1):IF CA(1)<0 THEN GOTO 1040
1050 RETURN
1060 REM *****************
1070 REM END OF THE PROGRAM
1080 REM *****************
1090 KEY ON:CLS:END
```

HCM

SAVING . . . from p. 117

SAVING (C-64)
Explanation of the Program

Line Nos.	
100-160	Program header.
170-210	Display title screen.
220-290	User-defined functions.
300-410	Display menu screen; get input of choice.
420-610	Get input of DATA common to all routines; branch to appropriate subroutine.
620-650	Upon return from subroutines, wait for RETURN to be pressed before continuing.
1000-1040	Subroutine for Compound Interest.
2000-2080	Subroutine for Increasing payments.
3000-3170	Subroutine for Increasing Payments.
4000-4070	Subroutine for Future Value.
10000-10030	Subroutine to get input of Present Amount.
20000-20030	Subroutine to get input of Payments/Year.
30000-30030	Subroutine to get input of Amount Needed.
40000-40030	Subroutine to get input of % Increase/Year.
50000-50030	Subroutine to get input for Payment Amounts.

COMMODORE 64

```
100 REM ************
110 REM *   SAVING   *
120 REM ************
130 REM BY JOEL S. MOSKOWITZ AND THE HCM STAFF
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.1.1
160 REM C64 BASIC
170 REM FIRST DISPLAY
180 PRINT "{SHIFT CLR}{8CRSRDOWN}          SAVING"
190 PRINT "{9CRSRDOWN}                    PRESS RETURN TO BEGIN"
200 GET A$:IF A$="" THEN 200
210 IF A$<>CHR$(13) THEN 200
220 REM INITIALIZE
230 DIM CA(25),PR(25): PRINT "{SHIFT CLR}"
240 DEF FNPV(N)=(1-1/(1+I)↑N)/I
250 DEF FNFV(N)=((1+I)↑N-1)/I
260 DEF FNG(F)=F*CA(Z)/P
270 DEF FNCP(P)=P*ND/F
280 DEF FNC(IN)=PR(X)*(1+IN/(100*K))↑(K*TY)
290 DEF FNRD(A)=INT(A*100+.5)/100
300 REM MENU SCREEN
310 PRINT "{SHIFT CLR}"
320 PRINT "{2CRSRDOWN} YOUR CHOICE:"
330 PRINT "{CRSRDOWN}  1. COMPOUND INTEREST"
340 PRINT "{CRSRDOWN}  2. LEVEL PAYMENTS"
350 PRINT "{CRSRDOWN}  3. INCREASING PAYMENTS"
360 PRINT "{CRSRDOWN}  4. FUTURE VALUE,{CRSRDOWN}{12SHIFT CRSRLEFT}FIXED PAYMENTS"
370 PRINT "{CRSRDOWN}  5. END PROGRAM"
380 GET CH$:IF CH$="" THEN 380
390 IF ASC(CH$)<49 OR ASC(CH$)>53 THEN 380
400 CH=ASC(CH$)-48
410 IF CH=5 THEN PRINT"{SHIFT CLR}":END
420 PRINT "{SHIFT CLR}"
430 INPUT "  1. PRESENT{CRSRRIGHT}MONTH:  ";PM
440 IF PM<1 OR PM>12 THEN PRINT "{2SHIFT CRSRUP}":GOTO 430
450 INPUT "            YEAR";PY
460 IF PY<1 THEN PRINT "{2SHIFT CRSRUP}":GOTO 450
470 INPUT "  2. FUTURE MONTH:  ";FM
480 IF FM<1 OR FM>12 THEN PRINT "{2SHIFT CRSRUP}":GOTO 470
490 INPUT "            YEAR:  ";FY
500 IF FY<PY THEN PRINT "{2SHIFT CRSRUP}":GOTO 490
510 IF TY<=0THEN PRINT "{SHIFT CRSRUP}"
520 Y=FY-PY:M=FM-PM
530 IF M<0 THEN M=M+12:Y=Y-1
540 TY=Y+M/12
550 IF TY<=0 THEN PRINT "TIME MUST BE GREATER THAN ZERO{HOME}":GOTO 430
560 INPUT "  3. PCT INTEREST:  ";IN
570 IF IN<=0 THEN PRINT "{2SHIFT CRSRUP}":GOTO 560
580 R=IN/100
```

COMMODORE 64

```
590 INPUT "   4. COMPOUNDS/YEAR:  ";K
600 IF K<=0 THEN PRINT "{2SHIFT CRSRUP}":GOTO 590
610 ON CH GOSUB 1000,2000,3000,4000
620 PRINT "{CRSRDOWN}   PRESS RETURN TO CONTINUE"
630 GET A$:IF A$="" THEN 630
640 IF A$<>CHR$(13) THEN 630
650 CLR :GOTO 230
1000 REM COMPOUND INTEREST
1010 GOSUB 10000
1020 PRINT "{CRSRDOWN}   FUTURE SAVINGS= $";STR$(FNRD(FNC(IN)))
1030 PRINT "{CRSRDOWN}   TOTAL INTEREST= $";STR$(FNRD(FNC(IN)-PR(X)))
1040 RETURN
2000 REM LEVEL PAYMENTS
2010 GOSUB 20000
2020 GOSUB 30000
2030 I=R/K:TP=INT(NP*TY):IF TP=0 THEN TP=1
2040 N=K/NP*TP:F=FNFV(N):N=K/NP:P=FNPV(N)
2050 TY=(M/(12/NP)-INT(M/(12/NP)))/NP
2060 IF TY=0 THEN 2070:ND=ND/(1+IN/(100*K))↑(K*TY)
2070 PRINT "{CRSRDOWN}   LEVEL PAYMENTS= $";STR$(FNRD(FNCP(P)))
2080 RETURN
3000 REM INCREASING PAYMENTS
3010 GOSUB 20000
3020 GOSUB 30000
3030 GOSUB 40000
3040 IF TY>=1 AND TY<=25 THEN 3060
3050 PRINT " TIME PERIOD MUST BE FROM 1 TO 25 YEARS USE ROUTINE 2":RETURN
3060 I=R/K:X=0:A=0:CA(1)=1:N=K/NP:P=FNPV(N):N=K:F=FNFV(N):Z=1:PR(1)=FNG(F)
3070 FOR Z=2 TO Y+1:CA(Z)=CA(Z-1)*(1+RA/100):PR(Z)=FNG(F):NEXT Z
3080 IF Y=0 THEN 3130
3090 FOR X=1 TO Y:TY=TY-1
3100 IF TY=0 THEN A=A+PR(X):GOTO 3120
3110 A=A+FNC(IN)
3120 NEXT X
3130 IF M<12/NP THEN 3150
3140 Z=X:N=K/NP:P=FNPV(N):N=K/NP*INT(M/(12/NP)):F=FNFV(N):A=A+FNG(F)
3150 TY=(M/(12/NP)-INT(M/(12/NP)))/NP:IF TY=0 THEN 3160:PR(X)=CA(X):A=A+FNC(IN)
3160 PA=ND/A
3170 PRINT "{CRSRDOWN}   1ST PERIOD PAYMENT=   $";STR$(FNRD(PA)):RETURN
4000 REM FUTURE VALUE
4010 GOSUB 20000
4020 GOSUB 50000
4030 I=R/K:Z=1:TP=INT(NP*TY):IF TP=0 THEN TP=1
4040 N=K/NP:P=FNPV(N):N=K/NP*TP:F=FNFV(N):A=FNG(F)
4050 TY=(M/(12/NP)-INT(M/(12/NP)))/NP:IF TY=0 THEN 4060:PR(X)=A:A=FNC(IN)
4060 PRINT "{CRSRDOWN}   SAVINGS WILL=   $";FNRD(A)
4070 RETURN
10000 REM SUB FOR PRESENT AMOUNT
10010 INPUT "5. PRESENT AMOUNT:  ";PR(X)
10020 IF PR(X)<=0 THEN PRINT "{2SHIFT CRSRUP}":GOTO 10010
```

COMMODORE 64

```
10030 RETURN
20010 PRINT "   5. PAYMENTS/YEAR:  ":INPUT NP
20020 IF NP<=0 THEN PRINT "{3SHIFT CRSRUP}":GOTO 20010
20030 RETURN
30010 PRINT "   6. AMOUNT NEEDED:  ":INPUT ND
30020 IF ND<=0 THEN PRINT "{3SHIFT CRSRUP}": GOTO 30010
30030 RETURN
40010 PRINT "   7. % INCREASE/YEAR:  ":INPUT RA
40020 IF RA<0 THEN PRINT "{3SHIFT CRSRUP}":GOTO 40010
40030 RETURN
50010 PRINT "   6. PAYMENT AMOUNTS:  ":INPUT CA(1)
50020 IF CA<0 THEN PRINT "{3SHIFT CRSRUP}":GOTO 50010
50030 RETURN
```

HCM

SAVING . . . from p. 117

SAVING (VIC-20)
Explanation of the Program

Line Nos.	
100-160	Program header.
170-200	Display title screen.
210-270	User-defined functions.
280-380	Display menu screen; get input of choice.
390-580	Get input of DATA common to all routines; branch to appropriate subroutine.
590-620	Upon return from subroutines, wait for RETURN key to be pressed before continuing.
1000-1040	Subroutine for Compound Interest.
2000-2080	Subroutine for Increasing payments.
3000-3170	Subroutine for Increasing Payments.
4000-4070	Subroutine for Future Value.
10000-10030	Subroutine to get input of Present Amount.
20000-20030	Subroutine to get input of Payments/Year.
30000-30030	Subroutine to get input of Amount Needed.
40000-40030	Subroutine to get input of % Increase/Year.
50000-50030	Subroutine to get input for Payment Amounts.

VIC-20

```
100 REM ************
110 REM *   SAVING   *
120 REM ************
130 REM BY JOEL S. MOSKOWITZ AND THE HCM STAFF
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.1.1
160 REM V20 BASIC
170 PRINT "{SHIFT CLR}{8CRSRDOWN}   SAVING"
180 PRINT "{9CRSRDOWN}PRESS RETURN TO BEGIN"
```

Continued on p. 149

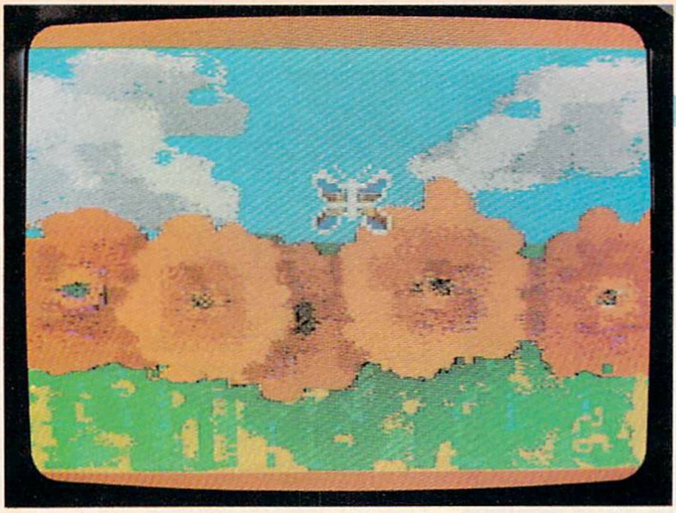


Photo 1: **A screen found on the *StarSprite I Demo* disk. The background was created using Sprite Painting, and the butterfly is an animated sprite. Appropriate insect sounds complete the demonstration.**

The Future is Now in Apple Graphics

The User's Perspective

by Evelyn Lee

Even though you love your Apple, you may have been sneaking peeks at other home computer systems that offer better graphics. If you have been watching the newspaper for holiday computer bargains or listening wistfully to those TV ads for "new, improved home computers," you will be happy to learn that you can achieve outstanding graphics not by replacing your Apple, but by updating it.

In The Past

We have known for years that Apple graphics were clumsy and slow and less than satisfying, compared to other systems. Historically, Apple's priorities did not include built-in, high-quality, or user-friendly graphics. The designers of the Apple did not foresee how important quality graphics would be to home computer users. Instead, the company decided to concentrate on providing an ever improving, but stable and unspecialized product, and to let peripherals manufacturers cover the graphics area.

For these manufacturers, the challenge was to provide smooth, colorful, fast animation which lacked flicker and was easy to use—all while maintaining compatibility with existing Apple graphics features. In other words, we needed a great new system that worked easily and did not throw out the Apple with the bath water.

It looks like the challenge has finally been met by two new products: the *Supersprite* board and *StarSprite* software.

The Present

The hardware and software come from separate companies. Synetix Systems, of Redmond, Washington, and Avant-Garde Creations, of Eugene, Oregon, have collaborated to introduce an impressive hardware/software package.

The board by Synetix provides features that let you combine existing Apple graphics and sounds with new

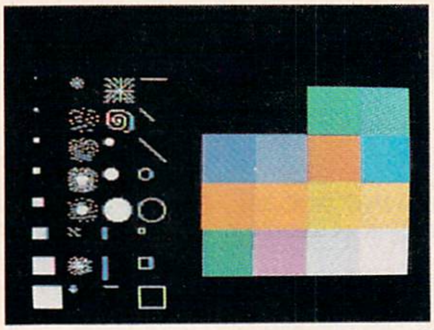


Photo 2: The Sprite Painting color and paintbrush palette. You may choose from any of these 16 colors (the first two are transparent and black), and you can use any of these paintbrushes or design any style of your own.

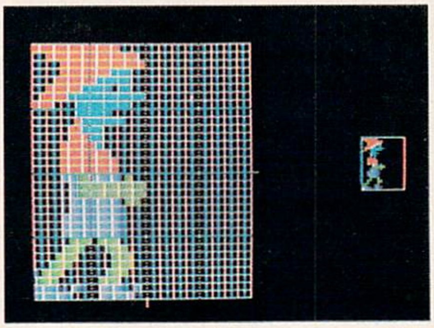


Photo 3: A sprite enlarged in an editing grid and the actual-size sprite beside it. Grids such as these are used to create and edit sprites.

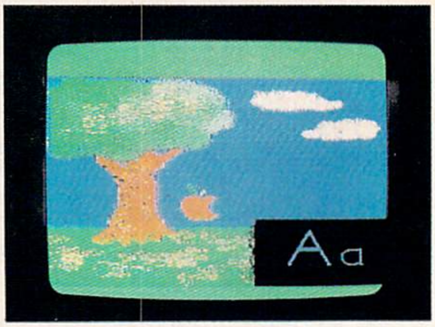


Photo 4: A screen that is part of *AlphaSprite*, an educational game created for and with the *SuperSprite/StarSprite* system.

An Interview with Don Fudge

Don Fudge, programmer and co-founder of Avant-Garde Creations, is the author of several books and utilities dealing with Apple graphics, including Hi-Res Secrets, *the* Graphics Applications System, *and the* Paint Master Scene Utility. *In collaboration with Synetix Systems, a hardware manufacturer, Avant-Garde is producing a revolutionary Apple graphics system, which features Ampersprite, an enhanced Apple BASIC language developed by Fudge. The hardware/software package is accompanied by a three-volume software system, utilities, sample games, tutorials, and instruction manuals.*

We interviewed Don at his multilevel home overlooking Eugene, Oregon and the southern Willamette Valley. He agreed to fill us in on his perspective on Apple graphics, past and present.

HCM: Don, you've been frequently quoted as saying that Apple graphics have always had a number of serious drawbacks. Would you review, from the programmer's standpoint, what these problems have been?

DF: For years, Apple has lived with a reputation for the most complex and confusing graphics around. There was only one video plane. There was the color-clash. There were preshifted shapes, jumbled hi-res mapping, and only 7 bits per byte displayed. There were the background erasing, the mumbley sounds, and the minuscule opportunities for text with graphics. The problems were obvious, and the ramifications were serious in terms of educational software limitations, non-competitive commercial games, and headaches for the relatively few programmers who tackled machine language graphics. It took a long time to overcome all these problems because they were so complicated, and because Apple decid-

The Animated Avant-Garde Logo

This logo is an example of simultaneous and synchronized use of Apple 6502 graphics and VDP (*SuperSprite*) graphics. The background logo is 6502 graphics. The walking figure is a VDP sprite.

The sprite is seen at magnification 1. At magnification 0 it would have twice the resolution and half the size. The animation was created in Applesoft using Ampersprite.

The combination of VDP and 6502 is mixed on the screen by using a simple GOSUB 23009, which contains only two I/O POKEs. All screen switches require a mere two POKEs, regardless of whether you want pure VDP, pure 6502, or mixed.

```
2309 POKE 49396:
     POKE 49398: RETURN
```

graphics and sound features, including natural speech. The software by Avant-Garde is friendly, requires little or no programming experience, and adapts well to creative applications.

The system includes the *SuperSprite* board and an instruction manual which provides technical information on the board and Echo speech. In addition, the package contains the *StarSprite I* software system, which comes with two disks of demonstrations and tutorials on installation and adjustment, the *Ampersprite* language which interfaces with the board, utility programs, and an accompanying detailed instruction manual. The price tag on the entire package is $395.

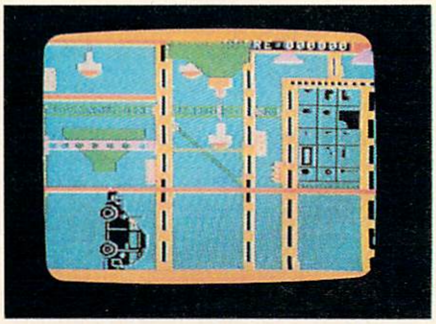


Photo 5: A screen from *Assembly Line Madness* is shown in this photo. This is a new hand/eye coordination style arcade game now on the market for *SuperSprite* owners.

Synetix Systems is also beginning to market different versions of the *SuperSprite* board with various combinations of features. There is a version, called *Sprite I*, which includes the basic graphics capabilities without sound or natural speech. This board can be used in any vacant slot in your Apple, comes with software, and sells for $149. *Sprite II* adds sound effects to the basic graphics capabilities and sells for $249.

The *SuperSprite* board is compatible with any 48K Apple II, Apple II+, or Apple IIe. (Unfortunately, it will not work with Apple IIe's revision A motherboard, but that board can be updated or hot-wired. See your Apple dealer or call Avant-Garde.) When you get a chance to see a demonstration at your computer store, you will notice several exciting changes on the screen.

First the colors. There are 16 of them, and they are bright and crisp, as you can see in the screen photos. At one point in the demo, you will see an animated picture of a butterfly gliding and fluttering (not flickering) through a pastoral scene. The scene has the quality of a painting, not a stick drawing with added color. Several of the 16 colors are mixed to create shading, new colors, and depth. And the butterfly makes a little lepidoptera sound that in no way slows down the action. This feature lets even novice programmers add sophistication and quality to their graphics.

Next, you will probably notice the arcade-style games full of dazzling fast action. Three types of games come with the system, representing the three most popular arcade games now on the market. The *StarSprite* software also includes utilities that allow even non-programmers to add to or modify the games. Programmers will want these utilities to create entirely new games.

This demo ought to convince you that visual quality on the Apple has finally reached the level of other systems.

Go Home With It

Synetix furnishes instructions for installation of the board, and Avant-

Continued on p. 167

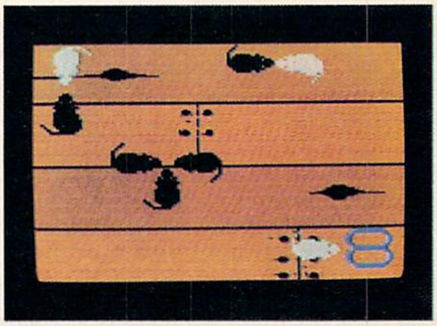


Photo 6: A screen from *NumberSprite*, an educational program for early learners. Adapted for use with the *SuperSprite* board, this program teaches the alphabet using quick, colorful sprite animation.

ed to let the peripherals manufacturers tackle the problems on their own.

HCM: Why is all of that different now?

DF: Apple graphics have become state-of-the-art thanks to a two-part invention. The *SuperSprite* board (by Synetix Systems, Inc.) is a peripheral board [that plugs into the Apple] combining a Video Display Processor (VDP) with a Progammable Sound Generator (PSG) and speech synthesizer. The exciting part of the hardware is that all the features on the board are compatible with concurrent Apple 6502 graphics and sounds.

The *StarSprite* software system that we are publishing at Avant-Garde provides the interface for Applesoft programming and utilities. We made sure the instructions are friendly, and there are a lot of explanations for programmers who want to get right in and adapt the system to their own styles. The programs are unlocked so people can be their most creative with them. But any novice can use them just as easily.

HCM: What precisely are the features of the board?

DF: Synetix Systems designed the *SuperSprite* board to combine existing Apple graphics and sounds with sprites, 32 independent animation planes, a background plane, a backdrop color, non-interfering sound effects and tones, plus natural speech. The specifications of the board sound like some whiz kid's Christmas list: 16K of video RAM, the Texas Instruments Video Display Processor (TMS 9918A), General Instruments Programmable Sound Generator (AY-3-8912), the Echo Speech Synthesizer, a 0.4 watt audio amplifier, a speaker, volume control, programmable audio envelopes, sound filters, and the capacity for 3 musical tones, sound effects and simultaneous graphics.

The board also provides the ability to mix and adapt Apple graphics with it's own VDP graphics. It works on any 48K Apple II, Apple II+ or Apple IIe.[See accompanying product overview for a slight exception—Ed.]

Continued on p. 148

SuperSprite enables multilevel video planes and *sprites*, which are the quick, crisp, programmable objects you animate independently of the background and each other. It offers a text mode with 40 characters per line and 24 vertical text lines. The 16 colors are mixable. Sound effects and three channels for tones can be accessed from two different I/O addresses. They have no effect on graphics whatsoever. You 6502 programmers will realize what that means. You will no longer get that disappointing message, SORRY, GRAPHICS, THE MICROPROCESSOR IS BUSY WITH SOUND—YOU HAVE TO WAIT.

HCM: What are the multilevel video planes?

DF: The board produces 34 video planes on top of the Apple 6502 screen. The first 32 planes are all for sprites—one sprite each. These planes are arranged in a priority system so a sprite on plane zero will cover a sprite directly below it on a lower plane, thus allowing a three-dimensional effect. The 33rd plane is the pattern plane, designed to display scenes such as backgrounds for adventure games, pictures,and stuff like that. The 34th plane is the VDP backdrop plane, which can be any of 16 solid colors. And just to keep things interesting, the Apple's own 6502 graphics screen is fully functional, and available for including graphics or text display.

HCM: Of what significance are sprites to the new system?

DF: Sprites can be created and animated so quickly and easily it will make your socks fall off. The video planes where these little sprites cavort enable them to roam the entire screen without interfering with each other or the background. There are none of the animation programming headaches of the past such as color-clash or background erasing. Not only that, but programming for these fun "creatures" (as for most graphics, text, or sound applications) can almost all be done in good old familiar Applesoft BASIC. That is because sprites are so fast, that using them does not require machine language, except in the fastest action arcade game applications.

HCM: Is this ease of programming possible because of the support software that you developed?

DF: Avant-Garde Creations is publishing *StarSprite* software to accompany the *SuperSprite* board. *StarSprite* is a three-part software system that provides demonstrations, tutorials, utilities, and Ampersprite, the Apple BASIC enhancement language that makes programming for the board so easy that it's really no different from Applesoft. In fact, you use them together.

StarSprite I is included with the *SuperSprite* board. It includes an extensive self-running demonstration; tutorials for using the Ampersprite language in graphics, sound, and educational applications; three sample games which can be modified by even non-programmers; and a variety of utilities that are unlocked and easy to customize. One of the most exciting utilities in *StarSprite I* lets you create scenes as you do with oils or watercolors, using no line drawing. You use sprites as the paintbrush.

> **"The speed and smoothness of animation possible with Ampersprite will preclude the need for most assembly applications."**

HCM: What about the other two parts of the software?

DF: *StarSprites II* and *III* are sold separately and provide more utilities and information for users interested in programming their own graphics. Line drawing, color-fill, instant scenes, scrolling, pattern editing—it's all there. One of the utilities in *StarSprite II* gives you the capacity to transpose existing Apple graphics into VDP graphics. A 6502 picture can be transposed into a VDP screen for the pattern planes at the touch of a button. Any type of shape (HPLOT, vector, block, 6502 picture, VDP picture, text character) can be made into

SAVING . . . from p. 145

VIC-20

```
190 GET A$:IF A$="" THEN 190
200 IF A$<>CHR$(13) THEN 190
210 DIM CA(25),PR(25): PRINT "{SHIFT CLR}"
220 DEF FNPV(N)=(1-1/(1+I)↑N)/I
230 DEF FNFV(N)=((1+I)↑N-1)/I
240 DEF FNG(F)=F*CA(Z)/P
250 DEF FNCP(P)=P*ND/F
260 DEF FNC(IN)=PR(X)*(1+IN/(100*K))↑(K*TY)
270 DEF FNRD(A)=INT(A*100+.5)/100
280 PRINT "{SHIFT CLR}"
290 PRINT "{2CRSRDOWN} YOUR CHOICE:"
300 PRINT "{CRSRDOWN} 1. COMPOUND INTEREST"
310 PRINT "{CRSRDOWN} 2. LEVEL PAYMENTS"
320 PRINT "{CRSRDOWN} 3. INCREASING{CRSRDOWN}{11SHIFT CRSRLEFT} PAYMENTS"
330 PRINT "{CRSRDOWN} 4. FUTURE VALUE,{CRSRDOWN}{12SHIFT CRSRLEFT}FIXED PAYMENTS"
340 PRINT "{CRSRDOWN} 5. END PROGRAM"
350 GET CH$:IF CH$="" THEN 350
360 IF ASC(CH$)<49 OR ASC(CH$)>53 THEN 350
370 CH=ASC(CH$)-48
380 IF CH=5 THEN PRINT"{SHIFT CLR}":END
390 PRINT "{SHIFT CLR}"
400 PRINT " 1. PRESENT{CRSRRIGHT}MONTH: ":INPUT PM
410 IF PM<1 OR PM>12 THEN PRINT "{3SHIFT CRSRUP}":GOTO 400
420 PRINT " YEAR":INPUT PY
430 IF PY<1 THEN PRINT "{3SHIFT CRSRUP}":GOTO 420
440 PRINT " 2. FUTURE MONTH: ":INPUT FM
450 IF FM<1 OR FM>12 THEN PRINT "{3SHIFT CRSRUP}":GOTO 440
460 PRINT " YEAR: ":INPUT FY
470 IF FY<PY THEN PRINT "{3SHIFT CRSRUP}":GOTO 460
480 IF TY<=0THEN PRINT "{2SHIFT CRSRUP}"
490 Y=FY-PY:M=FM-PM
500 IF M<0 THEN M=M+12:Y=Y-1
510 TY=Y+M/12
520 IF TY<=0 THEN PRINT "TIME MUST BE GREATER THAN ZERO{HOME}":GOTO 400
530 PRINT " 3. PCT INTEREST: ":INPUT IN
540 IF IN<=0 THEN PRINT "{3SHIFT CRSRUP}":GOTO 530
550 R=IN/100
560 PRINT " 4. COMPOUNDS/YEAR: ":INPUT K
570 IF K<=0 THEN PRINT "{3SHIFT CRSRUP}":GOTO 560
580 ON CH GOSUB 1010,2010,3010,4010
590 PRINT "{CRSRDOWN} PRESS RETURN TO":PRINT " CONTINUE"
600 GET A$:IF A$=""THEN 600
610 IF A$<>CHR$(13) THEN 600
620 CLR :GOTO 210
1010 GOSUB 10000
1020 PRINT "{CRSRDOWN} FUTURE SAVINGS=":PRINT "$";STR$(FNRD(FNC(IN)))
1030 PRINT "{CRSRDOWN} TOTAL INTEREST=":PRINT "$";STR$(FNRD(FNC(IN)-PR(X)))
1040 RETURN
2010 GOSUB 20010
```

VIC-20

```
2020 GOSUB 30010
2030 I=R/K:TP=INT(NP*TY):IF TP=0 THEN TP=1
2040 N=K/NP*TP:F=FNFV(N):N=K/NP:P=FNPV(N)
2050 TY=(M/(12/NP)-INT(M/(12/NP)))/NP
2060 IF TY=0 THEN 2070:ND=ND/(1+IN/(100*K))↑(K*TY)
2070 PRINT "{CRSRDOWN} LEVEL PAYMENTS=":PRINT "$";STR$(FNRD(FNCP(P)))
2080 RETURN
3010 GOSUB 20010
3020 GOSUB 30010
3030 GOSUB 40010
3040 IF TY>=1 AND TY<=25 THEN 3060
3050 PRINT " TIME PERIOD MUST BE FROM 1 TO 25 YEARS USE ROUTINE 2":RETURN
3060 I=R/K:X=0:A=0:CA(1)=1:N=K/NP:P=FNPV(N):N=K:F=FNFV(N):Z=1:PR(1)=FNG(F)
3070 FOR Z=2 TO Y+1:CA(Z)=CA(Z-1)*(1+RA/100):PR(Z)=FNG(F):NEXT Z
3080 IF Y=0 THEN 3130
3090 FOR X=1 TO Y:TY=TY-1
3100 IF TY=0 THEN A=A+PR(X):GOTO 3120
3110 A=A+FNC(IN)
3120 NEXT X
3130 IF M<12/NP THEN 3150
3140 Z=X:N=K/NP:P=FNPV(N):N=K/NP*INT(M/(12/NP)):F=FNFV(N):A=A+FNG(F)
3150 TY=(M/(12/NP)-INT(M/(12/NP)))/NP:IF TY=0 THEN 3160:PR(X)=CA(X):A=A+FNC(IN)
3160 PA=ND/A
3170 PRINT "{CRSRDOWN}1ST PERIOD PAYMENT= ":PRINT "$";STR$(FNRD(PA)):RETURN
4010 GOSUB 20010
4020 GOSUB 50010
4030 I=R/K:Z=1:TP=INT(NP*TY):IF TP=0 THEN TP=1
4040 N=K/NP:P=FNPV(N):N=K/NP*TP:F=FNFV(N):A=FNG(F)
4050 TY=(M/(12/NP)-INT(M/(12/NP)))/NP:IF TY=0 THEN 4060:PR(X)=A:A=FNC(IN)
4060 PRINT "{CRSRDOWN} SAVINGS WILL=":PRINT "$";FNRD(A)
4070 RETURN
10000 REM SUB FOR PRESENT AMOUNT
10010 PRINT "5. PRESENT AMOUNT: ":INPUT PR(X)
10020 IF PR(X)<=0 THEN PRINT "{3SHIFT CRSRUP}":GOTO 10010
10030 RETURN
20010 PRINT " 5. PAYMENTS/YEAR: ":INPUT NP
20020 IF NP<=0 THEN PRINT "{3SHIFT CRSRUP}":GOTO 20010
20030 RETURN
30010 PRINT " 6. AMOUNT NEEDED: ":INPUT ND
30020 IF ND<=0 THEN PRINT "{3SHIFT CRSRUP}": GOTO 30010
30030 RETURN
40010 PRINT " 7. % INCREASE/YEAR: ":INPUT RA
40020 IF RA<0 THEN PRINT "{3SHIFT CRSRUP}":GOTO 40010
40030 RETURN
50010 PRINT " 6. PAYMENT AMOUNTS: ":INPUT CA(1)
50020 IF CA<0 THEN PRINT "{3SHIFT CRSRUP}":GOTO 50010
50030 RETURN
```

HCM

a sprite just as easily. We have also included the *Paint Master Scene Utility*—a comprehensive Apple 6502 drawing and coloring utility—in the *StarSprite II* package. With this system, people getting their first taste of graphics have everything they need. Using the package's scene-creating utilities, they can have sprites up running around in front of 6502 scenes.

StarSprite III is for serious programmers who want to get into the heart of the matter. It offers source code and in-depth assembly code explanations of each machine language routine in *StarSprite I* and *II*, and *Paint Master*.

HCM: How easy is your Ampersprite language to use?

DF: I think programmers will be particularly excited by how easy Ampersprite makes it to take charge of the *SuperSprite/StarSprite* system. The speed and smoothness of animation possible with Ampersprite will preclude the need for most assembly applications. Even some of the most sophisticated, commercially viable programs will no longer need assembly language. There is also, by the way, an interrupt-driven version of Ampersprite for collision checking by status register reading. All the information a programmer needs to begin creating new products is included with the *SuperSprite* board in the *StarSprite I* package.

HCM: Are you working on any applications software for this new Apple graphics/sound system?

DF: At Avant-Garde we are publishing a whole line of educational and home games for the new system. The first two to come will be *NumberSprite* and *AlphaSprite*, educational programs aimed at youngers learning to count and use the alphabet. And *Assembly Line Madness*, a hand/eye coordination arcade style game, will be coming out soon. We also plan to provide user follow-up to answer the questions of programmers wanting to forge new ground with the system.

HCM: How much demand do you expect for the *SuperSprite* board and software?

DF: I think large numbers of Apple users are now ready to update their machines. It seems so much more reasonable to stick a *SuperSprite* board in slot 7 of your friendly and familiar Apple rather than struggle with the old Apple graphics, or go get a whole new system. Now Apple users can have graphics that are unsurpassed by any system, without losing any of their past creations. The *SuperSprite/StarSprite* system lets them have it all.

HCM

If you have ever tried to program in BASIC on the Commodore 64, you know that it has some frustrating limitations. It often seems like every operation requires a number of complicated POKEs and PEEKs. Fortunately, a remedy is at hand. Commodore Business Machines has released two software packages, the *Super Expander 64* and *Simon's Basic 64*, that enhance the ease of programming their popular home computer.

Bigger Better Basic

by W. K. Balthrop
HCM Staff

The Super Expander 64
Commodore Business Machines, Inc.
Software Division
1200 Wilson Drive
West Chester, PA 19380
$50.00

Simon's Basic 64
Commodore Business Machines, Inc.
Software Division
1200 Wilson Drive
West Chester, PA 19380
$60.00

The Super Expander 64

The solid-state *Super Expander 64* cartridge goes a long way towards easing the constraints of BASIC by putting an additional 21 commands and 11 functions at your fingertips. The *Expander* will help you in five principal areas: high resolution graphics, sprites, game ports, function keys, and music.

Graphics
The graphics commands of the *Super Expander 64* increase your flexibility. The DRAW command can evoke a single dot, a line from the last pixel pointed to, or a line from any pixel position to any other pixel position. The LOCATE command tells the *pixel cursor* (an invisible cursor from which many of the graphics commands work) where to point.

But the *Super Expander* can do much more than simply draw lines on the high resolution screen. BOX and CIRCLE let you create those shapes in the same way other, more expensive, computers do. This implementation of BOX and CIRCLE is one of the best we have seen. You specify seven parameters, five of which are optional. If you want to, you can simply supply one corner of the box. The opposite corner will be assumed to be at the current pixel cursor position. An ANGLE option supplied with this command makes it unique. By setting this parameter you can rotate the box to any angle.

The ANGLE parameter also sets the *Super Expander 64*'s CIRCLE command apart from those in other implementations. You might wonder what possible advantage an ANGLE parameter could add to a CIRCLE statement; after all, when you turn a circle, it still looks like a circle. But the ANGLE parameter allows you to assign different values to the x and y radii, so that you can draw ellipses. You can also draw only segments of the circle, say, from the 90 degree position to the 270 degree position, to get a half circle.

The commands GSHAPE and SSHAPE can save an area of the screen in a string variable and then place it at any other area of the screen you want. The CHAR statement will let you print with any of the available colors, at any location on the screen, and in multi-color mode.

Sprites
The *Super Expander 64* contains all the commands you need to create and manipulate sprites, including *auto motion*. The term auto motion means that you can set your sprite to sustain movement in a particular direction and at a certain speed without any further input.

"The solid-state *Super Expander 64* cartridge eases the constraints of BASIC by putting an additional 21 commands and 11 functions at your fingertips."

Two new commands provided by the *Super Expander 64* revolutionize the way sprites are created. No longer will you have to sit down with a piece of scratch paper and pencil to figure out all those binary-to-decimal conversions. The computer does them for you with the SPRDEF command. When this command is issued from either a program or in Immediate mode, the screen goes blank and a large grid appears on the left. With cursor controls you can set any of the squares in the grid to either the background, foreground, or one of the multi-colors. Once the sprite has been defined and you return to BASIC, you can save the shape in a string variable with the SPRSAV command.

Once you have your sprites defined and saved, you are ready to set them up: SPRCOL sets the two multi-color colors of the sprite; and SPRITE sets up the foreground color, the priority, the X and Y expansion, whether or not the sprite is turned on, and the sprite mode (multicolor or high resolution). Now that the sprite is both defined and set up, you can place it on the screen with the MOVSPR command. MOVSPR lets you either move the sprite to a new location with one big jump or set a direction in degrees and a velocity from 0 to 15, and watch as the sprite moves smoothly across the screen. Once you get your sprite moving, you can check for collisions with other sprites and the background using the COLINT command and the RBUMP function. The COLINT command will cause the program to branch to a subroutine that you specify with a line number whenever the event occurs. The program will branch once the line currently being executed is completed because of what's called an *interrupt*. While in the subroutine, you can check for specifics with the RBUMP function.

Game Ports
No game is complete unless you can read the game ports for joysticks, paddles, or even a light pen. *Super Expander 64* gives you three commands to simplify this process. RJOY will return the coordinates of either of the two joysticks with values from 0 to 8 and locate the fire button. The RPOT function returns the value of the game paddles with values from 0 to 255 and senses the fire button. Finally, the RPEN function reads the light pen, which can select information from the screen or issue commands to the computer.

Function Keys
With the *Super Expander 64*, you can at long last put those eight Commodore function keys to excellent use. You can redefine each function key with a list of commands up to 255 characters long. Such commands might include LIST, RUN, SAVE, and LOAD. You can also insert special characters that use ASCII code to simulate the pressing of the [RETURN] key.

Music

One of the most impressive features of the Commodore 64 is the sound effects capability provided by the SID (Sound Interface Device) chip. Again *Super Expander* comes through with a set of commands to simplify the process. Three commands set up the SID chip for you. The FILTER command activates and sets up the filtering within the SID chip. TEMPO adjusts the speed at which the music is played, and TUNE adjusts one of the 10 sound envelopes that are preset to sound like 10 different musical instruments. To play music, you need only PRINT either a [CTRL][F] or CHR$(6). You then PRINT a string of commands. Instead of going to the screen, they will be processed for sound reproductions. To complete the music and return to printing on the screen, you issue another [CTRL][F] or CHR$(6).

Ease of Use

I found *Super Expander 64* easy to learn and use. Its commands solve a long list of problems faced by the BASIC programmer. Because almost all of the commands use optional parameters, you can pass those parameters you wish to use, or change them, in many cases making one command multi-functional. This technique also saves programming time and memory.

The documentation for the *Super Expander 64* consists of a 73-page manual. A Command Summary in front of the manual lists the syntax for each command, and the five chapters which follow describe the commands. The manual is written for the person who wants to sit down and devour it from cover to cover, but I found it difficult to use as a spot reference tool. It is not organized to let you go straight to the right page; instead, you might have to do some reading to figure out what command a particular page is talking about. Because it's hard to memorize the commands the first time around, you'll want to keep the manual by your side for a time. During this orientation period you'll constantly look up the different commands as you need to use them.

Simon's Basic for the C-64

Even more powerful than Commodore Business Machines' *Super Expander 64* is their *Simon's Basic 64*. Developed by David Simon, a sixteen-year-old Englishman who started programming at age thirteen, *Simon's Basic 64* adds 114 new commands to the resident BASIC vocabulary of the Commodore 64. We can't cover all of these new commands in this review, so we'll look at a sampling of the more important ones in each of the eleven areas *Simon's Basic* covers.

Programming Aids

Simon's Basic offers the programmer a toolbox of aids. You can MERGE programs. You can assign the function keys to complex commands that will execute at the touch of a button. You can have the computer automatically generate line numbers for you as you enter a program, say, from a magazine. The RENUMBER command will resequence all of the line numbers for you. (The GOTO and GOSUB statements will not be renumbered.) The FIND command you usually see only in word processors enables you to locate a given string of characters. Programmers will find this helpful in debugging a long program. The TRACE command allows you to follow the program's flow as it sequences through the line numbers.

You can use the DUMP command to list the values contained in all non-array variables. If you wish to restart your computer, you can enter COLD. This will erase the program from memory and take your system back to where it was just after the power was turned on. You can use the DISAPA and SECURE commands to hide selective lines of code for security reasons.

Have you ever accidentally typed NEW and erased several hours of programming work? With the OLD command you can recall those lost programs.

Input and Text Manipulation

Simon's Basic has added a number of useful commands to manipulate strings and communicate via the keyboard.

The INSERT command will let you insert one string into another without having to go through the messy string manipulations of MID$ and such. You can also locate a substring within a longer string with the PLACE command. The DUP command will allow you to duplicate a string any number of times. The word AT added to the print statement will enable you to print at any position on the screen without having to use the arrow keys within quotes.

The FETCH command can be used for input validation. The command ON KEY can specifically test characters being entered from the keyboard. This command will cause a branch to a line number no matter when the key is pressed.

Numeric Aids

Number crunching is what computers do best. *Simon's Basic* provides a number of new commands to help make the C-64 sing that binary music. MOD will return the remainder from the division of two integers. DIV returns the integer result of the division of two floating point numbers, and the FRACT function returns only the fractional part of a number. Several characters are used for numeric conversions. The % and $ symbols are used to represent numbers in both binary and hexadecimal form.

Disk Operation

For those who have disk drives, two new commands are available. DISK replaces several other commands that open the disk, print information to it, and then close the disk. With the DISK command you need only supply the information you want to send to the disk following the DISK command. The DIR command will list all or part of the disk directory. You can supply editing commands that will select only certain titles from the catalog.

Graphics

Simon's Basic offers graphics capabilities that were formerly possible only with complicated machine language programs. High-resolution (HIRES) and multi-color (MULTI) plotting and drawing are now possible with a wide variety of commands. The simplest, PLOT, lets you turn on a single pixel. LINE will draw a line from one point to another. With CIRCLE you can specify both the x and y radii to create ellipses. If you want only part of a circle, you can use the ARC command. ANGL will allow you to draw a line from the circle's center to a point on the circumference. If a block of color is what you want, the BLOCK command is the one you need. The PAINT command fills in any of the shapes with color.

The one drawback to the implementation of the commands just described is their lack of versatility. The PLOT, LINE, and BLOCK commands could have been incorporated into one command as in other BASIC languages. The CIRCLE and ARC commands could have been one command. The fewer commands there are in a language, the fewer the programmer has to remember.

One BASIC feature that I have grown to love on the more expensive machines is available in *Simon's Basic*. This is the ability to draw pictures with a string of commands and expand and rotate the picure. The DRAW command defines the shape you want with a string of numbers. The ROT command sets the rotation value and the scale of the shape.

You can print text on the graphics screen in two ways: CHAR allows you to print just one character on the screen. With TEXT you can print a string of characters anywhere on the graphics screen.

Screen Manipulation

One of the flashier commands in *Simon's Basic* is appropriately named FLASH. This command will cause the screen to flash a desired color.

The FCHR command lets you fill an area of the screen with a desired character. You use the FCOL command to change the color of an area of the screen. FILL will do the same thing as

FCHR and FCOL together. The MOVE command will let you relocate an area of the screen. The INV command will let you inverse all of the characters in an area of the screen. The four commands LEFT, RIGHT, UP, and DOWN will let you scroll an area of the screen in any of the four directions. This set of instructions alone has great potential in games design.

The SCRSV and SCRLD commands allow you to save and recall low-resolution graphics screens on either disk or cassette. The last two screen manipulation commands are useful, but their names are a little confusing. The COPY command copies a high-resolution screen to the printer, and HRDCPY copies a low-resolution screen to the printer.

Sprites

Sprites are graphics shapes that can be moved around the screen with high resolution. Most systems that use sprites call them sprites. At the risk of sounding petty, I wonder why *Simon's Basic* insists on calling them MOBs (for Moveable Object Blocks). But regardless of the terminology, a number of *Simon's* commands are quite helpful for designing and using sprites/MOBs.

The DESIGN command allocates the memory for the patterns. Following the DESIGN command is a line-by-line matrix embedded right in the program of the pattern. A high-resolution MOB would be 24 characters wide and 21 program lines long. A multi-color MOB would be 12 characters wide. A standard character pattern would be 8 characters wide by 8 lines long.

Once the MOB is defined, you can set up its parameters with MOBSET, and MOB. You can give the MOB a speed from 1 to 255, with 1 being the fastest. To move the MOB to another location use RLOCMOB, which lets you move a MOB that is already on the screen. Several commands determine collisions and turn the MOB off.

User-defined characters are easy with the MEM command, which moves the characters stored in ROM into the RAM area of the system. You then use the DESIGN command to create your own characters.

Structured Programming

With the new structured programming commands in *Simon's Basic*, you can almost eliminate GOTOs and GOSUBs from your programs. Consequently, programs become a lot easier to read and understand.

Some of the structured commands in *Simon's Basic* include REPEAT–UNTIL, LOOP–ENDLOOP, and RCOMP. RCOMP will retest the last IF–THEN condition to see if it was true or false.

In the structured program, the first statement of a subroutine must be the PROC statement, which creates the subroutine's name. For example, a PROC statement can label the beginning of the program "START" or the end of the program "DONE." Then, to branch to these points, use the command CALL, followed by the name of the procedure. This command is similar to a GOTO. You can declare local and global variables by using the LOCAL and GLOBAL commands.

Error Trapping

Three commands in Simon's BASIC help programmers handle their own error conditions. ON ERROR will send the program to branch to a specific line number whenever an error occurs. Once the program gets to the error-trapping routine, you can elect to use the computer's own error routine with the OUT command. If you want complete control to return to the C-64 error routines, then use NO ERROR.

Music

Simon's Basic gives you several commands that let you set up the sound chip and play music without the usual POKEs. VOL sets the music volume, and WAVE sets up the wave forms for each of the three voices. ENVELOPE will set up the attack/ decay and sustain/release of the envelope generator. Once you have the voices set up, you can compose with MUSIC, which lets you create music through a series of commands embedded in a string. Once the music is composed, you can play it with PLAY.

Game Ports

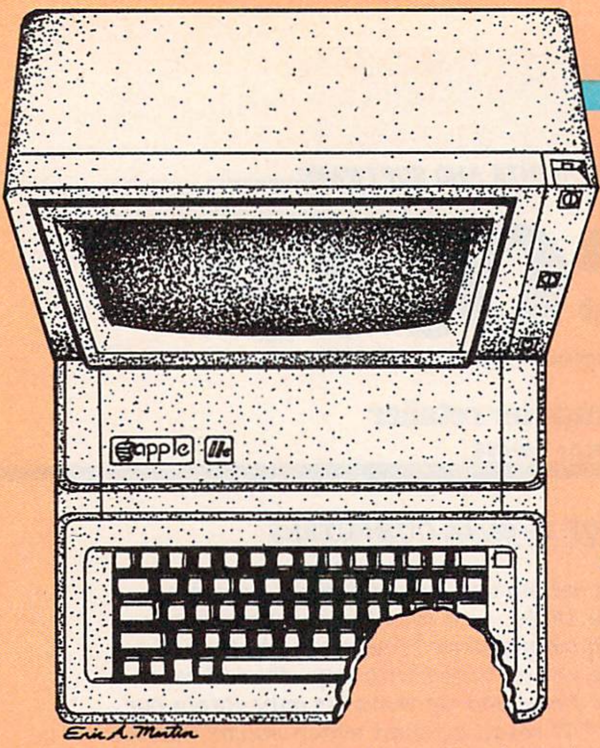
The Commodore 64 has two multi-functional game ports. A joystick, game paddle, or light pen can be attached. There are four commands that get information from the two ports. PENX and PENY read the port for the position of the light pen. POT reads the game paddle potentiometer (a resistor that determines the position of the paddle), and JOY reads the joystick.

Ease of Use

I found *Simon's Basic* powerful, but fairly difficult to learn. This is, in large part, due to the redundancy of many of the commands. A number of them could have been combined into single commands, especially the graphics and MOB (sprite) commands. In addition, the fact that none of the parameters in the commands is optional makes the language cumbersome. Once the language is mastered, however, there are unlimited possibilities. It's hard to grasp the significance of a lot of the new commands at this early stage, as they have never before been tried with BASIC

The documentation for *Simon's Basic* is complete and easy to follow. However, with a language this complex there is no way to convey all the information the beginning programmer might need to know, and there are just too many commands to give each one enough space in the manual. For this reason, I think a volume or two of tutorials for *Simon's Basic* will grow out of its use.

HCM



Biting into Your Apple

by Robert Ackerman
HCM Staff

In some ways, an Apple computer is the '32 Deuce Coupe or the '57 Chevy of the computing world: Those cars rolled out of the factory with a standard design, but once in their owners' hands, they were customized—they got chopped, channeled, sectioned; they got raked and lowered; they got tuck and roll upholstery. Their owners made them into *their* versions of the '32 Deuce Coupe or the '57 Chevy.

Open Architecture Uncovered

Apple manufactures a machine that appeals to this streak in all of us. Its computers can readily be customized by any user for any number of uses. Two features of the Apple machines encourage this customization: *open architecture* and *expansion slots*.

Open architecture doesn't refer to the physical construction of the computer (although that is also open—the top of the console is held down with two snap closures only, and pops right off); rather it refers to the information you can garner about the *functions* of all the parts. Unlike some other systems, the Apple lets you find out what's going on in any part of its system: where information is stored, what form that storage takes, where it is processed, how it is processed, and so on. And in addition to providing this information, Apple provides users with ways to take advantage of it, by equipping the machine with expansion slots. These slots allow users to insert electronic parts that expand the machine's capabilities. This not only helps individual users alter their machines, but also gives entrepreneurs easy access to an enormous market; they have enthusiastically jumped in with every conceivable device serving every conceivable purpose.

The Apple Peeled

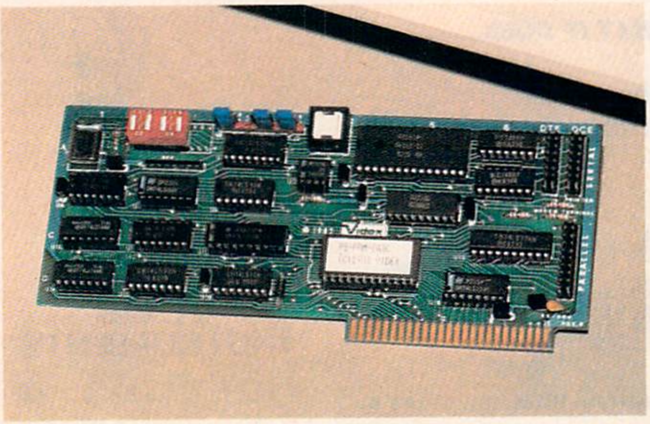
To see where these possibilities lead, we first have to look at the barebones system. The Apple IIe, which is the largest-selling Apple product these days, comes as a bare console. If you buy only this bare console, you get 64K of memory, the *resident* language, Applesoft BASIC and limited access to the machine's Assembly Language. At this stage, it has only a limited operating system, called the Monitor. If you really want to do anything with the Apple IIe, though, you need another kind of monitor—a video display. Once you have that, you're ready to begin programming.

But with that setup—console and monitor—that's all you can do: program. If you want the computer to do something, you have to type in the instructions. And at this stage, you have to type in the program lines every time you want the computer to do something, because you have no way to save any of your work. Not very handy.

Apples Branch Out

Here's where expansion begins. First of all, you could add some sort of storage device to your computer so that you wouldn't have to type in every program every time. A simple cassette recorder is the least expensive storage device available, and the Apple provides for its use. On the back of the console are two jacks for connection to a

Getting It Out



PSIO (Parallel, Serial, Input-Output) Card **$229.00**
Videx, Inc.
897 NW Grant Avenue, Corvallis, OR 97330

With your Apple IIe all set up, you have a self-contained system. You can read your favorite disks with the Disk Operating System, look at their contents on the screen, play games, write programs in a variety of languages (Applesoft or Integer BASICs, Assembly Language, LOGO perhaps) and save them on disk. But what if you want to talk to the outside world? Word processing without the ability to send information to a printer, for instance, seems a bit futile.

On the Apple IIe, you need to add a card to the system in order to operate output devices or to receive input from other devices. If you plan to use a variety of devices, the Videx PSIO card may satisfy your needs. It is a versatile card that gives you the capability to send data to, or receive it from, devices that use either *parallel or serial* input or output communication. (In most cases, this means a printer, but there are other possibilities.) The Videx card plugs into one of the vacant slots in your Apple. The manufacturer recommends that you follow the conventions used by Pascal, which ex-

Continued on p. 160

cassette recorder. With that attached, you can transfer programs you've written from the computer to a cassette, or you can transfer programs you've saved from the cassette back into the computer. But in addition to programs you've written, you can also transfer programs that other people have written from cassette tape to your computer.

Cassette tape, however, is a slow and cumbersome medium for storage of programs. For that reason, the Apple IIe system is designed to use floppy diskettes as a storage medium. And virtually all of the commercial software for the Apple computer comes on diskette.

The disk system on the Apple IIe consists of three parts: a disk drive, a card that goes into one of the expansion slots and controls the disk drive, and the Disk Operating System (DOS) software. If you pop open the top of the console by pulling up on the tabs at the back of the lid, you'll see the expansion slots inside the machine. Installing the disk controller card is simplicity itself: Just push its edge connector down into the slot you choose. (In most cases, you'll want to use slot 6. It's easy to find: The numbers are imprinted on the circuit board.)

The disk controller card connects to the disk drive with a cable that passes out through the prepositioned openings in the back of the console. You merely pop the cover off one of the openings, pass the cable through the opening and plug it into the card. The disk system comes with clamps that fix the cable to the back of the console. If you move the components around on your workspace, the clamps will prevent the cable from pulling on—and possibly damaging—the relatively weak pins on the card.

Give It the Boot

With the disk controller card installed, you can take advantage of the Apple IIe's *autoboot* feature. When the power is first turned on, the computer causes the controller to activate the disk drive, read the first file on the disk, and load the software it finds into the computer. (*Autoboot* comes from *bootstrap loader*, which was the system early computers used for loading their operating systems.) Any time you prepare a disk for use by this system, called *initializing* a disk, the system puts a copy of the operating system software on it. Because of this feature, any time you turn the computer on with an initialized disk in the drive, the computer automatically loads the Disk Operating System. (If, for some reason, you turn on the system without a disk in the drive, the drive will continue to look for a disk forever—or until you press [CONTROL] and [RESET] simultaneously.)

The Disk Operating System has all the elements necessary to use the disks effectively. You can SAVE files, VERIFY that they were stored properly on disk, and later RENAME them. The CATALOG command lets you look at a list of all the files on a disk, and the DELETE command lets you remove any file you choose from a disk. You can also load a program into the computer in order to modify it by using the LOAD command, or you can load a program and have it immediately run by using the RUN command. If there are files on the disk that you don't want to destroy accidentally, you can LOCK them; later, if you decide that you want to alter them, you can UNLOCK them. With the PR# and IN# commands, you can tell the computer which slots you want the output to go to or the input to come from. In addition, DOS has the capability to run a system with multiple disk drives. If you need more than two disk drives, you can add another disk controller card and run as many as four disk drives.

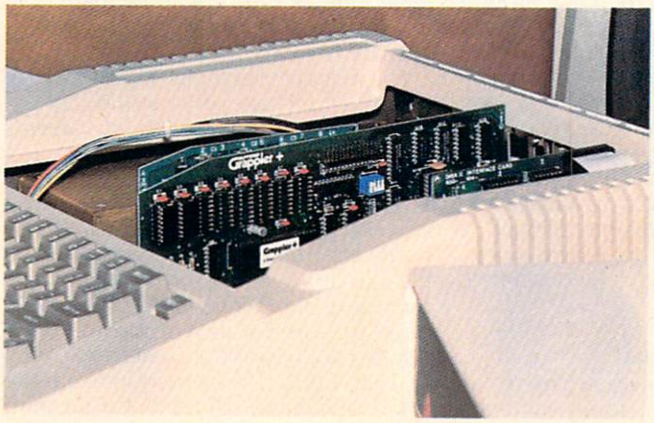
The System Master Disk that comes with the Disk Operating System contains a number of utility programs that help you manage your files and alter your programs. With them, you can copy files from one disk to another or copy entire disks. You can find out how much space is left on a disk.

The *Chain* program will load a second program into memory without destroying the first, so that you can run two related programs together.

The *Renumber* program also offers a number of program management utilities. With these utilities, you have a great deal of flexibility when you need to modify programs. After inserting or deleting program lines, you can renumber all or any portion of the program you've been working on. That feature makes it easy for you to reflect the organization of your program in the line numbering—with subroutines beginning at 1000, 2000, etc., for instance—so you can look at a program long after you've written it and have a better chance of understanding how it does what it does. The *Renumber* program also has a *merge* utility, which will take two programs or program segments and

Continued on p. 160

# of Your System



Buffered Grappler $239

Orange Micro, Inc.
1400 N. Lakeview Avenue, Anaheim, CA 92807

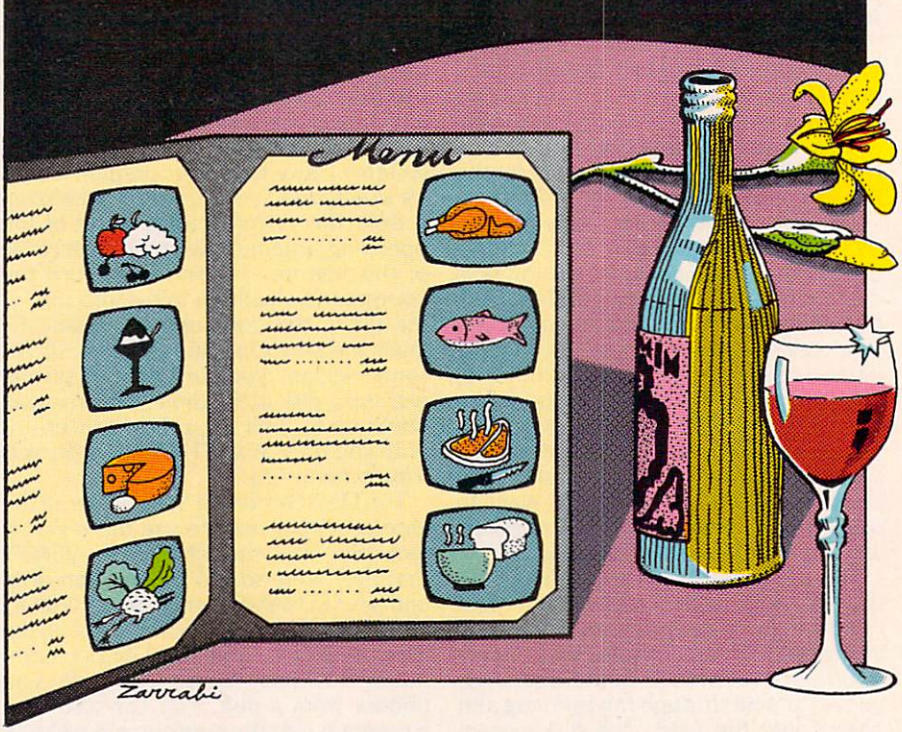
If you're looking for a printer interface board for your Apple II, II+, IIe, or III that can handle anything from text to high resolution (HIRES) color graphics, one of the Grappler+ boards is sure to fill the bill. The Grappler+ comes in three versions: a standard Grappler+, a Buffered Grappler+, and the IDS Grappler for printers manufactured by Integral Data Systems. In addition, Orange Micro makes a Bufferboard available as an add-on to the standard Grappler+, which gives it capabilities similar to the Buffered Grappler+.

The *buffering* refers to a 16K on-board RAM buffer (expandable to 32K and 64K). This frees the computer from waiting for the printer: the computer fills the buffer, and the Grappler+ then takes over the transfer of data. The Buffered Grappler+ plugs into any expansion slot on the APPLE II, II+, or IIe. It is not compatible with the APPLE III, but the standard Grappler+ teams up with the Bufferboard to provide similar functions for the III, or any of the II series.

These boards interface to a wide variety of Centronix-

Continued on p. 160

MOVEABLE • FEASTS: A • MENU • PLANNER



by Ted Martino

"Can't you do anything besides play games on that. . . .that *thing*."

Just about every home computer buff has heard that immortal line at one time or another from his or her disgruntled spouse. I was no exception!

Being one who can take a gentle hint, I could see by the smoke issuing from my wife Terri's ears that she considered my VIC-20 just another toy.

So I set out to write a program that she could call her very own. Not just any program either. She wanted it to be "practical" no less!

It wasn't as difficult as I thought. If you are in a similar situation, stop and think about the myriad of chores that your "number one" complains about. Pick a top contender and then figure out a way to have your VIC lend a hand.

In my case the answer was easy. Although there was no way to get our VIC to help with the ironing (Terri *really* hates ironing)—there was one chore it could really excel at—meal planning.

This weekly drudgery is a thing of the past, because now we let the computer choose our menus. That's right, our VIC-20 reads through tape files containing the names and locations (i.e., on what page in which cookbook) of hundreds of recipes. From this file, it checks each recipe to see if it fulfills the requirements we set at the beginning of each search.

It randomly selects the number of recipes we asked for from those that meet our requirements. Because selection is random, we get a lot more variety in our meals. Not only does this program reduce the "brain drain" on my darling wife, but she tells me we are spending less money at the grocery store each week as well. Now that's what I call practical!

It takes some ingenuity to devise fancy programs for the VIC-20 that don't quickly gobble up its memory. This is especially true for programs that use a lot of data. So any kind of data base system would at first seem out of the question for the little VIC.

Menu Planner is a system of three short programs which shows how a data base can be used with limited memory. One program creates a cassette tape file of recipe names, and the other two search the file for types of recipes.

First, you use *Make a Tape* to create a data tape containing the recipe names and the special code numbers used to identify them. Once the data is stored on tape you can run either of the other two programs. Use the *Menu Planner* program to find recipes that fit your requirements of cost, preparation time, and type of dish. *Menu Match* searches for recipes that contain any keyword you specify. Type in APPLE and the program will return APPLE PIE, APPLE STRUDEL, and so on.

It takes several minutes for the accessing programs to read in hundreds of recipe code numbers from a cassette. But you don't have to hang around waiting for the program to end. Go have a cup of coffee and come back later. Hit any key and the VIC will display the names of the recipes it has chosen and where to find the complete recipe.

The rest of this article is in two parts. First, the instructions. This is for folks who want to get the program up and running as soon as possible. Second, a brief outline of how the system works. That's for those of you who want to customize programs.

PART 1: Instructions

Before you even begin typing in the program, set aside some time to categorize your recipes. We searched our cookbooks and recipe file boxes and wrote a special code number lightly in pencil next to the name of each recipe. Then Terri and I worked as a team. I read off code numbers and recipe names while she typed at the keyboard to make the data tape.

You are probably asking "What code numbers?" You must create a code number for each recipe you want to include in the data file. It's simple to do.

Your codes will look something like this: 134/BC261 or 231MG1395. The first three digits tell the VIC three things: what kind of dish it is, how expensive it is to make, and how long it takes to prepare.

The final six characters indicate where the actual recipe can be found. For example MG1395 points to *Maida Given's Modern Encyclopedia of Cooking* and tells us the recipe is on page 1395. These final six characters may also contain a slash (e.g., /BC261).

The slash mark (/) is used as a space filler. That's because the program looks for a code number that is exactly 9 digits long. Be sure to always pad the code number with a slash or some zeros, such as 123/FC005. That would indicate the 5th recipe card in File box C (our third file box). You can use any abbreviations you like, since there are none built into the program. But your codes must have nine figures.

```
5 REM ***************
10 REM * MAKE A TAPE *
15 REM ***************
20 REM BY TED MARTINO
25 REM HOME COMPUTER MAGAZINE
30 REM VERSION 4.1.1
35 REM V20 BASIC
40 DIM B$(50),C$(50)
45 POKE 36879,27: PRINT "{SHIFT CLR}"
50 PRINT "{6 CRSR DOWN}{CTRL RVS ON}{CTRL BLU}     MENU PLANNER          TAPE PREPARATION";
55 PRINT"           {CTRL RVS OFF}":PRINT"{2 CRSR DOWN}  DO YOU WANT TO...  ";
60 PRINT "{CTRL RVS ON}1{CTRL RVS OFF}.CREATE A NEW TAPE OR{CTRL RVS ON}2{CTRL RVS OFF}.ADD TO EXISTING TAPE":GOSUB 180
65 ON-(A$="1")-2*(A$="2")-3*(A$<"1")-4*(A$>"2") GOTO 70,95,60,60
70 PRINT "{SHIFT CLR}INSERT {CTRL RVS ON}BLANK{CTRL RVS OFF} TAPE AND HIT ANY KEY":GOSUB 180: OPEN 2,1,2
75 A=A+1: GOSUB 210: INPUT B$(A)
80 PRINT#2,B$(A): IF B$(A)="END" THEN RESTORE: CLOSE 2: END
85 GOSUB 215: INPUT C$(A): PRINT#2,C$(A):IF A>49 THEN 190
90 GOTO 75
95 PRINT "{SHIFT CLR}{5 CRSR DOWN}REWIND TAPE TO BE READAND HIT ANY KEY": GOSUB 180: OPEN 2,1,0
100 A=A+1: INPUT#2,B$(A): IF B$(A)="END" THEN RESTORE: CLOSE 2:GOTO115
105 INPUT#2,C$(A): PRINTC$(A): IF A>49 THEN 190
110 GOTO100
115 PRINT"{SHIFT CLR}{8 CRSR DOWN}REMOVE OR REWIND THIS TAPE. HIT ANY KEY WHEN READY!{2 CRSR DOWN}":GOSUB180
120 OPEN3,1,2:A=0
125 A=A+1: IF B$(A)="END" THEN A=A-1: GOTO135
130 PRINT#3,B$(A): PRINT#3,C$(A): PRINT C$(A): GOTO125
135 A=A+1: GOSUB 210: INPUT B$(A): IF B$(A)="END" THEN PRINT#3,B$(A): GOTO 160
140 GOSUB 215:INPUT C$(A): PRINT "{CTRL RVS ON}{CTRL RED}     CORRECT ?  {CTRL BLK}{CTRL RVS OFF}": GOSUB 180
145 IF A$="Y" THEN PRINT#3,B$(A): PRINT #3,C$(A): GOSUB165:GOTO135
150 PRINT "{SHIFT CLR}{8 CRSR DOWN}HIT ANY KEY AND ENTER LAST RECIPE AGAIN !": GOSUB 180
155 A=A-1: GOTO135
160 CLOSE 3: END
165 IF A>49 THEN 200
170 RETURN
175 END
180 GET A$: IF A$="" THEN 180
185 RETURN
190 REM
195 PRINT "{2 CRSR DOWN}OUT OF MEMORY. YOU    MUST MAKE A NEW TAPE!.": CLOSE 2: END
200 PRINT#3,"END"
205 PRINT "{2 CRSR DOWN}OUT OF MEMORY. YOU    MUST MAKE A NEW TAPE!.": CLOSE 3: END
210 PRINT "{SHIFT CLR}{8 CRSR DOWN}ENTER THE RECIPE'S    CODE NUMBER OR <END>": RETURN
215 PRINT "{3 CRSR DOWN}ENTER RECIPE'S NAME:": RETURN
220 END
```

The first three digits carry information according to the chart in Figure 1. The first digit tells what kind of dish it is: main course, vegetable, snack, etc. The second digit represents how expensive it is to make, and the third digit represents how long it takes to prepare.

If the recipe is a main course, expensive to make, and takes more than four hours to prepare, its first three digits would be 122. A vegetable dish of average cost, quick to prepare, would be coded 234.

So make up a code for each recipe your family likes and eats regularly. Naturally you don't want to include every recipe in every book.

"*Menu Planner* could easily be improved and adapted to make your VIC-20 a small but useful data base management system."

Make A Tape

When you run *Make a Tape* for the first time, choose option 1, Create a New Tape. You will be prompted to put in a new tape and hit any key when ready. When you do, the program tells you to press RECORD and PLAY on the tape. Follow the directions and you will see the Datasette run for a few seconds and then stop. LEAVE THE BUTTONS ON THE DATASETTE ALONE.

The computer now asks you to ENTER THE RECIPES' CODE NUMBER OR END. Type the code number and hit [RETURN].

You are then prompted to enter the name of the recipe. We like to keep the name all on the same line, so that it doesn't "break" in strange places when the program prints it on the screen later. But if you want, you can enter names that are almost four lines long.

If you have never used data files on cassette before, I should warn you about one little quirk. After you have typed a dozen or so recipes, the keyboard will freeze up. You will also see the Datasette start up on its own.

Don't fret. The VIC is dumping the contents of its cassette buffer onto the tape, and the keyboard will unfreeze in a moment. The buffer only holds 192 characters, so expect to stop quite frequently.

When you have typed your last recipe, enter END when the program asks for a code number. Once again the Datasette will begin running, because the program is closing the file. Wait for it to stop, then hit the STOP button and EJECT (or REWIND) the tape.

Good news! That's the hardest part of this project. Now you are ready for your first menu planning session.

Menu Planner

First load and run the *Menu Planner* program. It will list on the screen the same choices as in Figure 1.

After you have selected the type of recipe, you are asked how many recipes you want. Key in any number and hit RETURN. If you have asked for more recipes than the computer later finds on the tapes, it will show you as many as it found.

The program now displays some instructions. Just follow the directions and rewind the tape if necessary. Hit any key and the computer will tell you to PRESS PLAY ON TAPE.

After reading the tape, the *Menu Planner* will tell you how many recipes it found that match your requirements. Now every time you hit a key it will display the name of a recipe and the code number which represents the name of the cookbook, and the page number.

Figure 1

TYPE	EXPENSE	PREPARATION TIME
1.Main Course	1.Very Expensive	1.Begin it the Night Before
2.Vegetable	2.Expensive	2.Four Hours or More
3.Salad	3.Average	3.One to Four Hours
4.Dessert	4.Inexpensive	4.Less than one Hour
5.Breads	5.Any	5.Any
6.Holiday Meals		
7.Snacks		

NOTE: *You may freely mix all categories on a tape. But it can sometimes speed things up if all recipes on a tape are of the same type. (i.e., main courses only on tapes 1, 2, and 3; vegetables on tapes 4, 5, and 6, etc.)*

Menu Match

The final section of this program is called *Menu Match*. Using the same data tapes you already made up, it looks for a particular dish according to a keyword you give it. For example, Christmas is over and you have a ton of leftover turkey. Load *Menu Match*, type in TURKEY, and it tells you to put in the data tape...etc.

When it is done reading the tape it gives you a list of every recipe name that has the word "turkey" in it. Watch for words that are embedded in other words. For example "ham" will return "Hamburger," and "nut" will return both "Butternut Squash" and "Minute Steaks".

```
5 REM ****************
10 REM * MENU PLANNER *
15 REM ****************
20 REM BY TED MARTINO
25 REM HOME COMPUTER MAGAZINE
30 REM VERSION 4.1.1
35 REM V20 BASIC
40 DIM NU$(51),NA$(51),C(51)
45 PRINT"{SHIFT CLR}{7 CRSR DOWN}{CTRL RVS ON}{CTRL RED} MENU PLANNER {CTRL BLK}"
50 FORX=1TO3000:NEXT X
55 PRINT"{SHIFT CLR}{CRSR DOWN}1.MAIN COURSE":PRINT"{CRSR DOWN}2.VEGETABLE":PRINT"{CRSR DOWN}3.SALAD":PRINT"{CRSR DOWN}4.DESERT"
60 PRINT"{CRSR DOWN}5.BREADS":PRINT"{CRSR DOWN}6.HOLIDAY MEALS":PRINT"{CRSR DOWN}7.SNACKS"
65 PRINT"{4 CRSR DOWN}CHOOSE ONE OF THESE":INPUTA$:IFVAL(A$)<1ORVAL(A$)>7THEN 55
70 PRINT"{SHIFT CLR}{CRSR DOWN}1.VERY EXPENSIVE.":PRINT"{CRSR DOWN}2.EXPENSIVE.":PRINT"{CRSR DOWN}3.AVERAGE."
75 PRINT "{CRSR DOWN}4.INEXPENSIVE.":PRINT"{CRSR DOWN}5.ANY."
80 PRINT"{4 CRSR DOWN}CHOOSE ONE OF THESE":INPUTB$:IFVAL(B$)<1ORVAL(B$)>5THEN 70
85 PRINT"{SHIFT CLR}{CRSR DOWN}1.NIGHT BEFORE":PRINT"{CRSR DOWN}2.MORE THAN 4 HOURS":PRINT"{CRSR DOWN}3.ONE TO FOUR HOURS"
90 PRINT"{CRSR DOWN}4.ONE HOUR OR LESS":PRINT"{CRSR DOWN}5.ANY
95 PRINT"{4 CRSR DOWN}CHOSE ONE OF THESE":INPUTC$:IFVAL(C$)<1ORVAL(C$)>5THEN 85
100 PRINT"{SHIFT CLR}{8 CRSR DOWN}HOW MANY RECIPES WOULDYOU LIKE":INPUTB
105 PRINT"{SHIFT CLR}NOW PUT TAPE TO BE READ INTO CASSETTE ANDHIT ANY KEY WHEN READY
110 GETZ$:IFZ$=""THEN110
115 OPEN2,1,0
120 A=A+1:IFA>50THENA=A-1:GOTO165
125 INPUT#2,NU$(A):IFNU$(A)="END"THEN165
130 INPUT#2,NA$(A):PRINT NA$(A)
135 IFA$<>LEFT$(NU$(A),1)THENA=A-1:GOTO120
140 IFB$="5"THEN150
145 IFB$<>MID$(NU$(A),2,1)THENA=A-1:GOTO120
150 IFC$="5"THEN120
155 IFC$<>MID$(NU$(A),3,1)THENA=A-1:GOTO120
160 GOTO120
165 A=A-1
170 CLOSE2:PRINT"{SHIFT CLR}":PRINT"{CTRL RED}FOUND"A:PRINT"RECIPES THAT MATCH":IFB>ATHEN B=A-1
175 IF B<0 THEN240
180 C=RND(-TI):IF A<B THEN B=A
185 FOR D=1 TO B
190 C(D)=1+INT((A-1)*RND(1))
195 FOR X=1 TO D
200 IFC(D)=C(X-1)THEN190
205 NEXT X:NEXT D:FORD=1TOB:PRINT"{CTRL GRN}HIT ANY KEY FOR A RECIPE{CTRL BLK}"
210 GETZ$:IFZ$=""THEN210
215 PRINTNA$(C(D)):PRINTRIGHT$(NU$(C(D)),6)
220 NEXT D
225 PRINT"{CTRL RED}{CTRL RVS ON}MORE? (Y/N) {CTRL BLK}{CTRL RVS OFF}"
230 GETZ$:IFZ$= "" THEN230
235 IFZ$="Y" THEN RUN 55
240 END
```

OK. Everyone who just wants to run the program should now get to work typing. The rest of us will now meet in PART 2 for insights into how the programs actually work.

PART 2: Programming Information

The *Menu Planner* system uses tape files instead of DATA statements within the program to save memory space. By using tape files, the computer only has to store the recipe names that match your requirements. With DATA statements, our little VIC would have to hold every recipe, whether it was a match or not. These would soon fill the memory.

With tape files, your computer can scan literally millions of non-matching recipes. Of course, using tape storage we are limited by the time it takes to read large amounts of data. But for our purposes, we have eliminated the problem of the VIC-20's limited memory.

As a precaution against filling the memory with lots of matching recipes and crashing the program, I put an arbitrary limit on how many matching recipes the VIC will accept. This version of the program, for the unexpanded VIC, will only accept 50 recipes that match your requirements.

When we create a new tape with the *Make a Tape* program, we first OPEN the cassette file, and use a 2 in the secondary address, the third number after the OPEN statement. The secondary address tells the VIC what kind of file this is. It is limited to 0, 1, or 2 for the cassette. Our choice, 2, writes to the tape, and puts an end of tape marker after the file is properly closed.

The first number following OPEN, called the file number, can be any number from 1 to 255. The number in the middle of the OPEN statement is the device number. It tells the VIC which peripheral it will be communicating with. Of course the number 1 selects the Datasette.

And now for those of you who want to customize the program, here is a brief outline of how *Menu Planner* works.

Lines 45-100 put the title on the screen, and input the requirements for the upcoming search. Lines 105 and 110 provide a waiting loop so the user can double-check that the data tape is ready.

The DIMension statements in line 40 tell the VIC how many array variables to expect. These DIM statements, along with IF A > 50, set the arbitrary limit I talked about earlier.

```
5 REM **************
10 REM * MENU MATCH *
15 REM **************
20 REM BY TED MARTINO
25 REM HOME COMPUTER MAGAZINE
30 REM VERSION 4.1.1
35 REM V20 BASIC
40 DIM NU$(50),NA$(50)
45 POKE 36879,27: PRINT"{SHIFT CLR}"
50 PRINT"{8 CRSR DOWN}{CTRL PUR}{CTRL RVS ON} MENU PLAN MATCH THE WORD ";
55 PRINT "{CTRL RVS ON} {CTRL BLK}"
60 FOR Z=1 TO 2500: NEXT Z
65 PRINT "{SHIFT CLR}{8 CRSR DOWN}WHAT WORD WOULD YOU LIKE TO MATCH?": INPUT A$: B=LEN(A$)
70 PRINT "{SHIFT CLR}{9 CRSR DOWN}INSERT A DATA TAPE, REWIND IT AND HIT ANY KEY WHEN READY!"
75 GOSUB 160
80 PRINT "{SHIFT CLR}{9 CRSR DOWN}"
85 OPEN 1,1,0: PRINT "{SHIFT CLR}{10 CRSR DOWN}{CTRL RED}{CTRL RVS ON} PLEASE WAIT! ";
90 PRINT "{CTRL RVS ON} {CTRL BLK}{CTRL RVS OFF}";
95 A=A+1: IF A>50 THEN A=A-1: GOTO130
100 INPUT#1,NU$(A): IF NU$(A)="END" THEN130
105 INPUT#1,NA$(A)
110 FOR C=1 TO LEN(NA$(A))
115 IF B>LEN(MID$(NA$(A),C,B)) THEN A=A-1: GOTO 95
120 IF A$=MID$(NA$(A),C,B) THEN M=M+1: GOTO 95
125 NEXT C
130 PRINT "{SHIFT CLR}": GOSUB 190
135 IF M<>0 THEN RESTORE: FOR D=1 TO (A-1):PRINT NA$(D):PRINT RIGHT$(NU$(D),6): GOTO145
140 GOSUB 170: GOTO 185
145 IF D<>A-1 THEN PRINT "{CRSR DOWN}{CTRL CYAN}HIT ANY KEY FOR MORE{CTRL BLK}": GOSUB 160: NEXT
150 GOSUB 175: GOTO 185
155 IF Z$="Y" THEN RUN 65
160 GET Z$: IF Z$="" THEN 160
165 RETURN
170 PRINT "{10 CRSR DOWN}NO MATCHES WERE FOUND ON THIS TAPE !"
175 PRINT "{2 CRSR DOWN}{CTRL BLU}DO YOU WANT TO TRY AGAIN? (Y/N)": GOSUB 160: IF Z$="Y" THEN RUN 65
180 RETURN
185 PRINT "{SHIFT CLR}{CTRL BLU}": END
190 POKE 36878,15: POKE 36876,220: FOR L=1 TO 5:NEXT: POKE 36876,0:FOR L=1 TO 150: NEXT
195 POKE 36876,200
200 FOR L=1 TO 5: NEXT: POKE 36876,0: POKE 36878,0: RETURN
```

The data base processing of the program happens in lines 115 to 170. We OPEN the file (which we call file 2) on device #1 (the cassette), and tell the VIC that it is a *read only file* (that's the meaning of the 0 in the secondary address).

We then INPUT the special code numbers and recipe names one at a time and check them. If the VIC finds a match, it stores that information in the previously DIMensioned arrays by incrementing the counter variable A.

Once the tape is read, the program closes the file and tells us how many recipes it matched. Then it moves into the random selection process.

Lines 180 through 205 generate as many random numbers as recipes we asked for at the beginning of the search (unless we asked for more than were found). Then it prints one less than the actual number found. Taking one less random number significantly increases the speed of selection.

The remainder of the program prints out each recipe's name and code number. Finally the program displays MORE?. Typing a Y for Yes takes you back to the beginning of the program. Hitting any other key will end the program.

The *Make a Tape* program begins with a title display and mini menu, from lines 50 through 60. Line 65 determines which path was selected. If you push any button except 1 or 2, you will be sent back to line 60 to try again. Lines 70 through 90 open the tape file and allow you to input the data. They also then output your data onto the tape via the PRINT# statements.

The Add to Existing Tape option begins at line 95. It is similar to the section which reads in info from the *Menu Planner*, but instead of testing each entry, it just reads it into memory. Later (lines 115 to 160) these same entries are returned to another tape file, but the file is left open, so you can add more recipes. Again, when you type END, it closes this new file and ends the program.

The *Menu Match* program uses the same principles as the other two. The real work is done in lines 95 through 125. Here is where the recipe's name is read in and checked for a match.

In order to speed up execution time, line 115 first looks to see if there are fewer unchecked characters in the name than in the key word. If they are less, then there is no possibility of a match and that recipe is rejected.

The program is written in three parts to leave as much of the VIC's memory free as possible. Minor modifications will allow the parts to be merged into one neat package if you have a memory expander, or a Commodore 64. Although *Menu Planner* is a simple and specific system, it could easily be improved and adapted to make your VIC-20 a small but useful data base management system.

Now if anyone out there finds a way to have the VIC do the ironing... HCM

Biting Into . . . from p. 155

from p. 155

make one program out of them. If you then save the merged program, you have a permanent copy of it.

Playing the Slots

The Disk Operating System is just the beginning of customization with the Apple, however. You still have seven open slots you can fill and a game port to plug into. Although the game port is primarily for the gaming enthusiasts—it's where they plug in joysticks or game paddles—other, custom, devices have been developed that plug in here as well.

One of the first choices for the other slots is usually a card that changes the screen display. Normally the screen displays 40 characters across its width; with an *80-column card*, it displays 80 characters on each line. With an 80-column card in place, the screen contains much more information and scrolls less often.

One of the first additions many computer owners want to make to their system is a printer. A number of cards are available that allow you to transfer the information you generate from its electronic form to ink on paper. (See the accompanying sidebars, "Getting It Out Of Your System," for two examples.) Cards exist for serial printers and/or parallel printers; they may have their own data storage (*buffers*) to speed up data transmission; they may receive data at a variety of speeds which you can select—the possible combinations can be bewildering.

Once you get beyond printers, the number of cards and their related peripherals is nothing short of mindboggling. There are cards that allow you to communicate with distant computers over phone lines. Other cards cause your computer to *emulate*, or act like, a computer terminal rather than a computer. Still other cards make it emulate other computers—these allow it to use software designed for those other computers.

You can plug in cards that add memory to your system. You can add cards that augment the sound capabilities, or others that enhance the graphics capabilities. If you need to have the *exact* time all the time (perhaps for some home energy control system), there are cards that will give it to you. There are cards that make the software you put on a diskette uncopyable, and there are other cards that allow you to copy any software on diskette—including uncopyable software.

Still other cards connect with a variety of peripheral devices, from printers to graphic tablets, from keyboard music synthesizers to speech synthesizers. In other words, you can probably find the ready-made parts necessary to customize your Apple in whatever way you can imagine. And if you can't find them, but have the inclination and skill, the information is available for you to do it yourself. Many a thriving new business has been started just this way. HCM

PSIO . . . from p. 154

pects to find a printer in slot 1, modem in slot 2 and remote terminals (like the 80-column card) in slot 3. You can, however, choose any slot that is convenient for your computer system.

The Videx PSIO offers a number of ways to set its parameters for your particular printer. The easiest way is to use the Utilities Disk, which comes with the card, and select line 3 of the menu when it comes up. You can then step through the configuration using the arrow keys or the [RETURN] key, examining the choices and altering them as you like. Here you'll notice one interesting feature of the Videx PSIO card: Even though it physically resides in one slot, it can act as if it occupies two slots. This card can put the serial interface in one slot and *phantom* the parallel interface to another slot. Once you've selected a particular configuration, you can save that configuration on the card permanently—or semi-permanently. If you open up the computer at this point and press the little square white NOVRAM button on the card, you'll hear a faint beep. That means your configuration is now on the card, will reappear on the card when you turn power on again, and remain until you repeat the whole process.

Although that configuration is permanent, you aren't limited to it alone. You can change part of the configuration temporarily in at least four ways:

1) from a program with PRINT statements;
2) from a BASIC program using BRUN with a configuration you've previously saved from the Utilities Disk menu as a binary file;
3) from the deferred mode in the DISK ACCESS menu; or
4) by sending configuration parameters to the PSIO card directly from the keyboard (or, using the EXEC command, as if they were being sent from the keyboard).

If you want to change the configuration via a program, address the slot holding the card (either real or phantom) and PRINT the parameters preceded by [CONTROL] [I]—or

Buffered Grappler . . . from p. 155

compatible parallel printers such as Epson, NEC, Okidata, Star Gemini , and Anadex. The IDS Grappler+ is also compatible with printers such as the Prism, Microprism and 450/560 printers.

For our tests, we used the Buffered Grappler+, an APPLE IIe, and an Epson MX-80 printer. Hook-up was easy and was quite clearly covered in the small, but complete 25-page manual that comes with the interface board. The manual took us step-by-step through the text and graphics commands needed to use the Grappler+, and included a short section on the additional commands for an Apple IIe equipped with the 80-column card.

We found the text commands easy to use. Text features include control of automatic line feed and carriage return, margin settings, and page length. Printout can be created in a format identical to the screen display (either 40- or 80-column), or with a variety of user-selected margins and page lengths. These formats can be changed from either program mode or immediate mode using simple PRINT statements. The exact syntax for all the commands is clearly spelled out in the manual.

The graphics commands allow you to print either of the HIRES pages to the printer, and even include a feature to print them both side by side, with just one command. To test the graphics capabilities of the Grappler+ we used the program described in "3D-IIe" (in this issue). By adding the following line, we obtained printouts of all the 3-D figures displayed on the screen:

```
221 PR#1: PRINT CHR$(9);"G": PR#0
```

This line simply prints the HIRES graphics page and then returns to the program. We used PR#1 in our PRINT statements as we had the card in slot #1. However, the card can go in any slot, with the PR# changed accordingly.

Using the Buffered Grappler+ in our tests, we found we could dump up to three figures before we noticed any significant slowdown in program operation. With the addition of more RAM, this could be stretched to even more pictures before the printer would have any effect on program time.

We also added these two short statements to the program:

```
891 PR#1
:
931 PR#0
```

These lines label the drawings by sending selected text to the printer along with the graphics. Examples of the printouts we obtained are included in Figures 1 and 2.

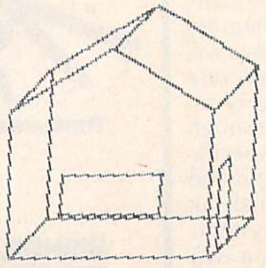


Figure 1

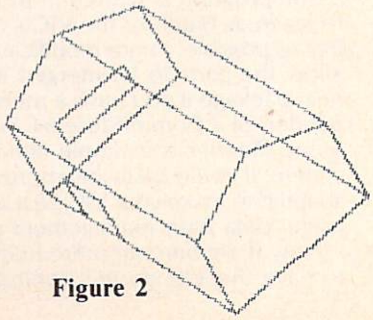


Figure 2

CHR$(9). This will reconfigure the card. For instance, if the serial interface occupies slot 1 and is set to 9600 baud for a serial printer, and for some reason you need to change to 4800 baud, the statement:

```
PRINT CHR$(4);"PR#1":PRINT CHR$(9);"12B":PRINT
CHR$(4);"PR#0"
```

will change the baud rate temporarily.

From the Utilities Disk, you can also load files with configurations preset for various devices and use those to reset the PSIO card. If none of the configurations that come on the disk matches your device, you can alter one that's close and then save that with a file name for your device. If you merely want to alter configurations that you'll put in a disk file, you can choose the deferred mode in the DISK ACCESS menu. The configuration choices then *don't* alter the card, but *will* be put on disk when you save a file. To reconfigure the card is then a matter of loading that configuration and pressing the NOVRAM button.

You can get the Videx PSIO card to dump graphics to your printer as well, and it will provide all the same sorts of configuration possibilities as it does for text. Since graphics printing is strongly dependent on the particular printer (as opposed to the RS232 serial or Centronics parallel protocols for text), successful operation will require a good deal of experimentation with parameters. "Setting up the PSIO [card] to transfer a high resolution picture to a new printer is not for the faint of heart," the *PSIO Supplement* warns on page one of its section on graphics configuration. If you need to dump graphics to your printer and it isn't one of the standards anticipated by the Utilities Disk, be prepared for some expenditure of time and effort, as well as a bit of frustration.

Keeping track of all the configurations for all the possibilities could be a chore if you had to load each configuration and display it on the screen to find out what you had. But the Utilities Disk allows you to print the configurations, so that you can have a printed record of each. Besides the configuration information, the Utilities Disk also includes addenda to the manuals and a section on use of the disk programs.

If you have to output to, or input from, a variety of different devices at different times, need both parallel and serial interfaces, need to dump high resolution graphics, or need to change the card's parameters under program control, the Videx PSIO card is a good choice. *—Robert Ackerman*

HCM

In adding the lines above we discovered one apparent flaw in the manual's explanation of how the printer interface is accessed from a program. According to the manual, PRINT CHR$(4) (where CHR$(4) is the ASCII form of [CTRL][D]) is necessary when using DOS. We found that our program ran fine without the PRINT CHR$(4), and in fact did not work correctly when we included it. The manual contains simple Applesoft and Integer BASIC program examples demonstrating the graphics commands. We found that changing statements like:

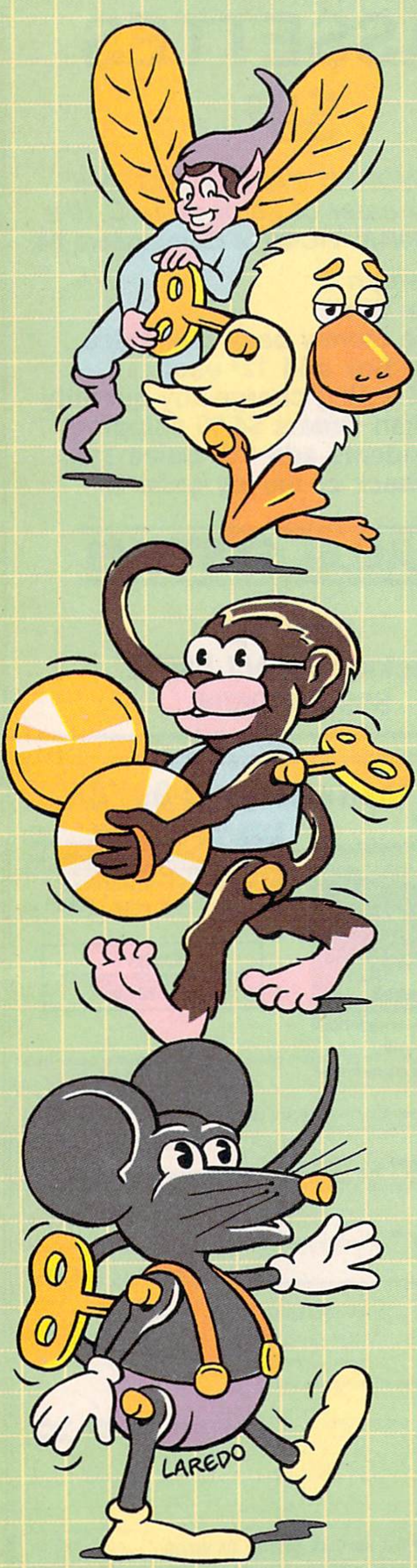
PRINT CHR$(4);"PR#";S

to

PR#S

where S is a variable representing the slot where the Grappler+ was plugged in, did not change the output at all, even though we were using DOS. Including the PRINT CHR$(4) statement in the *Applesoft 3-D Graphics* program stopped the printer output altogether. We talked with representatives of Orange Micro who explained that although the Apple DOS manual states that PRINT CHR$(4) is required with all DOS commands, the PR# statements nevertheless work fine by themselves in Applesoft without preceding them with the PRINT CHR$(4). One of the programmers told us that he never has any trouble if he omits the PRINT CHR$(4) and substitutes PR# statements when using Applesoft. Armed with this knowledge, you can expect an easy and successful interface with the Grappler. *—Roger Wood*

HCM



Don't Be a SlowPOKE:

Use Automatic Sprite Motion

by John Thrasher
HCM Staff

In addition to the great variety of graphic characters and symbols at our disposal on the Commodore 64, we also have eight *sprites* that bring speed and ease of motion to computer graphics. Because they are positioned by pixel location instead of character location, you can put sprites in any spot on the screen that you like. Then by simply changing the sprite position (via a POKE to the sprite position register), you can make the sprite move smoothly across the screen.

As long as your program remains simple and you don't try to move too many sprites at once, POKEing values is not too much of a chore. But as you make your program more complex this becomes difficult and cumbersome. In order to get good animated motion, your BASIC program must constantly POKE new values into the sprite position registers. While this is happening, other tasks in your program have to wait. Wouldn't it be nice if you could just start your sprite (or sprites) moving at a specified speed and direction and not have to bother with all those POKEs? The Auto Sprite Motion routine lets you do just that.

Auto Sprite Motion is a machine language routine that automatically moves sprites across the screen. It frees your Commodore 64 BASIC program from the continual POKEing of values into the sprite position registers. Thus it saves time and program space, while allowing for the smooth, simultaneous movement of all eight sprites at varying speeds in any directions you choose. The reason we say it is "automatic" is that it is *interrupt-driven*.

You might wonder what we mean by interrupt-driven. The 6510 microprocessor inside the Commodore 64 has an input pin called the Interrupt Request line (IRQ). This interrupt line is connected to a timer (the jiffy clock) that sends a signal to the 6510 sixty times every second. When the 6510 receives this signal, it automatically transfers control from the main program to what is called the *interrupt service routine*. This routine takes care of timing, keyboard scanning, and other "housekeeping chores." Before it transfers to this interrupt routine, the 6510 stores the pertinent information that returns control operation to the main program when the service routine is complete. You can think of this as marking your place in a book. During the power-up routine, the operating system (or KERNAL) has stored the address of the interrupt service routine at locations 788 and 789. This address is called the *interrupt vector*. For our program to be "driven" by this interrupt, we must replace the KERNAL's interrupt vector (at 788 and 789) with our own. Then at the end of our routine we must return control to the KERNAL's interrupt routine so that the housekeeping chores mentioned above are properly carried out.

Loading the Program

Although machine language routines may be loaded anywhere in RAM, we must locate them with an eye to several considerations. We don't want them to interfere with the BASIC program or with system variables and pointers. This effectively eliminates the first 40,960 bytes. We also do not want to have our routine "hidden" underneath any of the system ROMs because most programs using sprites are generated by BASIC programs which must use these ROMs. To make this Auto Sprite routine as easy as possible to implement, we will use about 700 bytes in the 4K block of RAM from addresses 49152 ($C000) to 53247 ($CFFF).

We'll load our assembly language routine beginning at address 50880 ($C6C0). We are actually loading two routines: the Auto Sprite Motion routine and the Change Interrupt Vector routine. The Change Interrupt Vector routine begins at the end of the Auto Sprite Motion routine (51,104 or $C70A). We access it with the BASIC command, SYS 51104. We don't need the SYS command to access the Auto Sprite Motion routine because after the program executes SYS 51104, Auto Sprite Motion is invoked automatically via interrupts. The BASIC program listing for this article loads these machine language programs into memory in lines 180 through 340.

If you plan to use this routine in more than one program, we recommend that you first type in these DATA statements and create a sequential data file, on either disk or tape. This way you won't have to retype and debug for each new program. The following lines show you how to create a sequential file on tape:

Continued on p. 168

LIMITED SUPPLY

Back Issues of 99'er HOME COMPUTER magazine

Each Only $3.95 postpaid

Canada $4.50, Foreign Surface $5.50, Foreign Air $7.50

ISSUE #1 ISSUE #3 ISSUE #5
ISSUE #2 ISSUE #4
OUT OF PRINT

ISSUE #6 (Partial Contents)
• How To Produce Sound Effects • Debugging a Game Program • How to Start a User's Group • Verbose: A speech Vocabulary Expansion Aid • Color Mapping • Dynamic Manipulation of Screen Character Graphics • The Beginner's Guide to Cassette Operation With the Home Computer • Pre-School Block Letters and Data Compaction • Picking the Ponies in TI BASIC • Battle Star Space Game • 3-D Animation on the Home Computer • Programming Tips • Who is LOGO for? • Tower of Hanoi in TI LOGO • A Review of the TI Lesson-Development Software • An Interview with a Game Designer • Learning Assembly Language with a Magic Crayon • and much, much more.

NOVEMBER 1982 (Partial Contents)
• Chatting with Your Micro: Languages for the Home Computer • A Review of the Smith Corona TP-1 Daisy Wheel Printer • The Micro Jaws Arcade Game • A Knight's Tour in TI BASIC • LOGO Has Style • ASPIC: A Language for Children • A p-System Beginners' Tutorial • An Interview with a p-System Pioneer • A Mini-Memory Screen Dump to the Home Computer Printer • Up Scope!—An exciting Undersea Combat Game • Strategy for Munch Man • A Brief Encounter with a TI Hand-Held Computer • 99'er Shopping Bus • A Pocket Battleship • Sub-Programs in Extended BASIC • Arcade & Adventure Game Reviews • and much, much more.

DECEMBER 1982 (Partial Contents)
• Tex-Scribe: A Text Editor for the Home Computer • A Christmas Computer Carol • Managing a Mailing List the Futura Way • Parsec: The Arcade Game • Plotting With the Home Computer—Pixel by Pixel • Preventing the Situation—Oh No! Memory Full • A Colorful Tour of TI-Fest: The Home Computer Show • Santa's Workshop: The Making of a Home Computer • The Turtle Arcade: Movies & Video Games in LOGO • Controlling a BASIC Termite • The 99'er Gold Rush—An Arcade/Adventure in the Home • 99'er Digest of News & Happenings in the TI World • Plus Games, Reviews, and much, much more.

JANUARY 1983 (Partial Contents)
• Computer Assisted Instruction for the Handicapped • p-System Basics • Debugging in LOGO • The Dow-4 Gazelle Flight Simulator • Note Whiz and Pitch Master Musical Game Reviews • Learning With the PLATO Computer Library • Strategies for Adventure Gaming • Death Drones • Using the Line-By-Line Assembler • Close Encounters of the Simon Kind • Electrical Engineering Education Program • Interview With an Arcade Game Designer • TI Invaders • Programming With Pascal • Cyber-Dice • News and Happenings in the Home Computer World • Arcade Game Reviews • The Thief Adventure Game • Programming Tips • and much, much more.

FEBRUARY 1983 (Partial Contents)
• Texas Instruments at the Winter Consumer Electronics Show • Home Computer Printers on Review • How to Create Math Daisies in LOGO • Vectors in LOGO • ASPIC: A Language for Teachers • The Joys of Adventuring—Part 2 • Programming Pointers with Chuck-A-Luck—Part 4 • Interview With the Voice of Parsec • Why You Need a Printer for Your Home Computer • Lifeline to Titan Space Game • Night Blockade Battleship Game • Tower of Hanoi Pocket Program • Computer Gaming Software Reviews • News of Late Developments in the World of Home Computers • and much, much, more.

MARCH 1983 (Partial Contents)
• An Introduction to the TI-99/2 Basic Computer • The Hex-bus and the 4/A Connection • Making Your Own Say and Spell Game • Disabled Children Learn and Grow • Super Cataloger—A Review of a Disk Library Utility Program • TI's New CC-40 Compact Computer • Robots and Their Social Impact • Twenty Questions With Robot Redford • The Gravity of LOGO • Joystick Jockey—An Overview of Remote Controllers • Parsec Strategy • Converting Extended BASIC to Assembly Language • Matrix Muncher • Mini Memory Disassembler Utility • Pulling the Shade on Sprites • Letters on LOGO • Tiny Tutorials • Games programs, reviews, and much, much more.

APRIL 1983 (Partial Contents)
• Computer Assisted Savings Planning to Build Your Nest Egg • Tex-Cipher Writes and Decodes Secret Messages • Crossbytes—Computer Vocabulary Crossword Puzzle • Cutting Corners On Your Food Budget Using Coupons • Introducing Financial Planning with Multiplan • The Design Philosophy of the Compact Computer • LOGO Takes On the Popular Fifteen Puzzle • Super Language—Programming Sprites in Mini Memory • Colorful Word World—Reading Readiness for Pre-schoolers • Gameware Buffet's A-Maze-ing Boa Alley Game • Giant and Dwarfs Entrapment Game • Game Reviews • Programming Tips • Money Saving Hints • and much, much more.

MAY 1983 (Partial Contents)
• A Consumer's Guide to Word Processing • Word Processing Market Basket • A Generalized Filing Program for VIPs • The Multiplan Medium Balances Your Checkbook and Budget • Activity Accountant Helps School Secretaries with Extracurricular Activities • Maximizing Your Mini Memory's 4K of RAM • Exploring Enhanced BASIC on the Compact Computer • The LOGO Tortoise Debates the BASIC Hare • A Pocket Program to Organize Data with Linked Lists • Mentally Handicapped Learners Team Up with the TI-99/4A • The Wonders of Diskette Storage • Beeline—a Multi-Screen Strategy Game • Lost Ruins—an Archeological Dig Game • 3-D Illusions with Sprites in Depth • Game reviews, Group Grapevine, and much, much more.

JUNE 1983 (Partial Contents)
• Children and Computers Make the 99/4H Connection • Tune Your Guitar with Our TI Tuning Fork • Talk to Your Computer—Voice Technology Is Here • Gameware Buffet's Eat or Be Eaten Aardvark Game • Protect Your Station in the Space Zapper Game • What Multiplan Can and Can't Do • Understanding Inputs and Outputs in Drive For Diskettes—Part 2 • Calculate Loan Schedules on the CC-40 • Go on a LOGO Vacation • Letters on LOGO • A Review of Upper Room Software's Programs for Special Learners • Construct an RS232/Joystick Interface • Group Grapevine • Shopping Bus • A Natural Language Interface for the Professional • Game Reviews • and much, much more.

JULY 1983 (Partial Contents)
• The Evolution of Home Computer Graphics Comes Alive in Graphics Grows Up • Five Data Organizers in Never Out of Sorts • TI & 99'er at the Consumer Electronics Show • WarGames: The Movie and the Book • Editing with Multiplan • The LOGO Logician Presents To Model Is to Learn • LOGO Mosaic Designs Fill the Screen • Your Speech Synthesizer as a Spelling and Foreign Language Teacher • Software for Your Low-cost Printer Port • Gameware Buffet's Treasure Island and the Colorful Switch-A-Row • A Book Review of Learn BASIC for CC-40 Users • 3-D Animation with the TMS 9918A Video Chip • Games Reviews • Group Grapevine • and much, much more.

AUGUST, 1983 (Partial Contents)
• The Home Computer Goes To Work • Bit One, Purl Two at the Fashion Factory • Better Business Bar Graphs in Graphic Persuasion • An Ensemble of Assemblers • Cashflow Helps Money Management • Keystrokes for Thrifty Folks—a Review of Typwriter • Game Reviews of Cavern Quest and Starprobe '99 • Counting Fun for Preschoolers • Peripheral Vision 99 • Mean Machines and Small Potatoes • Multiplan Medium Groups Cells into Rectangles • Turtle Text, a LOGO Word Processor • Group Grapevine • Hello, Little Brother—CC-40 Speaks to 99/4A • Plato's Progress Looks at Student Assessment • Pocket Sunrise • Public Investigator • Gameware Buffet's Jungle Jim, Success Formula, and much, much more!

SEPTEMBER, 1983 (Partial Contents)
•Adding on to Your Home Computer • 99'er Directory of Commercially Available Software, Accessories and Peripherals • Peripheral Vision 99 Hardware Reviews • Pocket Sunrise, Part Two in Extended BASIC • Byte Lightning—Mini Memory Plays a Game • TI-WRITER Tutorial • LOGO's Functions, Sets and Turtles • The CC-40 and 99/4A Take the Data, and RUN • PLATO's Progress Looks at Geometry Courseware and the Shape of Things to Come • Gameware Buffet's Challenge of Camelot and The Fly • Game Reviews of Nit Wit and Crime and Punishment • Group Grapevine • 99'er Hall of Fame • 99'er Digest Update on New Products, and much, much more!

OCTOBER, 1983 (Partial Contents)
• Adventures and Fantasy with Your Home Computer • Do-It-Yourself Adventure • Picoprocessor Emulator, a 4-Bit Microprocessor • Have No Fear: Assembly Language Won't Byte, Part 1 • Make Your Mark • Les Izmore and Debug Computing Cartoon • Once Upon a Tortoise Shell—A LOGO Adventure • Turtle Strut • Number Nibbler for Children • Lots of Plots on Your CC-40 and HX1000 Four-Color Printer/Plotter • Multiplan Medium—Bartender • A Grisly Adventure Bear Hunt • Escape From Wizard's Keep in Extended BASIC • Game Reviews • Shopping Bus • and much, much more.

NOVEMBER, 1983 (Partial Contents)
• Education with your Home Computer • Five Creative Learning Activities for Children • Let's Build America • Have No Fear: Assembly Language Won't Byte, Part 2 • Squeezing the Most Out of TI BASIC • TI-WRITER Tutorial • LOGO Lexicon • Interview with Dale Osborn • TI-WRITER: At Home In the Office • PLATO's Progress • Les Izmore and Debug • The Multiplan Medium • Computer Arrested Instruction—99'er Interviews the Kids • Gameware Buffet: Taco Man and Robo Chase • Game Reviews of Jail Break and Arithmetex • Hall of Fame • 99'er Digest • and much, much more.

Keep Your Back Issues & Tapes Together With a FINDER-99'er-BINDER T.M.

Only $11.95

- Big enough to hold 6 magazines and 12 tapes
- Uses wire straps to hold magazines so that no hole punching is necessary
- Attractive and Durable

$16.95 with 6 Blank Tapes

$21.95 with 12 Blank Tapes

12 High-Quality 99'er-WARE C-10 Digital Computer Cassettes Available Separately for $14.00

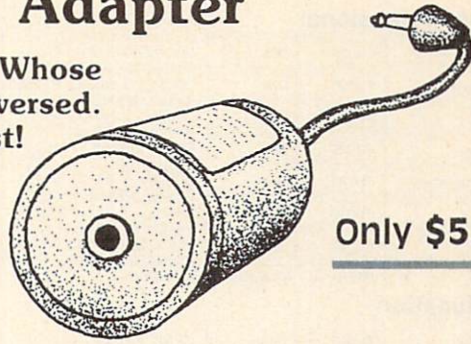
99'er.WARE™

P.O. Box 5537
Eugene, Oregon 97405
Tel. (503) 485-8796

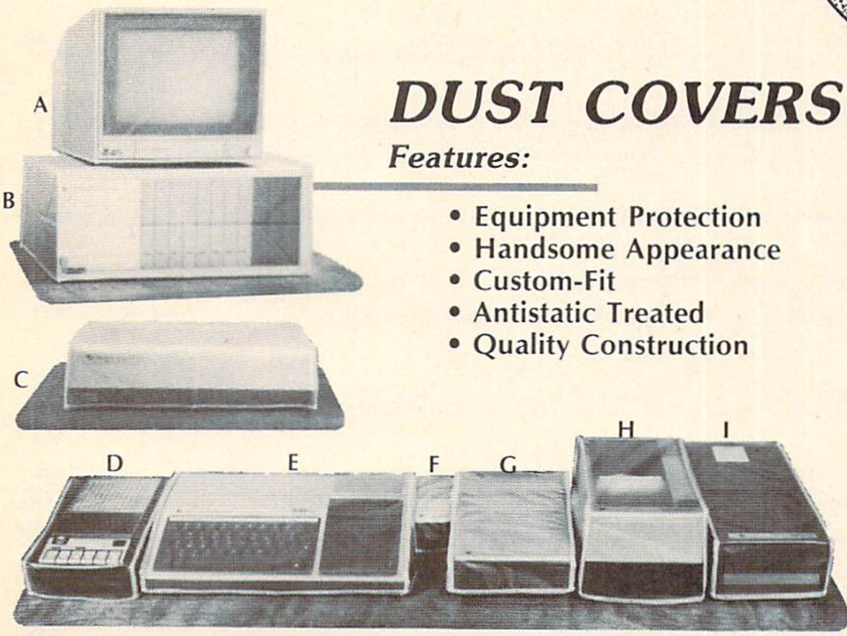
The TEX-SETTE™ Adapter

For Those Cassette Recorders Whose Remote Jack Wiring is Reversed. Cassette Compatibility At Last!

If the TI-99/4A will not control your cassette recorder's motor (automatic start/stop), we have the solution for you.



Only $5.95



DUST COVERS

Features:

- Equipment Protection
- Handsome Appearance
- Custom-Fit
- Antistatic Treated
- Quality Construction

A	10" Color Monitor Cover	$10.95
	13" Color Monitor Cover (now shown)	$12.95
B	Peripheral Expansion Box Cover	$12.95
C	TI's 99/4 Matrix Printer Cover (same as Epson MX-80 Cover)	$ 9.95
D	Cassette Recorder Cover (1 size fits up to 10" x 6")	$ 4.95
E	TI-99/4(A) Console Cover	$ 8.95
F	Speech Synthesizer Cover	$ 3.95
G	Peripheral Box Cover (Specify: 32K Memory Expansion RS232 Interface, or Disk Controller)	$ 5.95
H	Thermal Printer Cover	$ 8.95
I	Disk Memory Drive Cover	$ 5.95

Dealer Inquiries Invited

► THIS ORDER FORM MAY ALSO BE USED FOR: ◄
The Best of 99'er **(p.163),** ***99'er Directory*** **(p.166), and Simon's Saucer (p.148)**

NAME ______________________ PLEASE
ADDRESS ____________________ PRINT
CITY ______________ STATE _____ ZIP ________

Check enclosed ☐ MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

Bill my: ☐ VISA ☐ Master Card ► ________ Expiration Date

Account No.

Tel. No. ________ Signature ________

QTY	ITEM(S)	PRICE	SHIPPING	AMOUNT
	Best of 99'er	$19.95	$1.50 ea.☆	
	Best of 99'er On Tape	$35.00	$1.00 ea.☆	
	Book and Tape Package	$49.95	$2.50 ea.☆	
	Finder Binder (no tapes)	$11.95	$1.50 ea.☆	
	Finder – Binder w/6 blank tapes	$16.95	$1.50 ea.☆	
	Finder-Binder w/12 blank tapes	$21.95	$2.50 ea.☆	
	12 Blank Tapes	$14.00	$2.00 ea.	
	Texsette Adapter	$5.95 ea.	.50 ea.	
	Simon's Saucer	$9.95	$2.00 ea.☆	
			Subtotal	

Qty	ITEM(S)	PRICE	SHIPPING	AMOUNT
	Back Issues (Circle Issues Desired) Volume 1 No. 6, Nov '82, Dec '82, Jan '83, Feb '83, Mar '83, Apr '83, May '83, Jun '83, Jul '83, Aug '83, Sept '83, Oct '83, Nov '83 (Handling incl.)	$3.95 ea.	Foreign Surface add $2.00 ea. Foreign Air add $4.00 ea.	
	Dust Covers A B C D E F G H I		.50¢ ea. cover	
	99'er Directory (Handling incl.)	$5.00		

☆ Foreign surface add $3.00 for shipping.

Foreign Air Inquire

Subtotal ______
Total Order ______
Handling $1.50
TOTAL PAYMENT ENCLOSED ______

Enclose check, money order or credit card information to:
99'er ware, Dept M1
P.O. Box 5537
Eugene, OR 97405
or
Use VISA/MasterCard and Order Toll-Free
1-800-828-2212
In Oregon, Alaska, and Hawaii telephone
(503) 485-8796

Please allow 10 days for your order to be processed.

see page 165 to order

Addendum to *99'er Directory of Commercially Available Software, Peripherals, and Accessories*, Summer/Fall 1983 Edition

Home Management/Personal Finance

Program Title	Mfr. No.	Medium	Req'd Accs.	Opt. Accs.	Description	Sugg. Retail
Electricity Manager	FU020	Dsk	DD&C, X-BASIC, 32KMExp		Monitor/budget energy	34.95
Household Inventory	FU020	Dsk	DD&C, X-BASIC		Record home assets	34.95
Med Alert!	FU020	Dsk	DD&C, X-BASIC		Accident prevention	34.95
Amortization Schedule	FU020	Cass/Dsk	CR&C/DD&C, X-BASIC, RS232		Calculate mortgage	49.95/54.95

Business/Professional

Program Title	Mfr. No.	Medium	Req'd Accs.	Opt. Accs.	Description	Sugg. Retail
Accounts Payable	FU020	Dsk	DD(2)&C, X-BASIC, RS232, Printer, 32KMExp		Payable functions	149.95
Accounts Receivable	FU020	Dsk	DD(2)&C, X-BASIC, RS232, Printer, 32KMExp		Receivable functions	149.95
Billing	FU020	Dsk	DD(2)&C, X-BASIC, RS232, Printer, 32KMExp		Billing functions	149.95
General Ledger	FU020	Dsk	DD&C, X-BASIC, RS232, Printer	2nd DD&C	General ledger functions	149.95
Inventory Management	FU020	Dsk	DD(2)&C, X-BASIC, RS232, Printer, 32KMExp		Maintain inventory	149.95
Mailing List	FU020	Dsk	DD(2)&C, X-BASIC, RS232, Printer, 32KMExp		Versatile list	99.95
Payroll	FU020	Dsk	DD(2)&C, X-BASIC, RS232, Printer, 32KMExp		Print payroll checks	149.95
Trendline	IM010	Cass/Dsk	CR&C/DD&C	X-BASIC	Advanced stock market program for the 99/4A	27.95

Early Education

Program Title	Mfr. No.	Medium	Req'd Accs.	Opt. Accs.	Description	Sugg. Retail
Tell Me A Story	FU020	Cass	CR&C, X-BASIC		Encourage child's reading interests	19.95
Skill Builder I	IM010	Cass	CR&C, Joysticks		Two skill games: Bingo Duel and Number Hunt	19.95

Continuing Education

Program Title	Mfr. No.	Medium	Req'd Accs.	Opt. Accs.	Description	Sugg. Retail
Wall Street Challenge	IM010	Cass	CR&C		Computer simulation of the stock exchange	19.95
Wildcatting	IM010	Cass	CR&C		Simulate a hidden oil deposit, which you try to find	19.95
Learning Morse Code	IM010	Cass/Dsk	CR&C/DD&C		Teaches the user the correct sound of each character in easy steps	24.95

Teaching and Administrative Aids

Program Title	Mfr. No.	Medium	Req'd Accs.	Opt. Accs.	Description	Sugg. Retail
Galactic Spelling	IM010	Cass/Dsk	CR&C/DD&C, TE-II, SpSyn		Self-teaching aid, tutor, game, and quiz	19.95

Word Processing

Program Title	Mfr. No.	Medium	Req'd Accs.	Opt. Accs.	Description	Sugg. Retail
Word Processor	FU020	Dsk	DD&C, X-BASIC, RS232, Printer, 32KMExp	2nd DD&C	Create, maintain, print	149.95
Direct Writer II Word Processor	IM010	Dsk	DD&C, X-BASIC, 32KMExp, RS232, Printer		32 to 132-character line length	69.95

Programming Aids

Program Title	Mfr. No.	Medium	Req'd Accs.	Opt. Accs.	Description	Sugg. Retail
Compactor Plus	IM010	Dsk	DD&C, X-BASIC, RS232, Printer		Programmer tool—compact, analyze, delete, resequence	29.95

Entertainment

Program Title	Mfr. No.	Medium	Req'd Accs.	Opt. Accs.	Description	Sugg. Retail
Air Missile Command	FU020	Cass	CR&C, X-BASIC		Protect city from aliens	19.95
All *Star Baseball	FU020	Cass	CR&C, X-BASIC		Two-player game	19.95
All *Star Bowling	FU020	Cass	CR&C, X-BASIC		1 to 8 players	19.95
Battleground	FU020	Cass	CR&C, X-BASIC		Calculate enemy's position and destroy	19.95
Casino Pack	FU020	Cass/Dsk	CR&C/DD&C		Blackjack/Craps/Slot	19.95/21.95
Challenge I	FU020	Cass	CR&C		Leap-frogging	16.95
Challenge II	FU020	Cass	CR&C		Nim/Tic-Tac-Toe	16.95
'Chutes & Sharks	FU020	Cass	CR&C, X-BASIC, 32KMExp		Parachutist's fall	19.95
CIA Adventure	FU020	Cass/Dsk	CR&C/DD&C, X-BASIC, 32KMExp		Find missing ruby	19.95/21.95
'Copter Capers	FU020	Cass	CR&C, X-BASIC		Rescue from burning building	19.95
Cosmic Drift	FU020	Cass	CR&C, X-BASIC		Action game	19.95
Death Station	FU020	Cass	CR&C, X-BASIC		Try to break death spell	19.95
Doomcastle	FU020	Cass	CR&C, X-BASIC		Recover sapphires	19.95
Dr. Nuttier	FU020	Dsk	DD&C	SpSyn, TE II	Mind-blowing answers	21.95
Dragon Game	FU020	Dsk	DD&C, X-BASIC, 32KMExp, EdAss, Joysticks		Action game	24.95
Fly Snuffer	FU020	Cass	CR&C, X-BASIC, Joysticks		Snuff out flies	19.95
Flying Saucer	FU020	Cass	CR&C, X-BASIC		Action game	19.95
Fowl Play	FU020	Cass	CR&C, X-BASIC		Help hen across road	19.95
Galactic War	FU020	Cass	CR&C, X-BASIC		Battle the alien craft	19.95
Getcha!	FU020	Cass	CR&C, X-BASIC, Joysticks		Action game	19.95
Hopper	FU020	Cass	CR&C, X-BASIC		Jump frogs across river	19.95
The Hustler	FU020	Cass	CR&C, X-BASIC, Joysticks		Eight ball/rotation	19.95
Ice Caverns of Xen	FU020	Cass/Dsk	CR&C/DD&C, X-BASIC, 32KMExp		Reasoning ability	19.95/21.95
Laser Battle	FU020	Cass	CR&C, X-BASIC, Joysticks		Action game	19.95
London Blitz	FU020	Cass	CR&C, X-BASIC, Joysticks		Shoot down bombers	19.95
Mars Marauder	FU020	Cass	CR&C, X-BASIC, Joysticks		Navigation skills	19.95
Mars Rover	FU020	Cass	CR&C, X-BASIC		Action game	19.95
Memory Match	FU020	Cass	CR&C, X-BASIC		Make most matches and win	19.95
Minefield	FU020	Cass	CR&C, X-BASIC		Step on mine & explode	19.95
Moonbus	FU020	Cass	CR&C, X-BASIC		Defend pyramid, shoot aliens	19.95
Moonduster	FU020	Cass	CR&C		Try landing on moon	19.95
Monster Craze	FU020	Cass	CR&C, X-BASIC		Action game	19.95
Newton's Revenge	FU020	Cass/Dsk	CR&C/DD&C, MiniMem	Joysticks	Catch falling apples	24.95/24.95
Newton's Revenge	FU020	Dsk	DD&C, X-BASIC, 32KMExp/EdAss	Joysticks	Catch falling apples	24.95
Sam Defense II	FU020	Cass	CR&C		Three difficulty levels	19.95
Starship Concord	FU020	Cass	CR&C, X-BASIC		Real-live action game	19.95
Starship Concord	FU020	Cass	CR&C, X-BASIC, 32KMExp		Real-live action game	19.95
Viral Vendetta	FU020	Dsk	DD&C, X-BASIC, 32KMExp/EdAss		Protect cells from virus	24.95
Wall Street	FU020	Cass	CR&C		Realistic simulation	19.95
Wall Street	FU020	Cass	CR&C, X-BASIC		Realistic simulation	19.95
Word/Numberscope	FU020	Cass	CR&C, X-BASIC		Enhance speed reading	19.95
Tournament Brick Bat	IM010	Cass	CR&C, Joystick		Action-skill game	14.95
Strategy Pack I	IM010	Cass	CR&C	Joysticks	Two strategy games: Roman Checkers and Frame Up	19.95
Mind Master	IM010	Cass	CR&C		Strategy game in which computer designs hidden problems & reports the results of each guess	14.95
Demon of the Planes	IM010	Cass/Dsk	CR&C/DD&C, X-BASIC		Graphic adventure into the astral plane	19.95
Mr. Mouse	IM010	Cass/Dsk	CR&C/DD&C, X-BASIC	Joysticks, SpSyn	Talking game in X-BASIC	19.95
Asteroid Destroyer	IM010	Cass/Dsk	CR&C/DD&C, X-BASIC	Joysticks	Arcade game	19.95
Shuttle Command	IM010	Cass/Dsk	CR&C/DD&C, X-BASIC	Joysticks	Destroy the Russian Attack Military Satellites	19.95
Quimbee	IM010	Cass/Dsk	CR&C/DD&C, X-BASIC		Game of skill and chance	19.95

CC = Command Cartridge, CR&C = Cassette Recorder & Cable, DD&C = Disk Drive & Controller

Due to a misinterpretation of information from Wycove Systems, Ltd., their three Wycove Forth programming aids were listed in the September *99'er Directory* as N/A. The programs were and are available.

The telephone numbers listed for C. A. Root Associates in the September *99'er Directory* should be disregarded. To contact C. A. Root Associates, call (206) 941-6984.

Company	Mfr. No.
FUTURA Software, P. O. Box 5581, Fort Worth, TX 76108, (817) 732-1687	FU020
Image Computer Products, Inc., P. O. Box 3761, Cherry Hill, NJ 08034, (609) 667-2526	IM010

Future Is Now. . . from p. 147

Garde's demo disk provides a step-by-step tutorial to help you adjust and align the sounds and graphics to your monitor and speaker.

There are four female connector jacks on the SuperSprite board. Each requires a coaxial cable with a male plug at each end. One of those jacks is an optional connector to a sound amplifier—a feature you may not want to add immediately. The audio amplifier on the board is relatively large, and the sounds are already quite loud.

A second optional connector jack lets you hook up an additional CRT monitor to the system for use when you have the board in slot 7 but are not doing graphics. This is not necessary, but it is handy so that the signal does not need to go through the board on the way to the monitor. If you are not using the *SuperSprite* board, but it is in slot 7, you may get a slight decrease in signal. To alleviate this, you simply adjust the monitor's brightness and/or contrast, or hook up a second monitor using this jack.

The last two jacks are not optional. The video input of the board must be connected via cable to the video output jack on the back of the Apple. The board's video output should then be connected to a monitor or TV to display either the Apple 6502 or the *SuperSprite* graphics or both.

Who Needs It?

Programmers who have been struggling to create commercial products for the Apple will welcome *SuperSprite* and *StarSprite*. Not only will their task be easier, but it will give their programs an edge in a highly competitive market.

Several features of the system are notable from the programmer's standpoint. (See the accompanying interview with Don Fudge in this issue.) The Ampersprite language was developed to interface with the board so that most programming can be done from Applesoft. Assembly language programming is also simplified by the board, but animation is much faster with Ampersprite.

It is possible to compare standard Apple techniques with Ampersprite in an animation programming task. Let's take a simple task, such as moving a white, 32 × 32 object from left to right across the screen, and compare the ease of programming and the quality of animation.

An Ampersprite program which accomplishes this task looks like this:

FOR B=0 TO 209: &AX6,B: NEXT

Notice that the Ampersprite command is simply embedded in one of those old, familiar Applesoft FOR–NEXT loops. The animation that results will move across the screen in 0.77 seconds. The quality of the graphics is excellent.

We can achieve the same results using a routine with standard Apple techniques. One with good graphics results is a BASIC-driven assembly routine. It requires 129 programming steps. The animation flickers around the edges, and the object moves across the screen in 5 seconds. (The results of a comparison of several techniques appear in the Performance Chart.)

The software system available for the *SuperSprite* goes beyond the *StarSprite I* included with the board. *StarSprite II* provides additional programming utilities, programming hints, and sample games that demonstrate some of the programming techniques available. It also includes the *Paint Master Scene Utility*. *StarSprite II* is for intermediate level programmers and interested novices. The instruction manual included with the disk is easy for any Apple user to follow but doesn't sacrifice the finer details. *StarSprite II* sells for $79.95.

StarSprite III is for very serious programmers, and also costs $79.95. It includes source codes for the machine language routines in *Starsprites I* and *II* and *Paint Master*. And it gives advice on assembly for anyone wanting to get involved in assembly programming.

The Classroom Connection

Apples are the most popular computers used in schools. Educators can now obtain bright, natural-looking and -sounding learning tools to enliven the creative experience for youngsters.

Parents who have taken education into their own hands via the home computer now have access to the kind of visual and audio impact they need to open new worlds to their children. These creative opportunities are not limited to experienced or professional programmers. With the utilities and instructions included in the system, parents can easily create learning adventures for their children. The colors, the sounds, and the ease of adding text to graphics for instructional purposes all make this system vastly superior to those available for the Apple in the past. The utilities and instructions in the *StarSprite* software system will make it possible for you to concentrate on *creating* rather than on the *tool* you create with.

Continued on p. 193

CALL 2116
(Machine language subroutine CALLed from BASIC)

```
0840-  00 00 00 00 A0 09 A6 07
0848-  CA E0 00 F0 04 C8 4C 48
0850-  08 98 85 FB A9 00 85 FA
0858-  A5 FD 85 06 A2 00 A0 00
0860-  20 11 F4 A4 FE A2 00 A1
0868-  FA 51 26 91 26 88 18 E6
0870-  FA D0 02 E6 FB C0 FF F0
0878-  04 C4 FF B0 EA C6 06 A5
0880-  06 C9 FF F0 04 C5 FC B0
0888-  D3 60 00 00 00 00 00 00
```

```
DEMO 1
10 FOR A = O TO 209: XDRAW 1 AT A,Y
20 XDRAW 1 AT A,Y: NEXT

DEMO 2
10 FOR A = 1 TO 209
20 POKE 230,64: POKE -16300,0: XDRAW 1 AT A - 1,Y: XDRAW 1 AT A + 1,Y
30 POKE 230,32: POKE -16299,0: XDRAW 1 AT A,Y: XDRAW 1 AT A + 2,Y
40 NEXT

DEMO 3
10 FOR HL = 0 TO 29: HR = HL + 5: POKE 255,HL: POKE 254,HR
20 S = S + 1: IF S > 7 THEN S = 1: NEXT : GOTO 40
30 POKE 7,S: CALL 2116: GOTO 20
40 REM

DEMO 4
10 FOR HL = 0 TO 29: POKE 255,HL: HR = HL + 5: POKE 254,HR
20 SW = NOT SW: IF SW = 0 THEN POKE 230,64: POKE -16300,0: GOTO 40
30 POKE 230,32: POKE -16299,0
40 POKE 7,S: CALL 2116: POKE 7,S + 2: IF S + 2 > 7 THEN POKE 7,(S + 2) - 7
45 IF S > 5 THEN POKE 255,HL + 1: POKE 254,HR + 1: CALL 2116: POKE 255,HL: POKE 254,HR: GOTO 60
50 CALL 2116
60 S = S + 1: IF S > 7 THEN S = 1: GOTO 100
80 GOTO 20
100 NEXT
```

HCM

Performance Chart
Results obtained when moving a white 32 x 32 block from X = 0 to X = 209 on the screen.

Program Routine	DEMO #1	DEMO #2	DEMO #3	DEMO #4	Hi-Res Secrets, 28A, option (H), two-page flipping vector shape DEMO in assembly	Hi-Res Secrets, 28A, option (D), two-page flipping block shape DEMO in assembly with inputs	FOR B = 0 TO 209: &AX6, B: NEXT (Ampersprite command in an Applesoft FOR—NEXT)
Type of Shape	vector	vector	block	block	vector	block	sprite
Drawing Method Used	XDRAW from BASIC	XDRAW from BASIC	EOR in CALLed block shape routine (called from BASIC)	EOR in CALLed block shape routine (called from BASIC	XDRAW routine (Applesoft's) from mach. lang.	block shape drawing routine from mach. lang.	simple coordinate update from Ampersprite
Quality of Animation	terrible	good, but very slow	terrible speed ok	good, but slow	good, but slow	slightly unsteady around the edges speed ok	excellent animation quality, speed
Screen Flipping	no	yes	no	yes	yes	yes	not used for sprites
Time in Seconds	67	70	8	18	63	5	0.77

Auto Sprite Motion . . . from p. 162

```
100 OPEN 3,1,1,"DATA"
110 FOR L=50880 TO 51116
120 READ A
130 PRINT #3,A:NEXT
140 CLOSE 3
150 Begin data statements here.
```

If you are using disk, change line 100 to:

```
100 OPEN 3,<device#>,3,"@0:DATA,S,W"
```

To read a sequential file on tape use these lines:

```
100 OPEN 3,1,0,"DATA"
110 FOR L=50880 TO 51116
120 INPUT#3,A
130 POKE L,A:NEXT
140 CLOSE 3
```

For disk, change line 100 to:

```
100 OPEN 3, <device#>,3,"DATA,S,R"
```

Using Auto Sprite Motion

Table 1 shows the corresponding addresses (in decimal and hexadecimal code) for each sprite motion byte. The byte at address 50432 ($C500) enables auto motion for all eight sprites. Its format is exactly like the sprite enable register in the Commodore 64 (53269); that is, when bit 0 equals 1, then sprites 0's auto motion is enabled. When bit 0 equals 0, then sprites 0's auto motion is disabled. (Bit 1 corresponds to sprite 1, and so on.)

TABLE 1

ADDRESS (HEX.)	ADDRESS (DEC.)	DEFINITION
C500	50432	ENABLE SPRITE MOTION BIT 0-7 = SPRITE 0-7
C501	50433	SPRITE 0; X MOTION
C502	50434	SPRITE 0; Y MOTION
C503	50435	SPRITE 1; X MOTION
C504	50436	SPRITE 1; Y MOTION
C505	50437	SPRITE 2; X MOTION
C506	50438	SPRITE 2; Y MOTION
C507	50439	SPRITE 3; X MOTION
C508	50440	SPRITE 3; Y MOTION
C509	50441	SPRITE 4; X MOTION
C50A	50442	SPRITE 4; Y MOTION
C50B	50443	SPRITE 5; X MOTION
C50C	50444	SPRITE 5; Y MOTION
C50D	50445	SPRITE 6; X MOTION
C50E	50446	SPRITE 6; Y MOTION
C50F	50447	SPRITE 7; X MOTION
C510	50448	SPRITE 7; Y MOTION
C600	50688	NEW MOTION FLAG

The X and Y motion bytes for each sprite determine the direction and speed of auto motion. The X motion byte governs horizontal motion, and the Y motion byte governs vertical motion. Bit 7 controls direction, and bits 0 through 6 control speed. If bit 7 in an X byte is *set* (equal to 1), the sprite will move to the left; and if it is *cleared* (equal to 0), the sprite will move to the right. If bit 7 in a Y byte is set, the sprite will move up; and if it is cleared, the sprite will move down.

"Wouldn't it be nice if you could just start your sprites moving at a specified speed and direction and not have to bother with all those POKEs?"

The speed of the sprite is dependent on the values you POKE into bits 0 through 6, *and* upon the value of bit 7. If bit 7 is set to 1, then the fastest motion will be achieved by setting all bits 0 through 6 to 1. Therefore, you attain the fastest auto sprite motion for sprite 0 in the left and up directions by POKEing 50433 and 50434 with 255. If bit 7 is cleared, then the fastest motion will be achieved by setting bit 0 to 1 and clearing bits 2 through 7. So you get the fastest auto sprite motion for sprite 0 in the right and down directions by POKEing locations 50433 and 50434 with 1. Generally, a value of 128 or greater for the motion byte will move the sprite up or to the left, depending on whether it is an X or Y byte. If the value is less than 128, the sprite will move down or to the right.

Along with the motion bytes listed above, the Auto Sprite Motion subroutine uses *count bytes.* These bytes are initially loaded with values from the motion bytes. Then once every sixtieth of a second, the routine increments or decrements these bytes, depending on the state of bit 7 in the motion bytes. If the value of bit 7 is 1, then the count bytes are incremented, and if bit 7 is a zero, then the count bytes are decremented. When the count byte reaches zero, the corresponding X or Y sprite position register in the VIC (video interface chip) is incremented or decremented (again depending on the state of bit 7 of the motion bytes), causing the sprite to move on the screen. The count byte is then reinitialized to the value of its corresponding motion byte. You can find the address of the corresponding count byte by adding 256

($100) to any motion byte. The programmer need not be concerned with the values of these bytes at any given time because they are maintained by the Auto Sprite subroutine.

The Auto Sprite Motion routine also uses a *new motion flag* at address 50688 ($C600). We use this flag to tell the Auto Sprite subroutine that a new motion value has been placed in a motion byte. If this flag is any value other than zero, the Auto Sprite subroutine will change all count bytes to the values of their corresponding motion bytes. The subroutine then clears this flag to zero.

The BASIC Program Listing

This program READs the Auto Sprite and Interrupt Vector subroutines as DATA statements and then POKEs the machine code into the proper addresses. The program then goes on to position the eight sprites across the screen at arbitrary locations. Next it turns on all sprites for display and motion, with different speeds for each sprite. From then on it's automatically smooth and speedy sailing for your spritely characters.

BASIC PROGRAM FOR AUTOSPRITE MOTION
Explanation of The Program

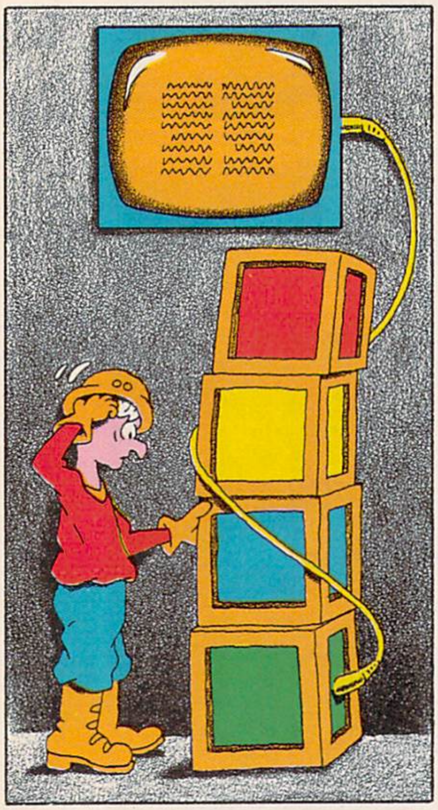
Line nos.	
100-170	Header
180	READ machine language routines and POKE into proper locations.
190-330	Auto Sprite subroutine.
340	Change Interrupt Vector subroutine.
350	Initialize S to the beginning of Motion bytes.
360	H and V are used as motion variables for the X and Y motion bytes of the sprites. H will be poked into X motion bytes, and V will be poked into Y motion bytes. Both will be changed to give different values in the upcoming FOR-NEXT loop.
370-390	Poke X and Y motion bytes with arbitrary values.
400	Enables New Motion Flag.
410-420	Assigns all eight sprite definitions to the same pattern description (14).
430-440	Loads the sprite pattern description of the solid sprite block into addresses 896 through 959.
450-470	POKEs sprite color registers.
490-520	Loads the initial sprite position values into the X and Y sprite position registers. The sprites are placed at arbitrary positions on the screen.
530	255 is POKEd into the sprite display enable register of the VIC to turn all sprites on.
550	Enables Sprite motion.
560	Branches to the Change Interrupt Vector subroutine.

COMMODORE 64

```
100 REM *********************
110 REM *    BASIC PROGRAM    *
120 REM * AUTOSPRITE TUTORIAL *
130 REM *********************
140 REM BY JOHN THRASHER
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION  4.1.1
170 REM C64 BASIC
180 FOR L=50880 TO 51116:READ A:POKE L,
    A:NEXT
190 DATA 169,255,45,0,198,240,16,169,0,
    141,0,198,162,21,189,0
200 DATA 197,157,0,198,202,208,247,162,
    1,169,1,141,80,197,173,80
210 DATA 197,45,0,197,240,3,76,243,198,
    232,232,14,80,197,208,238
220 DATA 76,49,234
230 DATA 169,0,29,0,197,208,3,76,97,199
    ,169,128,61
240 DATA 0,197,240,48,254,0,198,208,40,
    222,255,207,76,144,199,80
250 DATA 197,45,16,208,208,12,173,16,20
    8,13,80,197,141,16,208,76
260 DATA 43,199,173,16,208,77,80,197,14
    1,16,208,189,0,197,157,0
270 DATA 198,76,97,199,222,0,198,208,40
    ,254,255,207,208,29,173,80
280 DATA 197,45,16,208,208,12,173,16,20
    8,13,80,197,141,16,208,76
290 DATA 91,199,173,16,208,77,80,197,14
    1,16,208,189,0,197,157,0
300 DATA 198,169,0,232,29,0,197,208,3,7
    6,140,199,169,128,61,0
310 DATA 197,240,11,254,0,198,208,20,22
    2,255,207,76,134,199,222,0
320 DATA 198,208,9,254,255,207,189,0,19
    7,157,0,198,202,76,233,198
330 DATA 169,255,221,255,207,240,3,76,4
    3,199,173,80,197,76,17,199
340 DATA 120,169,192,141,20,3,169,198,1
    41,21,3,88,96
350 S=50433:PRINT "[SHIFT CLR]"
360 H=247:V=1
370 FOR L=S TO S+16 STEP2
380 POKE L,H:POKE L+1,V
390 H=H+1:V=V+1:NEXT
400 POKE 50688,FF
410 FOR L=2040 TO 2047
420 POKE L,14:NEXT
430 FOR L=896 TO 959
440 POKE L,255:NEXT
450 S=1
460 FOR L=53287 TO 53294
470 POKE L,S:S=S+1:NEXT
480 POKE 53281,0
490 S=5:A=10
500 FOR L=53248 TO 53263 STEP 2
510 POKE L,S+48:POKE L+1,A+40
520 S=S+25:A=A+25:NEXT
530 POKE 53269,255
540 REM
550 POKE 50432,255
560 SYS 51104
570 GOTO  570
580 END
```

HCM

sor made especially for the home.



A Trip Down Memory Lane

Moving The C-64 Screen Map

With 38911 bytes reserved for BASIC programs on the Commodore 64, you wouldn't expect to run out of memory when programming in BASIC. But when you start designing graphics you may come up short. Why does this happen, and what can you do about it? In this article we will investigate the causes of this common problem, and show you an ingenious way to get around it.

The first stop in our investigation is the *Commodore 64 Programmer's Reference Guide*, which offers an explanation of programmable characters. The instructions on page 109 show us how we can keep BASIC from writing over our character set by poking the value 48 into locations 52 and 56. If we follow this suggestion to the letter, our reserved BASIC memory is decreased by over two-thirds—a severe constraint on the size of our BASIC programs.

How does this one POKE limit the memory so much? To understand, we must see how the Commodore operating system normally allocates memory, and how it uses the poked values to keep your BASIC program from writing over your character definitions.

Current Address: 2048 BASIC

The starting address for BASIC memory is generally fixed at address 2048. Addresses 52 and 56 tell the operating system the "page" number where the BASIC program area ends. The memory is divided into 256 "pages," each containing 256 bytes. We can find the end of reserved BASIC memory by multiplying the value of the byte in address 52 or 56 by 256. When the Commodore 64 is turned on, the operating system loads a value of 160 into addresses 52 and 56, which sets the end of reserved BASIC memory equal to address 40960 ($256 \times 160 = 40960$). If you subtract 2048 from 40960, you will get 38,912. This is actually one more than the number of bytes displayed on the screen because 40959 is the last address that can be used by your BASIC program. Normal reserved BASIC memory, with no programmable characters, is represented by the section labeled A in Figure 1.

Now suppose you follow the instructions in the *Programmer's Reference Guide* and POKE the value 48 into locations 52 and 56. This sets the end of programmable memory at address 12288. Subtract the fixed starting address, 2048, and you get $12288 - 2048$, or 10240 bytes. Unfortunately, you have just decreased reserved memory size by over two-thirds. This is illustrated in section C of Figure 1.

Now that you understand the source of the problem, let's try to find a way around it. When you POKE 48 into locations 52 and 56 as described above, you are loading your character definitions beginning at address 12288. This memory location just happens to be right in the middle of the RAM memory normally reserved for BASIC programs. How can we program our own characters and still have a large amount of BASIC memory to work with? The trick is to place our programmed character set in a memory area outside the one normally reserved for BASIC programs. To do this, we relocate the memory area that is used by the VIC II chip (the IC chip that displays the picture on the TV screen).

To understand how to go about this, you must be acquainted with the Commodore 64 memory map and with how the VIC II accesses memory. Commodore breaks up its 64K of RAM memory into four equal blocks of 16K bytes each. At any one time the 6510 microprocessor can address all four of these blocks, but the VIC II can address only one 16K block. This is because the VIC II chip is designed with only enough address lines to access 16K. And, in fact, this is all it needs to fully utilize all its capabilities. With a few clever PEEKs and POKEs, you can control which 16K block it is able to access. This is the basis of our strategy for reclaiming memory space for BASIC programming.

Before we start rearranging things, let's see how the VIC II chip knows how to find the information it needs to display our characters and sprites on the screen. There are three addresses that tell the VIC II where to get the information for the display screen, sprites, and

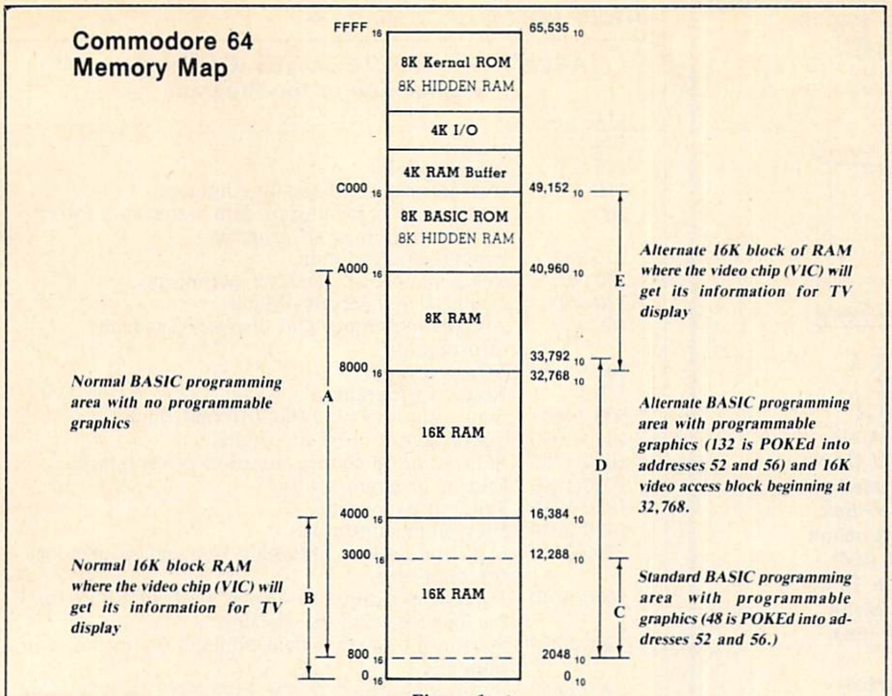


Figure 1

characters. These addresses are offsets relative to the 16K block in which the VIC II is operating. To the programmer the three addresses are absolute.

The first address is the beginning of screen memory. This address can be any one of sixteen 1K blocks of memory inside the 16K of RAM that the VIC II is accessing. The VIC II knows which one it is by the value of the upper 4 bits of location 53272.

We can use the following statement to change the location of screen memory:

POKE 53272, (PEEK(53272)AND15) ORA

where A is one of the following values:

A	location
0	0
16	1024
32	2048
48	3072
64	4096
80	5120
96	6144
112	7168
128	8192
144	9216
160	19240
176	11264
192	12288
208	13312
224	14336
240	15360

The AND15 keeps the lower 4 bits intact while clearing (making zero) the upper 4 bits. The ORA then changes the upper 4 bits to the value we want in that location. Remember that this address is an offset from the beginning of the 16K block of RAM that the VIC II is accessing. For our example we will set A equal to 16, giving us an offset of 1024.

The second address is that of the sprite pointers. These pointers are the last eight bytes of the 1K chunk of screen memory. Normally this means they begin at address 2040, but if screen memory is moved, these will move too.

The third address is that of character memory. This can be any one of the eight 2K blocks of memory the VIC II is accessing. The VIC II knows which one it is by the value of the lower 4 bits of address 53272. The following statement changes the pointer to the beginning of character memory:

POKE 532762,(PEEK(53272)AND 240)ORB

where B is one of the following values:

B	location
0	0
2	2048
4	4096
6	6144
8	8192
10	10240
12	12288
14	14336

The AND240 keeps the upper 4 bits intact, and the ORB changes the lower 4 bits.

When we want to program our own graphics, we should establish our character definition address at an offset different from the normal ones of 4096 and 6144. This is because the 4K character generator ROM also resides in this same 16K block of RAM. We can put our character set in any of the four 2K blocks available above these addresses. For our example we will set B equal to 12, giving us an offset of 12288 to the beginning of our character definitions.

Moving VIC II Memory

The 16K block of RAM that the VIC II normally uses is depicted by section B of Figure 1. Section E is the 16K block that we will assign to the VIC II in order to leave the reserved BASIC memory relatively intact. Section D is the reserved BASIC memory that will be available after we make the change in VIC II memory.

Moving VIC II memory is simple, and requires only the following two BASIC statements:

```
100 POKE 56578,PEEK(56578)OR3
110 POKE 56576,(PEEK(56576)
    AND252)ORC
```

Line 100 is preparatory. It programs a port in the Complex Interface Adapter chip (CIA #2) which will enable the hardware to execute line 110. Line 110 selects one of the four 16K blocks of memory for the VIC II as determined by the value of C. We select a value for C from the following table:

C	location
0	49152
1	32768
2	16384
3	0

How do we choose the best possible value for C in line 110? We can eliminate addresses 0 and 16384 immediately because using these areas would cut into BASIC programming space. Figure 1 shows us that the remaining two banks (32768 and 49152) have 8K blocks of ROM memory within the 16K blocks they encompass. These ROMs need to be considered only if the programmer is planning on reading the contents of the "hidden" RAM that is underneath. It is possible to put all the memory the VIC II needs in one of the 8K blocks that is not covered by a ROM. But as far as the VIC II is concerned, the ROMs are not even there because the VIC II's address lines are not even connected to these ROMs.

With this in mind, all we have left to consider is the contents of each block of RAM at any given time. In the 4th 16K block (starting at 49152) we have color memory and also a block of 4K RAM that is commonly used for assembly language programs (addresses 40960 to 45055). Since we may want to use machine language programs in conjunction with these techniques, we will leave this area available for that purpose.

On the other hand, the 3rd 16K block (starting at 32768) contains only the last 8K of the reserved BASIC programming area and a 4K character ROM. If we eliminate only this 8K block, we will still have plenty of room for our BASIC programs. With a standard Commodore system (including a monitor and disk drive or tape deck) this is probably the wisest choice. Therefore, we will make C in line 110 equal to 1. Once we have made this choice, we can let the operating system know our plans by using the following two statements:

```
120 POKE 648,132
130 POKE 52,132:POKE 56,132
```

Line 120 tells the Commodore 64's operating system where screen memory is located and thereby enables us to use PRINT and INPUT statements in our program. The value 132 POKEd into address 648 is a "page" used to reset the pointer to the absolute address of the upper left-hand corner of screen memory. When the Commodore is turned on, the value

Continued on p. 176

Uncle Larry . . . **from p. 15**

LARRY'S TEN FIDDLE TUNES (Commodore 64)
Explanation of the Program

Line nos.	
100-170	Header.
180-200	Initialization.
210-250	Enables restore of any line number.
260	Restores to beginning of data statements for entry into beginning of program.
270-280	Initializes sound chip.
290-310	Reads in special character definitions.
320-450	Displays first screen—Menu.
460-490	Accepts user input and displays pertinent information.
500	Selects a tune.
510	Restarts program.
600-1360	Subroutines to play the different tunes.
1410-1650	Subroutine to tune the fiddle.
1660-1720	Subroutine to change duration of each note.
1730-1840	End of program routine.
1850-1910	Print string routine.
1920-2020	Play Note subroutine.
2290-3310	Data statements containing frequencies of notes for each tune.
5000-5040	Subroutines called to display neck of fiddle for the Tune the Fiddle selection.
8200-8270	Subroutine to store data pointers or restore data pointers.
8600-8660	Subroutine called by the Play Note subroutine to play a note.
9000-9040	Graphics initialization subroutine called at the beginning of the program.

COMMODORE 64

```
100 REM    ****************
110 REM    *  LARRY'S TEN   *
120 REM    *  FIDDLE TUNES  *
130 REM    ****************
140 REM    BY LARRY SCHOTT AND
                 THE HCM STAFF
150 REM    HOME COMPUTER MAGAZINE
160 REM    VERSION 4.1.1
170 REM    C64 BASIC
180 PRINT "{SHIFT CLR}"
190 GOSUB 9000
200 T=0:D=0:SD=125:R=0
210 DIM D1(25),D2(25),D3(25),D4(25),A$(
    14)
220 FOR L=1TO24:READA:NEXT
230 FOR L=1 TO 28:READA$:NEXT
240 READ A:IF A=100 THEN GOSUB 8200
250 IF A<>-333 GOTO 240
260 RESTORE
270 G=54272
280 FOR L=0TO24:POKEG+L,0:NEXT
290 FOR I=45568 TO 45568+23:READA:POKEI
    ,A:NEXT
300 DATA 6,12,12,12,124,252,248,112,231
    ,231,231,231,231,231,231,231
310 DATA 255,255,255,255,255,255,255,25
    5
320 POKE 53281,1:PRINT"{HOME}":FORL=0TO
    25
330 PRINT"{CTRL BLK}{40SHIFT *}";
340 NEXT:FORL=0TO39:POKE34752+L,64:POKE
    56256+L,0:NEXT
350 REM PRINT MENU
360 PRINT"{HOME}{3CRSRDOWN}           LARRY
    'S FIDDLE TUNES                  "
370 FOR L=33952 TO 33952+39:POKEL,32:NE
    XT
380 FOR X=1TO14:READ T$,M$:A$(X)=M$
390 M1=33957+40*X:M$=T$:GOSUB 1850:NEXT
400 REM SELECT FROM MENU
410 M$="SELECTION:    KEY:    1ST NOTE:"
420 M1=34674
430 FOR L=34552 TO 34791:POKEL,32:NEXT
440 GOSUB 1850
450 PRINT"{HOME}{24CRSRDOWN}{5CRSRRIGHT
    }";
460 INPUT "  ";S
470 IF S<1 OR S>13 THEN 450
480 M$=A$(S):M1=34686:GOSUB 1850
490 X=0
500 ON S GOSUB 610,690,770,850,930,1010
    ,1090,1170,1250,1330,1430,1670,1740
510 GOTO 260
600 REM SUBROUTINES FOR THE 10 TUNES
610 DATA "  (1) SOLDIER'S JOY         ","
      D       F# ON 3"
620 T=1:R=1:GOSUB 8200:GOSUB 1920:IF X<
    2 THEN 620
630 T=1:R=2:GOSUB 8200:GOSUB 1920:IF X<
    4 THEN 630
640 RETURN
```

```
690 DATA "   (2) ARKANSAS TRAVELLER        ","
    D       A ON 4"
700 T=1:R=3:GOSUB 8200:GOSUB 1920:IF X<
    2 THEN 700
730 T=1:R=4:GOSUB 8200:GOSUB 1920:IF X<
    4 THEN 730
740 RETURN
770 DATA "   (3) CINCINNATI HORNPIPE       ","
    D       D/OPEN 3"
780 T=1:R=5:GOSUB 8200:GOSUB 1920:IF X<
    2 THEN 780
790 T=1:R=6:GOSUB 8200:GOSUB 1920:IF X<
    4 THEN 790
800 RETURN
850 DATA "   (4) POP GOES THE WEASEL       ","
    G       G ON 3"
860 T=1:R=7:GOSUB 8200:GOSUB 1920:IF X<
    2 THEN 860
870 T=1:R=8:GOSUB 8200:GOSUB 1920:IF X<
    4 THEN 870
880 RETURN
930 DATA "   (5) GARRY OWEN                ","
    G       G ON 1"
940 T=1:R=9:GOSUB 8200:GOSUB 1920:IF X<
    2 THEN 940
950 T=1:R=10:GOSUB 8200:GOSUB 1920:IF X
    <4 THEN 950
960 RETURN
1010 DATA "   (6) COCK AND HEN             ","
     G       B ON 2"
1020 T=1:R=11:GOSUB 8200:GOSUB 1920:IF X
     <2 THEN 1020
1030 T=1:R=12:GOSUB 8200:GOSUB 1920:IF X
     <4 THEN 1030
1040 RETURN
1090 DATA "   (7) TOM & JERRY REEL         ","
     D       A/OPEN 2"
1100 T=1:R=13:GOSUB 8200:GOSUB 1920:IF X
     <2 THEN 1100
1110 T=1:R=14:GOSUB 8200:GOSUB 1920:IF X
     <4 THEN 1110
1120 RETURN
1170 DATA "   (8) IRISH WASHERWOMAN        ","
     G       D ON 2"
1180 T=1:R=15:GOSUB 8200:GOSUB 1920:IF X
     <2 THEN 1180
1190 T=1:R=16:GOSUB 8200:GOSUB 1920:IF X
     <4 THEN 1190
1200 RETURN
1250 DATA "   (9) MC DONALD'S              ","
     G       B ON 2"
1260 T=1:R=17:GOSUB 8200:GOSUB 1920:IF X
     <2 THEN 1260
1270 T=1:R=18:GOSUB 8200:GOSUB 1920:IF X
     <4 THEN 1270
1280 RETURN
1330 DATA "   (10) TWO FORTY REEL          ","
     G       D/OPEN 3"
1340 T=1:R=19:GOSUB 8200:GOSUB 1920:IF X
     <2 THEN 1340
1350 T=1:R=20:GOSUB 8200:GOSUB 1920:IF X
     <4 THEN 1350
1360 RETURN
1410 DATA "                                ","
     "
1420 REM
1430 DATA "   (11) TUNE THE FIDDLE ","  "
1440 REM
1450 PRINT"{SHIFT CLR}":PRINT "{CTRL BLK}
     {5 CRSR DOWN}PRESS NUMBER":PRINT"OF
     STRING TO":PRINT"BE TUNED"
1470 PRINT "":PRINT"PRESS (0)":PRINT"FOR
      MENU"
1480 PRINT"{HOME}":PRINT"{4 CRSR DOWN}":GO
     SUB 5000:PRINT TAB(27)"{CTRL BLK}{3
     SHIFT B}{SHIFT A}{2 SHIFT B}"
1490 FOR X=1TO3:GOSUB 5010:NEXT:PRINT TA
     B(27)"{SHIFT B}{3 SHIFT A}{2 SHIFT B}
     "
1500 FOR X=1TO5:GOSUB 5020:NEXT:GOSUB 50
     00:FOR X=1TO4:GOSUB 5020:NEXT
1510 PRINT TAB(28)"4321"
1520 K$="":GETK$:IF K$=""THEN 1520
1530 K=ASC(K$)
1540 IF (S=0)OR(K<48)OR(K>52)THEN 1520
1550 ON(K-47)GOSUB 1570,1580,1600,1620,1
     640
1560 GOTO 1520
1570 PRINT"{SHIFT CLR}":GOTO 260
1580 PRINT"{HOME}{9 CRSR DOWN}":C$="E":GOS
     UB 5030
1590 F=659:D=20:GOSUB 8600:GOTO1520
1600 PRINT"{HOME}{5 CRSR DOWN}":C$="A":GOS
     UB 5030
1610 F=440:D=20:GOSUB 8600:GOTO1520
1620 PRINT"{HOME}{6 CRSR DOWN}":C$="D":GOS
     UB 5040
1630 F=294:D=20:GOSUB 8600:GOTO 1520
1640 PRINT"{HOME}{10 CRSR DOWN}":C$="G":GO
     SUB 5040
1650 F=196:D=20:GOSUB 8600:GOTO 1520
1660 REM
1670 DATA "   (12) CHANGE SPEED ","SD"
1680 REM
1690 M$="SPEED 50-500 (HIGHER-SLOWER):"
1700 M1=34674:GOSUB 1850
1710 INPUT "             {CRSR DOWN}";SD
1720 RETURN
1730 REM
1740 DATA "   (13) END ",""
1750 REM
1760 PRINT "{SHIFT CLR}"
1770 PRINT "{HOME}{7 CRSR DOWN}{2 CRSR RIGHT
     }{CTRL BLK} NEXT TIME - DOUBLE NOTE
     S"
1780 T=1:R=21:GOSUB 8200:GOSUB 1920
1790 POKE G+5,0:POKE G+12,0:POKE G+6,128
     :POKEG+13,128:POKE G+24,15
1800 POKE G,135:POKEG+1,33:POKE G+7,62:P
     OKEG+8,42:POKE G+4,33:POKEG+11,33
1810 FOR X=1TO200:NEXT:POKEG+4,32:POKEG+
     11,32
1820 PRINT"{SHIFT CLR}":POKE 648,4:POKE5
     6576,(PEEK(56576)AND252)OR3
1830 POKE 53272,(PEEK(53272)AND240)OR4
1840 END
1850 REM PRINT M$ ROUTINE
1860 F=1:FORL=M1 TO M1+(LEN(M$)-1)
1870 C$=MID$(M$,F,1):F1=ASC(C$)
1880 IF F1<65 THEN 1900
1890 F1=F1-64
1900 POKE L,F1:POKE 21504+L,0:F=F+1:NEXT
1910 RETURN
1920 READD:IF D=100 GOTO 1920
1930 IF D=99 THEN 2020
1940 IF D=0 THEN 1990
1950 IF D<110 THEN 1970
1960 GOTO 1990
1970 READ F
1980 GOTO 2000
1990 F=D:D=1
2000 GOSUB 8600
2010 GOTO1920
2020 X=X+1:RETURN
2290 REM  THERE ARE 16 NUMBERS ON EACH D
     ATA LINE
2300 REM  EXCEPT FOR BEGINNINGS OF PHRAS
     ES AND ENDS OF PHRASES
2310 REM SJ
2320 DATA 100,370,392,440,370,294,370,44
     0,370,294,370,2,440,2,587,2,587
2330 DATA 554,494,440,370,294,370,440,37
     0,294,370,2,392,2,330,2,330
2340 DATA 370,392,440,370,294,370,440,37
     0,294,370,2,440,2,587,2,587
2350 DATA 659,784,740,880,740,587,659,78
     4,659,554,2,587,2,587,2,587
2360 DATA 99
2370 REM  SJ2
2380 DATA 100,587,659,740,659,587,659,74
     0,880,784,740,659,587,554,587,659,7
     40
2390 DATA 784,659,740,659,587,659,740,78
     4,880,740,659,587,554,494,2,440
2400 DATA 2,784,740,659,587,659,740,784,
     880,740,659,587,554,587,659,740
2410 DATA 784,659,740,880,740,587,659,78
     4,659,554,2,587,2,587,2,587
2420 DATA 99
2430 REM  AT
2440 DATA 100,220,247,277,294,370,330,29
     4,2,247,2,247,2,220,2,220,4
2450 DATA 294,330,330,2,330,370,370,2,37
     0,294,370,330,294,2,247,2
2460 DATA 220,294,370,330,294,2,247,2,24
     7,2,220,2,220,4,294,587
2470 DATA 554,587,440,494,587,440,392,37
     0,330,294,277,2,294,99
2480 REM AT2
2490 DATA 100,880,784,740,880,784,740,65
     9,784,740,659,587,740,659,587,554,4
     40
2500 DATA 587,554,587,740,659,587,659,78
     4,740,659,587,740,2,659,740,784
2510 DATA 880,784,740,880,784,740,659,78
     4,740,659,587,740,659,587,554,440
2520 DATA 587,554,587,440,494,587,440,39
     2,370,330,294,277,4,294,99
2530 CH
2540 DATA 100,294,440,370,440,294,440,37
     0,440,587,440,740,440,659,440,740,4
     40
2550 DATA 784,440,740,440,659,587,554,58
     7,659,587,554,494,440,392,370,330
2560 DATA 294,440,370,440,294,440,370,44
     0,587,440,740,440,659,440,740,440
2570 DATA 784,440,740,440,659,587,554,49
     4,554,587,659,784,740,587,2,587
2580 DATA 99
2590 REM  CH2
2600 DATA 100,659,440,440,440,740,440,44
     0,440,784,440,440,440,740,440,440,4
     40
2610 DATA 659,440,740,440,784,440,740,44
     0,659,587,554,494,440,392,370,330
2620 DATA 294,587,587,587,554,659,659,65
     9,587,740,740,740,659,784,784,784
2630 DATA 740,784,880,740,988,784,689,55
     4,2,587,2,587,3,587,99
2640 REM  PGTW
2650 DATA 100,2,392,392,2,440,440,494,58
     7,494,2,392,0,2,392,392,2
2660 DATA 440,523,3,494,2,392,0,2,392,39
     2,2,440,440,494,587,494
2670 DATA 2,392,0,3,659,2,440,523,3,494,
     2,392,99
2680 REM  PGTW2
```

Continued on p. 176

A Trip Down Memory Lane

normally POKEd into this location is 4, so the usual absolute address of screen memory is 4×256, or 1024. You should be aware that this value is not reset to its usual value when the keyboard RUN/STOP RESTORE command is executed. To reset this value, simply type in POKE 648,4 (you won't be able to see this on the screen) and then press [RETURN], and the usual screen memory location will be restored.

Line 130 sets the upper limit of BASIC memory to address 33792. Notice that this is 1K above the 16K boundary of 32768. Because the VIC II doesn't use this 1K, we are giving it to BASIC. The VIC II has no use for it because we are not going to change the offset for screen memory, nor will we program sprite or character definitions in this area.

As far as the VIC II is concerned, screen memory is still at address 1024. But as programmers, we know that 1024 is only an offset and that the absolute value of the top of screen memory is 32768+1024 or 33792. This is the number we used to calculate the "page" in line 120 (33792÷256=132).

Summary

By making the simple changes described above, we end up with over 32000 bytes of memory, which leaves plenty of room for our BASIC programs, even when we're designing our own graphics characters. We accomplished this by simply changing the address where the VIC II gets its information for the TV display and setting a few pointers to let the VIC II know that we did it.

Using this new configuration is simple. To POKE to any location on the screen, add the offset of 1024 to the beginning of the 16K block the VIC II is looking at. In this case it is 32768, so 1024+32768 will equal 33792. This is the address of the upper left-hand corner of the screen. Programming your own graphics is handled in the same way. Simply add the offset of 12288 to 32768 to find out where you start programming your character definitions.

For an example of a BASIC program that uses techniques like those outlined in this article, turn to the Commodore 64 version of *Uncle Larry's Fiddle Tunes* in this issue.

HCM

Uncle Larry . . . from p. 175

COMMODORE 64

```
2690 DATA 100,3,784,2,659,784,740,880,740,3,587,2,784,784,2,659,784
2700 DATA 3,740,2,587,494,2,523,494,2,523,587,2,659,740,2,784
2710 DATA 0,659,2,0,2,440,523,3,494,2,392,99
2720 REM GO
2730 DATA 100,.5,784,.5,740,659,587,523,494,440,392,494,523,494,494,784,740
2740 DATA 659,587,523,494,440,392,440,494,440,440,784,740,659,587,523,494
2750 DATA 440,392,494,523,494,2,494,523,587,659,740,784,587,494,440,494
2760 DATA 440,3,440,99
2770 REM GO2
2780 DATA 100,.5,494,.5,523,2,587,494,2,587,494,2,587,494,587,784,740
2790 DATA 2,659,523,2,659,523,2,659,523,2,659,740,2,784,880,2
2800 DATA 988,880,784,740,659,587,523,494,587,659,740,784,587,494,440,494
2810 DATA 3,440,99
2820 REM C&H
2830 DATA 100,494,523,494,494,440,494,2,794,659,494,523,494,494,440,494,587
2840 DATA 523,494,494,523,494,494,440,494,2,784,659,2,740,587,2,659
2850 DATA 494,587,494,2,440,99
2860 REM C&H2
2870 DATA 100,880,740,587,587,659,740,2,784,659,880,740,587,587,659,740,784
2880 DATA 740,659,880,740,587,587,659,740,2,784,659,740,659,587,2,659
2890 DATA 494,587,494,2,440,99
2900 REM T&J
2910 DATA 100,2,440,2,587,740,587,659,740,784,659,2,587,740,587,659,740
2920 DATA 784,659,587,740,880,1175,1109,988,880,784,740,784,880,988,880,784
2930 DATA 740,659,2,587,740,587,659,740,784,659,2,587,740,587,659,740
2940 DATA 784,659,587,740,880,1175,1109,988,880,784,740,784,659,740,2,587
2950 DATA 99
2960 REM T&J2
2970 DATA 100,440,392,370,440,587,440,370,440,587,440,392,494,587,494,392,494
2980 DATA 587,494,440,494,554,587,659,740,784,659,740,784,659,740,587,554
2990 DATA 494,440,370,440,587,440,370,440,587,440,392,494,587,494,392,494
3000 DATA 587,494,440,494,554,587,659,740,784,659,740,784,659,740,3,587
3010 DATA 99
3020 REM . IW
3030 DATA 100,.5,587,.5,523,494,392,392,294,392,392,494,392,494,587,523,494
3040 DATA 523,440,440,294,440,440,523,440,523,659,587,523,494,392,392,294
3050 DATA 392,392,494,392,494,587,523,494,523,494,523,440,587,523,494,392
3060 DATA 392,2,392,99
3070 REM IW2
3080 DATA 100,.5,784,.5,880,988,784,784,587,784,784,988,784,988,988,880,784
3090 DATA 880,740,740,587,740,740,740,587,740,880,784,740,659,784,784,587
3100 DATA 784,784,523,784,784,494,784,784,587,523,494,440,587,523,494,392
3110 DATA 392,2,392,99
3120 REM MCD
3130 DATA 100,494,3,587,659,587,494,784,494,587,494,784,494,880,440,440,494
3140 DATA 3,587,659,587,494,784,494,440,523,494,440,494,392,392,494,494
```

COMMODORE 64

```
3150 DATA 587,587,659,587,494,784,494,587,494,784,494,880,440,440,494,494
3160 DATA 587,587,659,587,494,784,494,440,523,494,440,494,392,392,99
3170 REM MCD2
3180 DATA 100,494,587,784,988,784,880,784,988,784,587,784,988,784,880,784,659
3185 DATA 784,587,784,988,784,880,784,988,784
3190 DATA 587,659,587,523,494,392,392,330,294,392,394,392,440,392,494,392,294
3195 DATA 392,494,392,440,392,330,392
3200 DATA 494,440,392,370,392,370,330,294,330,370,393,440,494,392,392,99
3210 REM TFR
3220 DATA 100,.7,294,.7,330,.7,370,2,392,494,392,523,392,494,392,523,392
3230 DATA 494,392,440,294,330,370,2,392,494,392,523,392,494,392,523,392
3240 DATA 523,440,392,99
3250 REM TFR2
3260 DATA 100,.7,587,.7,659,.7,740,2,784,.7,880,.7,784,.7,740,784,587
3270 DATA 494,587,2,784,.7,880,.7,784,.7,740,784,880,988,880,2,784
3280 DATA .7,880,.7,784,.7,740,784,587,494,587,659,784,740,880,2,784
3290 DATA 99
3300 REM END
3310 DATA 100,2,587,.7,440,.7,440,2,494,2,440,2,554,2,587,2,0,99,-333
5000 PRINT TAB(27)"{CTRL BLK}{6 SHIFT B}":RETURN
5010 PRINT TAB(27)"{CTRL BLK}{2 SHIFT B}{2 SHIFT A}{2 SHIFT B}":RETURN
5020 PRINT TAB(27)"{CTRL BLK}{SHIFT B}{4 SHIFT A}{SHIFT B}":RETURN
5030 PRINT TAB(34)"{CTRL BLK}{3 SHIFT B}":PRINT TAB(33)"{2 SHIFT B}";C$;"{SHIFT B}":PRINT TAB(34)"{3 SHIFT B}":RETURN
5040 PRINT TAB(23)"{CTRL BLK}{3 SHIFT B}":PRINT TAB(23)"{SHIFT B}";C$;"{2 SHIFT B}":PRINT TAB(23)"{3 SHIFT B}":RETURN
8200 IF T=1 THEN 8250
8210 R=R+1
8220 D1(R)=PEEK(63):D2(R)=PEEK(64)
8230 D3(R)=PEEK(65):D4(R)=PEEK(66)
8240 GOTO 8270
8250 POKE63,D1(R):POKE64,D2(R)
8260 POKE65,D3(R):POKE66,D4(R)
8270 RETURN
8600 FD=INT(F/.06097):HF=INT(FD/256):LF=FD-(256*HF)
8610 POKEG+5,169:POKE G+6,179
8620 POKE G+24,15
8630 POKE G+1,HF:POKE G,LF
8640 POKE G+4,33
8650 FOR ZX=1TOINT(SD*D):NEXT
8660 POKE G+4,32:RETURN
9000 POKE 56578,PEEK(56578)OR3:POKE 56576,(PEEK(56576)AND252)OR1:POKE648,132
9010 POKE56334,PEEK(56334)AND254
9020 POKE1,PEEK(1)AND251:FORI=0TO511:POKEI+45056,PEEK(I+53248):NEXT
9030 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1:POKE53272,(PEEK(53272)AND240)+12
9040 RETURN
```

HCM

FACEMAKER

A Review
by Sharyn Lyon
HCM Staff

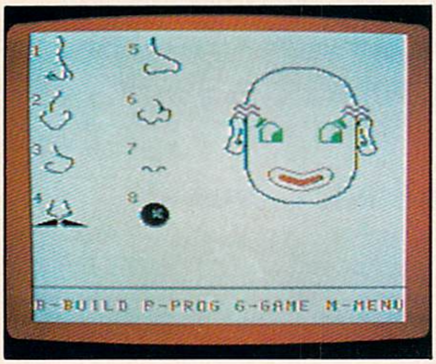
Name:	Facemaker
ProgramType:	Educational game
Machines:	Apple, Commodore 64, TI-99/4A, Atari
Distributor:	Spinnaker Software Corp. 215 First Street Cambridge, MA 02142 Texas Instruments P.O. Box 10508 Lubbock, TX 79408
Price:	C-64 cartridge and Apple (II+,IIe only) disk $34.95 TI cartridge $29.95

	Poor	Fair	Good	Excellent
Performance				
Documentation				
Ease of Use				
Engrossment				
Graphics				

All of us have, at some point in our childhood, heard the phrase, "Don't make faces like that! Your face will freeze that way!" That prediction usually flew off the lips of some well-intentioned adult attempting to persuade children to stop their contortions. Now there is *Facemaker*, and the prospect of hours of perfectly harmless and temporary facial transformations.

Using *Facemaker*, children aged 4-10 can entertain themselves for hours while exercising their imaginations, learning how it feels to control a computer, and stretching their memories.

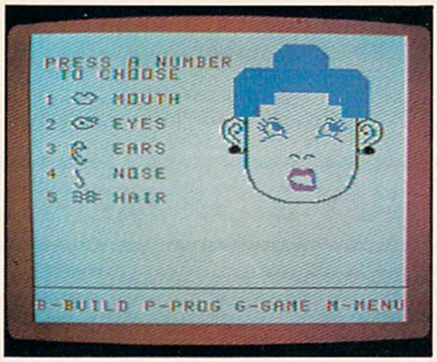
The game starts with a catchy tune, a blank face, and a menu of three options. The user can choose to Build a Face, Program a Face, or Play a Game. You don't have to take up these options in any particular order: If you want to play the game or program a face, you can do it without first building a face.



TI build-a-face option screen

Build a Face

There is an implied hierarchy in this trio, however; so let's start by picking the Build option. This selection has a menu of its own that lists the parts you can use to create your face: Mouth, Eyes, Ears, Nose, and Hair. Next to each word is a picture that makes it possible for non-readers to play on their own with almost no adult intervention. Building a face with five facial parts in eight different styles gives a child all the fun of Mr. Potato Head without any pieces to break or lose. One feature I did miss from the old vegetable version was the selection of hats. Maybe *Facemaker* is missing a good bet by not including this topper.



TI main menu screen

Program a Face

When you and your imagination have settled on a face you'd like to see in action, it's time to try your hand at programming your face. Without losing your creation, you can go back to the first menu and select Program a Face. Each part of your face can perform its own action, and each of the six actions has its own tonal accompaniment. In other words, your face not only smiles (S), frowns (F), sticks out its tongue (T), winks (W), wiggles its ears (E), and cries (C)—it also sings! (Although this addition may seem like a mere frill at this stage, the action/music combination will work like a mnemonic device when you get to the Play a Game option.) To program your face, simply type in the appropriate letter and watch the face react. If you reach the maximum "program" length of 25 actions, the face will perform the actions you assigned automatically, without your having to press any keys.

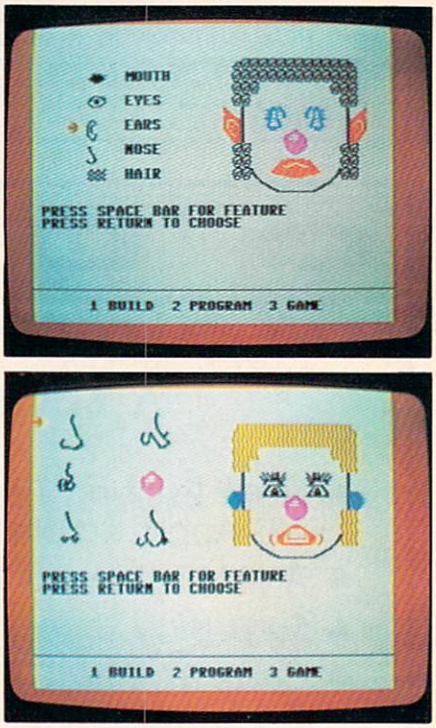
Play a Game

When you think you can keep all these actions in your head, try the Game option. The game does not have instructions, except in the TI version, but it does take you through a trial run to demonstrate the response it's after. In this memory game the face will perform an action. If you type in the letter that stands for that action, it will repeat the action and add a new one. This process continues through up to 25 actions. If you are a whiz, and can keep all 25 actions straight, a special reward comes your way!

You Can't Tell One from the Other Without a Program

There are minute differences among the three different machine versions that we looked at. The main difference among them is in the graphics programming. My personal favorite was the TI version because of the variety of faces possible and because there was a pattern, an inherent order, that satisfied my left brain. To children, who love discovering patterns in things that seem to be without order, this was an appealing feature. I watched a group of children play the game, and they were so delighted with themselves when they discovered that all the option 8 features made a sensible face instead of a silly one, that they felt compelled to share their discovery with anyone who happened to be within shouting distance!

The Apple version has precise, colorful, and varied graphics. The faces range from strangely grotesque to appealingly funny. The Apple version also puts color to good use, but its slowness of execution is a bit annoying. Each time you branch to a different mode, you must wait through two screen re-paintings before you can play the new option.



Commodore menu screens

The Commodore 64 version was the most disappointing, principally because its graphics were so imprecise. For example, of the 8 noses, 2 looked very similar, only one was in color, and the other 5 were more like squiggles than noses.

But let's *face* it—even with these minor flaws, *Facemaker* is a program the entire family will enjoy. It's a fun way to learn a little about the computer keyboard and to feel the excitement of programming the computer to do what you tell it to do. It exercises both your memory and your imagination. HCM

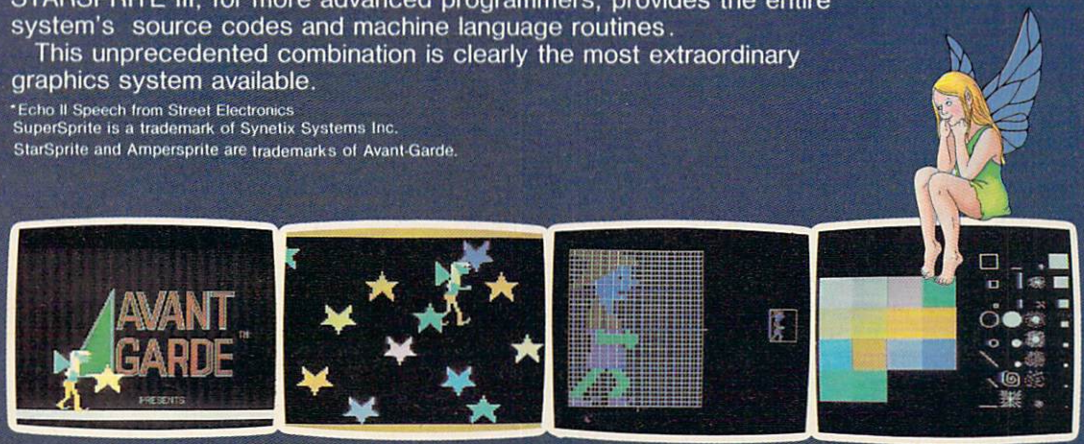
LET YOUR IMAGINATION BLOSSOM!

The STARSPRITE™ SYSTEM from Avant-Garde, paired with the SuperSprite™ Board from Synetix, is the most dynamic and versatile innovation since the Apple itself. Imagine Sprite graphics and Apple graphics together, 16 hi-res colors, 35 video planes, realistic sound and actual speech* . . . the possibilities are limitless.

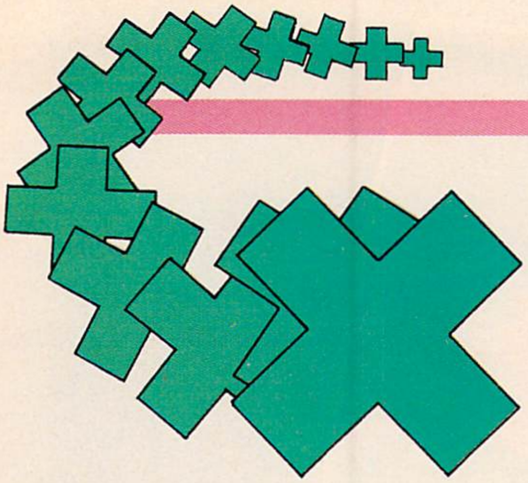
STARSPRITE I shows how to use the SuperSprite Board's graphics and sound, its music and voice capabilities, and includes demos, exciting and alterable arcade games and an introduction to the Ampersprite™ language. STARSPRITE II contains games in addition to programming tools and tutorials for animation, sound effects and scene creation. STARSPRITE III, for more advanced programmers, provides the entire system's source codes and machine language routines.

This unprecedented combination is clearly the most extraordinary graphics system available.

*Echo II Speech from Street Electronics
SuperSprite is a trademark of Synetix Systems Inc.
StarSprite and Ampersprite are trademarks of Avant-Garde.



AVANT-GARDE
(503) 345-3043 P.O. Box 30160 Eugene, OR 97403



Easy as Apple Pie

Starting Simple With Apple Graphics

by W. K. Balthrop
Technical Editor

The first time I sat down at an Apple IIe, I was most excited about trying out its graphics capabilities. I wasn't disappointed. The Apple has a variety of built-in commands that let you plot graphs, draw circles, and create animation. Even three-dimensional figures can be drawn and then rotated to give different perspectives. The programs that follow are a result of my first experiments with Apple graphics. I think you might enjoy them and find uses for them in your own programs.

Spirograph

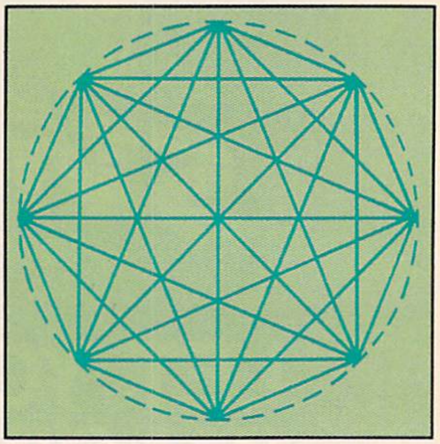
When I was a kid I spent many hours drawing beautiful patterns with a toy "Spirograph." At the time, I had no idea of the advanced mathematics involved in creating my intricate designs. Now, with the help of the computer, those complex equations are a cinch to implement. Applesoft BASIC uses the HPLOT command to draw a line from one point on the screen to another. All you need to figure out is where to start the line and where to stop it: in other words, its end points. For this purpose there is a set of built-in mathematical functions that reduces our work to child's play. The SIN and COS functions can be used to plot the circumference of a circle. To do this we need to pass a value to the function, which is placed between parentheses following the function name. For example, COS(4.33) will yield the cosine of 4.33. The value returned will be a number from 0 to 1. The SIN function works the same way, returning the sine of the number.

The *Spirograph* program uses the HGR command to select the high resolution graphics mode. In this mode, screen resolution is 280x200 pixels instead of the 40x48 displayed by the low resolution mode. You must use high resolution mode to plot lines or draw on the screen. Line 180 sets the high resolution color to 3, which is white. If you changed line 180 to:

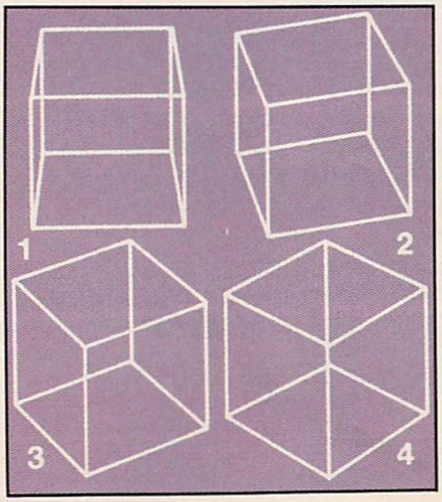
180 HGR:HCOLOR = 2

the spirograph "pen" would draw in violet. Because of the way the Apple creates colors on the screen, different color combinations can be used to give unusual effects.

In line 170, the variable PI is set equal to the value of two times *pi*. If you remember any basic geometry, you'll recall that *pi* is a constant that represents the ratio of the circumference of a circle to the diameter, which is equal to 3.14159265359. . . When using the COS and SIN functions, you need the value two times *pi*, so save a step by including the multiplication in the value of the variable. If you don't understand the finer points of the geometric functions



```
100 REM  **************
110 REM  * SPIROGRAPH *
120 REM  **************
130 REM   BY WILLIAM K. BALTHROP
140 REM   HOME COMPUTER MAGAZINE
150 REM   VERSION 4.1.1
160 REM   APPLE II SERIES APPLESOFT
170 PI = 6.283185307
180 HGR : HCOLOR= 3
190 HOME
200 VTAB 21: HTAB 1: INPUT "NUMBER OF P
    OINTS IN THE CIRCLE?";N
210 INPUT "COLOR (1-7):";C: HCOLOR= C
220 FOR X = 0 TO PI STEP PI / N: FOR Z
    = 0 TO PI STEP PI / N
230 A = INT ( SIN (X) * 110) + 140
240 B = INT ( COS (X) * 62) + 70
250 C = INT ( SIN (Z) * 110) + 140
260 D = INT ( COS (Z) * 62) + 70
270 HPLOT A,B TO C,D
280 NEXT Z: NEXT X
290 PRINT "PRESS ANY KEY": GET A$: GOTO
    180
```



(such as *pi*, sine, and cosine) used in this program, don't worry. The computer does these calculations for you.

You will be prompted to input the number of points you wish to plot in your spirograph. A value of 1 will plot a single point, while a 2 plots a single line, a 3 plots a triangle, and 4 plots a square. For any number 4 or higher, the spirograph will also draw diagonal lines connecting the corners of the figure. The higher the number of points, the more closely the shape will resemble a circle. Try experimenting with different numbers to create some elaborate shapes. Here are some values you might like to try:

5,6,8,12,16,64

Now we come to the tricky part. Understanding the use of the SIN and COS functions in this program may take some effort, but it's still easier than doing the computations yourself, without the help of the computer.

Nested Loops

Line 220 sets up the two nested FOR–NEXT loops that drive the spirograph and cause it to create the intricate web of criss-crossing lines. The loops are identical except that the inner loop will complete one full execution for each step of the outer loop. The outer loop uses a counter, X, to determine a starting point on the circumference of the circle, and the inner loop uses the variable Z as a counter to determine the end points. The spirograph will draw lines from the starting point to connect it with each of the end points.

> **"If you don't understand the finer points of the geometric functions (such as *pi*, sine, and cosine) used in this program, don't worry. The computer does these calculations for you."**

To find the starting point, we calculate the SIN and COS of X (using pre-defined functions, of course). This value stays the same throughout all executions of the inner loop. To find the end points, we calculate the SIN and COS of the variable Z each time the inner loop executes. Z is incremented by steps that equal a value between 0 and PI. Each value represents a position on the circle: For instance, 0 represents the origin, .25*PI would be one-quarter of the way around the circle, and .5*PI would be halfway around. The STEP option in the FOR–NEXT loop is set to PI/N, where N is the number of points you want to have drawn in the spirograph.

To establish the size of our circle, we multiply SIN and COS by a chosen number to determine the radius in pixels. If the size is different for the SIN and COS, as it is in the program above, then you will not get a perfect circle. You will get an oval shape instead. I made the spirograph wider than it is tall to fill the screen out better. The value added to the equation is the offset. This determines positioning on the screen. This value should always be at least as large as the multiplier; otherwise, you will draw off the edge of the screen.

Line 270 of the program draws the actual line. The line will start at the pixel positions specified by the coordinates of the starting point, A and B, and end at coordinates C and D.

Rotation

Now that you know how SIN and COS can be used to create interesting patterns, you might like to see some other applications. The computer can perform some amazing acrobatics when we put it to the task. The ability to create a box on the screen in three dimensions is interesting, but consider the ability to take that same box and rotate it 360 degrees. The following program will place a stick-figure box on the screen and then rotate the box.

```
100 REM   ************
110 REM   * ROTATION *
120 REM   ************
130 REM    BY WILLIAM K. BALTHROP
140 REM    HOME COMPUTER MAGAZINE
150 REM    VERSION 4.1.1
160 REM    APPLE II SERIES APPLESOFT
170 HGR
180 PI = 6.283185307
190 FOR A = 0 TO PI STEP .2
200 X = INT ( SIN (A) * 20) + 140
210 Y = INT ( COS (A) * 10) + 30
220 X1 = INT ( SIN (A + PI * .25) * 20) + 140
230 Y1 = INT ( COS (A + PI * .25) * 10) + 30
240 X2 = INT ( SIN (A + PI * .5) * 20) + 140
250 Y2 = INT ( COS (A + PI * .5) * 10) + 30
260 X3 = INT ( SIN (A + PI * .75) * 20) + 140
270 Y3 = INT ( COS (A + PI * .75) * 10) + 30
280 HCOLOR= 0
290 HPLOT H,I TO H1,I1 TO H2,I2 TO H3,I3 TO H,I TO H,I + 20 TO H1,I1 + 20 TO H2,I2 + 20 TO H3,I3 + 20 TO H,I + 20: HPLOT H1,I1 TO H1,I1 + 20: HPLOT H2,I2 TO H2,I2 + 20: HPLOT H3,I3 TO H3,I3 + 20
300 HCOLOR= 3
310 HPLOT X,Y TO X1,Y1 TO X2,Y2 TO X3,Y3 TO X,Y TO X,Y + 20 TO X1,Y1 + 20 TO X2,Y2 + 20 TO X3,Y3 + 20 TO X,Y + 20: HPLOT X1,Y1 TO X1,Y1 + 20: HPLOT X2,Y2 TO X2,Y2 + 20: HPLOT X3,Y3 TO X3,Y3 + 20
320 H = X:I = Y:H1 = X1:I1 = Y1:H2 = X2:I2 = Y2:H3 = X3:I3 = Y3
330 NEXT A: GOTO 190
```

This program uses the same PI variable (equal to 2**pi*) as the previous one. The rotation of the box is controlled by a loop beginning in line 190. Every time the loop is completed, the box will make one full rotation. Line 330 will cause the loop to repeat infinitely, spinning the box until you press [CONTROL] [C]. You can adjust the speed of rotation by changing the STEP value of .2 in line 190. Increasing the STEP will speed up rotation, and decreasing the value of the STEP will slow down rotation.

Continued on p. 182

HCM Program Bug

DeBUGS on DisPlay

Bugs—problems that cause programs not to function properly—are an inevitable consequence of programming. Programs are complex systems that are difficult to test exhaustively. And occasionally bugs slip through even the best testing procedures. When such a bug in one of our programs comes to our attention, we print a correction in this column so that our readers can correct their programs as soon as possible.

The author of the article **Public Investigator** (August, 1983) informs us that two errors slipped by him and onto the master tape he sent us. Lines 930 and 950 should be altered as follows:

```
930 FOR I=I TO Q
950 NEXT I
```

This adds enough "@" signs to finish the response string if a user stops in the middle of the questionnaire.

As we noted before, our version of **Success Formula** wasn't too successful. The author has sent along the following changes necessary to make it work better:

```
2920 RESTORE 5830
2960 RESTORE 5835
```

and

```
5835 DATA 7,6,10,10,2,16,5,20,10,
     22,3,24,4,25,5
```

Our thanks to David M. Douglas and Oris Bud Davis for straightening us out.

Robert Schenk, author of **Grisly Adventure** (October 1983), has just written to say that a reader has informed him of a bug in the program. Line 2850 should read as follows:

```
2850 IF (I<>G) + (P<>F) THEN 2880
```

"As written," he writes, "it is imposible to kill the bear by shooting either right or left. The only way to kill the bear is to shoot it from above or below.

"Also, I have had many calls and letters form people who got a data error in line 270. They do not realize that their problem is not a mistake in the **READ** statement in 270, but in a **DATA** statement at the end of the program. To isolate the mistake, one can type in **PRINT B,A$** when the error occurs and the computer will tell one the last value that the program read correctly. Perhaps you can pass this along to your readers because they will have similar errors in many of the other programs you publish."

We're glad to publish Robert's debugging hint, and thank him for the correction to **Grisly Adventure.**

October's gremlins must have slipped into the wizard's keep. In your **Escape from Wizard's Keep,** if you get into the room with no exit, you really get trapped. Instead of a chance to play again, you got * SYNTAX ERROR IN 970, and the Extended BASIC screen. Line 970 should read:

```
970 CALL COLOR(13,1,1)
```

If you tried to modify **The Poor Man's Program Loader** by Rick Rothstein (Letters to the Editor, p. 7, November 1983) according to the directions accompanying it, you probably got as befuddled as we did when we tried. The directions for modifying the program names LOADER and CAT should refer you to lines 230 and 250 respectively.

HCM

Apple Pie

Lines 200-270 calculate the four corners of the top of the box. There are two coordinates for each corner. We don't need to calculate the bottom four corners of the box because they will always be directly under the top four corners. Notice that A is the only variable used to find all four points of the top of the box. This is because the value of PI represents one complete rotation around an imaginary circle, and the other points of the box can be represented as positions on the circle by adding fractions of PI to A. If you add .25*PI to A, you get a point one-quarter of the way around the circle from A. The corner opposite A is .5*PI+A, and .75*PI gives the corner three-quarters of the way around the top of the box.

The size of the box is specified by multiplying COS and SIN by a chosen number, in this case 10 or 20. The number added to this total is the offset, and is used for screen positioning. The actual box is drawn in line 310, using the points which were calculated earlier. Line 300 sets the color, which is white in this listing.

If you don't erase your old box before creating a new one, you will soon end up with a white blob on the screen. The HPLOT statement in line 290 will draw the box created in the previous pass in black, effectively erasing it from the screen. Line 280 sets the line color to black. In line 320 the coordinates from the new box are saved so that *it* can be erased on the next pass, before a new one is drawn.

Animation Plus

```
100 REM  *******************
110 REM  *  ANIMATION PLUS  *
120 REM  *******************
130 REM    BY WILLIAM K. BALTHROP
140 REM    HOME COMPUTER MAGAZINE
150 REM    VERSION 4.1.1
160 REM    APPLE II SERIES APPLESOFT
170 HGR
180 FOR X = 7676 TO 7686
190 READ A: POKE X,A: NEXT X
200 DATA 1,0,4,0,44,46,62,62,60,44,0
210 POKE 232,252: POKE 233,29
220 HCOLOR= 3
230 FOR X = 1 TO 63 STEP 2
240 ROT= X * 2
250 SCALE= X / 2 + 1
260 XDRAW 1 AT 140,96
270 XDRAW 1 AT 140,96
280 NEXT X
290 GOTO 230
```

The first time I worked with graphics on the Apple, I looked for a facility to redefine the character set and discovered an even better way of dealing with graphics shapes. Often when you are designing a game, the line-drawing commands aren't quite suitable. That's where the Apple DRAW statement comes in handy. This statement lets you draw complex shapes on the screen without having to redefine characters or draw every line. In addition, you can easily expand the shape with the SCALE= function or rotate it with the ROT= function. By defining several shapes and using these functions you can create animation that rivals "sprite" (smoothly moving screen objects) capabilities. In this program I created a simple plus sign, then expanded and rotated it at the same time. The result is interesting, and quite easily accomplished, as you can see by the size of the listing.

The FOR-NEXT loop in lines 180-190 reads the data in line 200 and places it in memory beginning at address 7676, which is just below the high resolution page 1.(High resolution has two pages for memory storage, but can display only one at a time.) This data sets up a shape table to create the plus sign. The first four numbers in the table are the table index. The first value indicates the number of shapes defined in the table. The second number is not used. The third and fourth numbers specify the offset from the start of the index to the first shape data. In this case the offset for the only shape is 4 since there are four bytes before the shape data starts. If you add another shape to the end of this one, you will need an entry in the index. The first value will have to be changed from 1 to 2, and the third value will have to be changed from 4 to 6. Then the start of the new table must be inserted between the offset of the first table and the start of the first shape table. The new DATA line would look something like this:

DATA 2,0,6,0,13,0,44,46,62,62,60,44,0,. . .(new shape data)

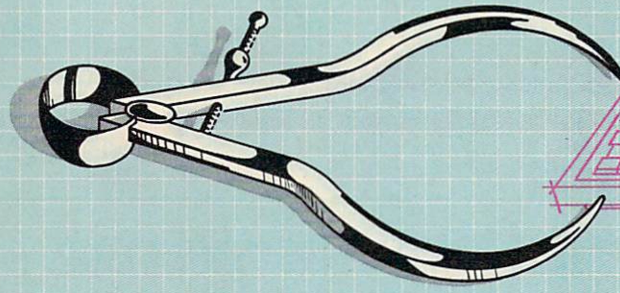
Before the shape table can be used, the address of its index must be POKEd into memory (line 210); otherwise, the computer won't know where the table is. The table can be put anywhere in free memory.

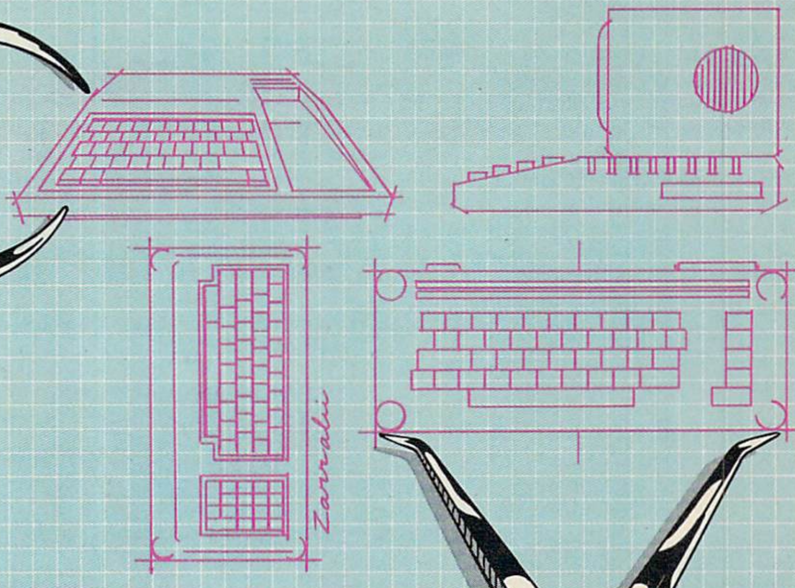
The HCOLOR= function in line 220 sets the color for the DRAW and XDRAW commands, as well as the HPLOT command. The ROT= function controls rotation. The loop in line 230 increments up to 63, representing one full rotation. Values from 64 to 127 will cause one rotation, and any number over 255 will cause an error. The SCALE= function in line 250 sets the scale of your shape with only one command.

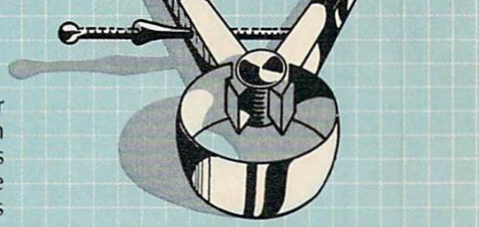
Lines 260-270 are the statements responsible for placing the shape on the screen and then erasing it. The XDRAW statement is used because it will cause the shape to automatically erase itself the second time it's drawn in the same place. Try leaving line 270 out of the program, and you will see the results. Then slip it back in, sit back, and enjoy the show. The old spirograph was never quite like this.

HCM

MICROCOMPUTER ACCURACY







by D. W. Whitcombe

The computer is supposed to be accurate beyond question, and some people go so far as to equate computer output with Truth. It may come as a surprise, then, to find that most microcomputers are not as accurate as the scientific Texas Instruments (TI) or Hewlett Packard (HP) hand-held calculators.

Using the BASIC programming language provided with a particular computer, you can perform a simple test to evaluate computer accuracy. The test can be applied to any computer on first inspection. It evaluates the following forms for X= .01 to 1.0 in small steps:

1. ATN(TAN(X)) – X
2. TAN(ATN(X)) – X
3. LOG(EXP(X)) – X
4. EXP(LOG(X)) – X
5. (X ∧ 10) ∧ .1 – X

The results are as follows:

COMPUTER	SIGNIFICANT FIGURES
Control Data Cyber	15
TI-99/4,TI-99/4A,TI CC-40	13
TI 59 (hand-held)	11
Apple,(1) Commodore,VIC-20	9-10
TRS 80 Model 1	7
TRS Model 100	7
IBM PC, COMPAQ	7
Epson HX-20	7
Atari 1200 XL	7
Sanyo MBC-1000	6

(1) The Apple gives 9 figures on 4*ATN(1).
NOTE: The Cyber and the TI provide numerical round-off protection so that glaring errors within the precision of the software do not occur.
[NOTE: The IBM PCjr, with Cartridge BASIC, is capable of accuracy to 16 significant figures with variables defined as double precision.—Ed.]

I have three home computers: one at home for myself, one for my wife and son to play games with, and one at work. I chose the TI Home Computer because it is accurate enough to perform engineering and scientific computations without modification. These include operations as complex and diverse as matrix inversion, numerical differentiation and integration, satellite orbit computation, Monte Carlo simulation using random numbers, and filter and optimization analysis.

"Now we have home computers—and we expect them to do nearly everything that the large computers do."

I performed similar computations, but with smaller programs, on hand-held calculators such as the TI 59 or the HP 41. The TI 59, for example, was programmed for 3D satellite orbit computation including two orbit adjusts, using closed-form formulas. This program filled the storage on the TI 59. If you wanted to add any embellishments (e.g., a powered flight vehicle stage or more orbit adjustments), then you would need a computer's larger memory. Significantly increased storage is also desirable because there is a tendency for a program to grow to many times its original size as needs change with time.

The TI-99/4A fulfills this requirement for increased storage without sacrificing accuracy. It provides a minimum of 13 significant figures, and 10 of these are displayed in TI BASIC programming. All 13 figures can be displayed when Extended BASIC is used.

The Apple and the Commodore versions of BASIC provide only 9 to 10 significant figures and are not protected against round-off errors. For example, these computers give an incorrect answer of 1.65 when the following BASIC program is RUN:

```
10 PRINT INT ((1.655 + .005)*100)/100
```

The TRS 80, IBM PC, COMPAQ, Epson, and Atari computers use a BASIC supplied by Microsoft that provides about 7 significant figures. Some of these BASICs use double precision, which provides a 16-digit display. However, most of the mathematical functions are good to only 8 figures, even in the double precision mode. (It is considered undesirable for any computer to print or display more than one incorrect digit, if this can be avoided.) To see for yourself, run the following program on any of these computers:

```
10 PRINT 4*ATN(1)
```

Then compare the answer with 3.141592653589793. The Commodore and Apple print the answer out to 9 figures; the TI prints 10 and the IBM prints 7. You could obtain the 16-digit figure with the proper software accompaniment, and Apple users are fortunate in having a large selection of software alternatives available. In general,

Continued on p. 193

ART AT YOUR FINGERTIPS

A COLOR GRAPHICS SYSTEM FOR YOUR APPLE

By W. K. Balthrop
HCM Staff

Apple graphics are making great strides these days. There are more and more products available—printer interface cards, touch-sensitive pads, graphics printers—for the home user who wants to try his hand at the formerly formidable task of designing Apple graphics. We review here three devices that can work together as a system to help you create, save and print beautiful screen displays without a lot of complicated programming. We also include program listings for the Apple computer that allow you to draw graphics directly on your screen.

Koala Pad Graphics on the Apple

If you've ever wished you could program intricate graphics without keeping track of a lot of addresses for PEEKing and POKEing numerous strange codes, you'll appreciate the Koala Pad. Even an inexperienced programmer can create graphics displays on the Apple computer with a minimum of programming. The Koala Pad allows you to create shapes on the screen simply by drawing on a pad with a blunt stylus (provided)—or even with your finger.

The Koala Pad and Transtar 315 printer make a perfect match for producing graphics.

Koala Pad
Koala Technologies Corp.
3100 Patrick Henry Drive
Santa Clara, CA 95050
$125.00

Transtar 315
Transtar-Vivitar
P.O.Box 96975
Bellevue, WA 98009
$599.00

PICS Interface Card
Transtar-Vivitar
P.O. Box 96975
Bellevue, WA 98009
$119.95

The Pad

The Koala Pad is a lightweight, pressure-sensitive input device for several popular computers such as the Apple II series, Commodore 64 and IBM PC. Although it's intended primarily for graphics, a number of software packages are available to set up the Koala Pad as a custom keyboard with overlays. It can also be set up to select items from a menu screen.

The Koala Pad's black and cream color and sleek, professional look allow it to fit in at the office or at home, right alongside your Apple. The pressure-sensitive drawing surface is made of durable, black plastic, and we found it still looked brand new even after repeated use with the stylus. The manufacturer does warn, however, not to use sharp or pointed objects on the pad, as this may damage it. The pad can be cleaned with a damp cloth, but it is not waterproof.

The pad's five-foot cable is long enough for any application at the computer. The cord comes out the back of the pad and goes into the game port inside the Apple. One minor drawback is the lack of strain relief for the cable; a good tug could pull it out of the socket and possibly damage the socket plug.

The Koala Pad connects to the game port using the paddle input line. To read the pad from BASIC, you use the PDL(n) function, where n = 0 for the X coordinate and n = 1 for the Y coordinate. The range of input is from 0 to 255. There are also two buttons available, and their status can be read by PEEKing locations −16286, and −16287.

The Graphics Program

The Koala Pad package includes a disk which contains the *Micro Illustrator* software. This software allows you to easily draw pictures on the screen and save them to disk. In a matter of minutes, you can create excellent graphics without doing any real programming at all. Once the program is loaded and running, you never need to touch the keyboard except to load or recall a graphics screen on the disk.

After you boot up the *Micro Illustrator* disk, the title screen appears; then you press one of the two buttons on the pad to go to the menu screen. Unlike most menu screens, this one gives you a graphic representation of the functions as well as a descriptive word or two. There are 15 modes, 8 brushes, and two color palettes with 9 colors and color combinations on each. To select an item on the menu, you move your stylus or finger across the pad until the cursor moves over the item you want, then press one of the two buttons. The option which has been selected displays a white triangle in the upper left corner of the items box.

We noticed that at times the response of the cursor seemed a bit erratic. This was generally due to a slight, involuntary lifting of the stylus or finger from the surface. The pad's response is quite sensitive so you must be careful to keep the pressure even in order to send clear signals to the computer.

The Micro Illustrator menu screen gives the user a large number of options. All options can be selected by moving the cursor at the pad.

Another drawback we noticed was that you cannot always use all the colors offered on the menu screen in your drawings. Instead, you must select one of the two COLOR SETs at the beginning of the drawing. You cannot change to the other set in the course of the drawing without disastrous effects. These features take some getting used to, but on the whole, the *Micro Illustrator* is versatile. The various modes and brushes make creating graphics fun and easy.

Micro Illustrator Commands

DRAW lets you draw continuous lines on the screen. The line drawn corresponds to the path you trace on the pad.

POINT turns on a single pixel when you touch the pad. This command is good for final, small, intricate designs or touch-ups.

LINE lets you draw a line between two points. You select the first point, move the stylus across the pad, and watch as the line forms on the screen. When the line is positioned where you want it, press the button.

LINES is identical to LINE except that the next line starts at the current cursor position. This feature lets you draw connected lines.

RAY has you select the center of a ray, move the stylus to the end point, and press a button when you have what you want.

FILL fills an area with a selected color. The FILL continues until it encounters a border color other than the one chosen. One important point about FILL—be sure you have your stylus off the pad when you press the button to finalize your fill selection. If you are not in complete control, this command can fill the whole screen almost instantly.

FRAME lets you draw boxes the easy way. You place the stylus where you want one corner of the box, move the stylus around to where you want the opposite corner, and when you press the button, the box forms on the screen.

BOX works the same way as FRAME except that the box is filled in with the color you select.

CIRCLE has you select the center of the circle, then move the stylus to determine where the edge will be.

DISC works the same way as CIRCLE except that the circle is filled in with the color of your choice.

MAGNIFY enlarges the image on the screen 7 times. This mode is essential for fine-tuning your drawing, as it takes you inside your drawing. You can no longer see the whole picture—now each pixel is enlarged to the size of a character. The size of your stylus is also magnified, so moving it now is like moving a small window around on your drawing. Of course, the size of your movements is also magnified. You may begin to feel as though you are moving around in a zero-gravity environment! It takes concentration, a steady hand, and perseverance to create good curves in this mode.

NORMAL returns the magnification to normal.

ERASE clears the screen and lets you change the screen color to any of those available.

STORAGE escapes from the main sequence and takes you back to the keyboard to save or load a screen via disk. Screens stored on disk can be called up from BASIC using BLOAD.

HELP displays instructions about the menu and how to use the system.

BRUSH SET lets you select any one of eight "brush strokes." Each brush creates a different effect.

COLOR SET selects one of the two palettes of color at the beginning of a drawing. You use only one palette at a time.

Koala Pad Is Easy and Fun

The Koala pad is one of the easiest graphics products we have ever used—and one of the most enjoyable. You create computer art by simply drawing on the pad. A child could master the skills required in no time; yet it's interesting enough to keep an adult engrossed for hours. Koala pad can bring out the latent artist in you.

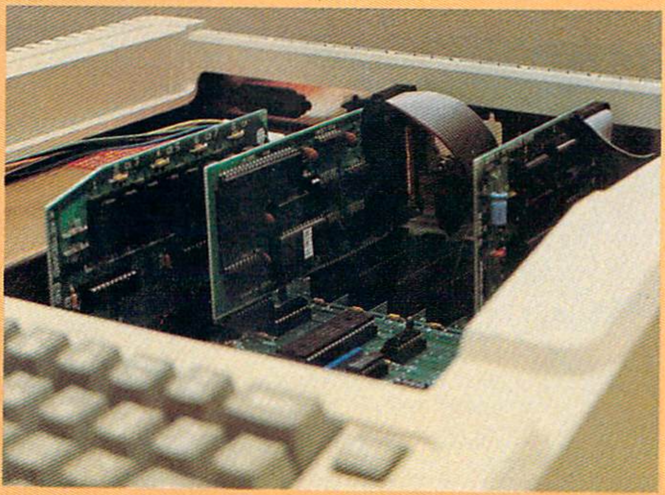
The Koala Pad lends itself to "serious" programming applications too. With its capabilities you can develop lavish graphics for your BASIC programs, store the screens on disk, and then call them up as needed. You can enhance your BASIC programs without generating miles of complicated graphics code.

The Transtar 315 Printer

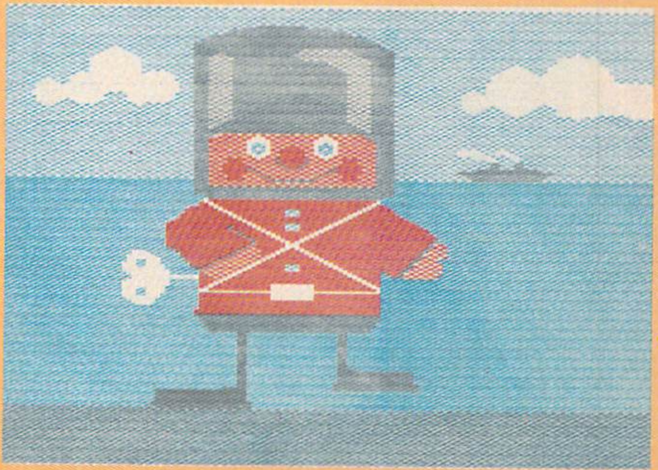
Your Apple and the Koala Pad allow you to easily design colorful graphics on the screen, but what if you want a hard copy of your creation? Conventional printers are fine for text applications, but your Koala color graphics require something better. Vivitar Computer Products has recently released two products that handle full-color screen dumps with excellent results. The Transtar 315 printer and the PICS Interface Card work together to make high-quality screen printing simple and faithful to the screen image.

The Transtar 315 is a *unidirectional* dot matrix printer. This means that the head prints only when it is traveling from left to right. The special four-color ribbon and the four print hammers are the secret of this printer's ability to create seven different colors. In addition to the

The PICS Interface Card transmits a full-color screen dump of any screen to the Transtar 315 color graphic printer.



The picture shown above is a screen photo of graphics created with the Koala Pad, and the Micro Illustrator software. Below is a hardcopy of that screen produced on the Transtar 315 printer, using the PICS Interface Card.



ribbon's straightforward red, blue, green and black, other colors can be created by striking different combinations of heads in the same spot. The range of colors can be extended even further by "dithering" colors together. This involves alternating dots of different colors. When they are very close together, the dots appear to blend to form another color. Take a close look at pictures in a comic book, and you'll see this technique at work.

The printer can be set for two different line densities. At 10 cpi (characters per inch), the printer is capable of speeds up to 38 cps (characters per second). At 13.3 cpi, the speed is 50 cps. This is relatively slow compared to single-color printers in the same price range, but not a bad speed when you consider the extra work the Transtar must do to get the colors mixed properly. To control the color of each dot on the page requires the decoding of a great deal of information.

The Transtar uses friction feed and tractor feed at the same time, so no switching of mechanisms is required. The first 300 printers shipped had a paper feed problem caused by too much tension on the friction feed rollers at the point of paper entry. As a result, paper fed via the tractor holes tore and jammed the machine. Vivitar informs us that the problem has been remedied.

On the top right side of the printer are four buttons and three LEDs. The STOP, LINE FEED, and FORM FEED buttons are fairly standard. The COPY button will start a screen copy of the graphics displayed on the screen, assuming you have the PICS Interface Card installed in your Apple.

The Transtar 315 boasts several "soft switches" that are not found on most printers. A "soft switch" is a programming feature that lets you send special instructions to the printer to alter its performance. You can specify the color of the text and graphics, select single or double pass printing (double pass will print each line twice), and repeat a graphics character a specified number of times. The RGB raster scan mode lets you specify the color of each individual dot, and the print hammer raster scan mode gives you direct control of the print hammers for the different colors.

The PICS Interface Card

The PICS Interface Card allows Apple users to send full-color screen dumps to the Transtar 315 printer. The printer interface cable comes with the card, and installation is simple. The PICS card must go in slot #1, so you will have to relocate a card if you normally use slot #1 for something else.

Using the PICS Interface Card

Using the PICS card is almost as easy as installing it. Screen copies can be made from within a program, or externally via the printer COPY switch. There are three methods of printing with the card:

1. Printing from within your BASIC program requires only two simple statements. First you need to direct output to slot #1 with the PR#1 command. Then use PRINT CHR$(9) followed by a number from 0 to 1023 and the letter H in quotation marks. In the example:

PRINT CHR$(9);"137H"

the number in the quotes specifies certain parameters for the copy, and H specifies the COPY SCREEN command.

2. You can also press [CONTROL][I] in Immediate mode, then enter the parameter numbers followed by the letter H. To begin printing, press the space bar.

3. Finally, to use the the COPY button on the printer, simply press the COPY button, enter the parameters you have selected, and press the space bar.

Here is a list of parameters which can be selected from the options in the screen dump command:

1. Print from low-resolution (LORES), or high-resolution (HIRES) page.
2. Print 24 or 20 lines per page.
3. Print black on the screen as white on paper, or black on the screen as black on paper.

"In a matter of minutes, you can create excellent graphics without doing any real programming at all."

4. Left-justify or center the printed results on paper.
5. Fill every dot, or copy dots to correspond with those on the screen.
6. Copy from HIRES page 1 or 2.
7. Print at normal or double width.
8. Print at normal or double height.
9. Print in color or black and white.
10. Make one or two passes on each line.

In addition, there are several codes that control line length and line feed. For example, to output text to the printer you can use:

PR#1:PRINT CHR$(9);"40N"

where the value 40 sets the page width. If a value larger than 40 is used, no characters will be displayed on the screen until printing is completed.

Made for Each Other

The PICS Card is so easy to install and use that within half an hour you should be printing color graphics without any problem. Because Vivitar designed the card and the printer to work together, there is excellent firmware support, which simplifies the programmer's task.

The Transtar 315 color printer comes with a 72-page manual that covers installation, configuration, and programming printer graphics. The manual is well-illustrated and easy to read. The PICS Card manual contains 13 pages of instruction on how to access the card and print screen dumps. The size of the PICS manual is a testament to how simple the card is to use.

Putting It on Paper

The screen graphics we created with the Koala Pad were saved on disk under the name MOUNT using the STORAGE command. The BASIC program listings below bring the graphics into memory (using BLOAD, or Binary Load) and dump the picture to the printer.

The following program (Listing 1) loads and displays the screen and then dumps the screen to the Transtar 315. To load your own screens, simply replace the word MOUNT in line 110 with the title of your screen. Lines 120 through 150 output the screen to the Transtar 315 color graphics printer via the PICS Interface Card. From high resolution mode 2 (Line 100), line 110 loads and displays the screen. HIRES page 2 will be loaded even if you're not in that screen mode. So you need to POKE 0 into locations −16304, −16297, and −16299 for the screen to appear without erasing its contents as the HGR2 command does (see Listing 2).

LISTING 1

DISPLAY GRAPHICS AND PRINT TO TRANSTAR 315 COLOR PRINTER

```
100 HGR2
110 PRINT CHR$(4);"BLOAD
      PICTR.MOUNT"
120 PR#1
130 PRINT CHR$(9);"741H"
140 PR#0
150 TEXT
```

LISTING 2

DISPLAY GRAPHICS WITH POKES

```
100 PRINT CHR$(4);"BLOAD
      PICTR.MOUNT"
110 POKE -16304,0:POKE -16297,0:
      POKE-16299,0
120 GOTO 120
```

Simple but Rewarding

When you look at the screen graphics and printer graphics reproduced here, you might imagine that they are the result of long and arduous programming. And indeed, without the fine tools reviewed here, such complex graphics on a printed page are long, complicated programming tasks. But with the Koala Pad, Transtar 315, and PICS Interface Card, they become nearly child's play.

HCM

PLEASE DON'T FORGET TO RETURN THE QUESTIONNAIRE ON THE FRONT BIND-IN CARD.

Slots . . . from p. 74

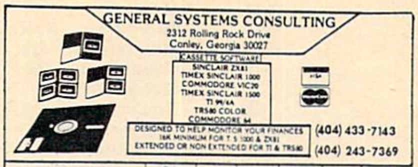
from p. 74

Slots
Explanation of the Program

Line Nos.	
100-170	Program Header.
180-320	User instructions.
330-1110	Set up and display slot machine.
1120-1150	Set up for new player.
1160-1290	Play.
1300-1870	Scoring.
1880-1930	Display score.
1940-1990	Check keyboard for new input; loop accordingly.
2000	End of main program.
2010-2330	Data to display slot symbols.

TI-99/4A

```
100 REM  ***************
110 REM  *    SLOTS    *
120 REM  ***************
130 REM  BY BOB STOFFERS
140 REM  HOME COMPUTER MAGAZINE
150 REM  VERSION 4.1.1
160 REM  TI BASIC
170 REM
180 CALL CLEAR
190 PRINT "NEED INSTRUCTIONS (Y/N)"
200 CALL KEY(0,K,Y)
210 IF K=78 THEN 330
220 IF K<>89 THEN 200
230 CALL CLEAR
240 PRINT "THE SLOT MACHINE CONTAINS
    CHERRIES,SEVENS,BARS,LEMONS,BELLS,
    AND ACES.   POINTS ARE"
250 PRINT "SCORED FOR MATCHING THE SYM-
    BOLS IN THE MACHINE'S THREE WINDOWS
     FOR THE FOLLOWING   COMBINATIONS:"
260 PRINT TAB(26);"PTS":"*ALL THREE MAT
    CH.........100*FIRST TWO MATCH & AR
    E:         ACES,SEVENS OR BARS.... 50
    "
270 PRINT "    CHRYS,LMNS, OR BELLS... 30
    *FIRST & THIRD MATCH.....  15*LAST T
    WO MATCH..........10"
280 PRINT "*NO MATCH.................-10
    "::"TO PLAY PRESS:              ""N
    "" TO CLEAR PREVIOUS"
290 PRINT "       PLAYER'S TOTALS.
    ""P"" TO ACCUMULATE TOTALS.     ""S""
     TO STOP GAME.":::
300 PRINT "***PRESS ANY KEY TO BEGIN***
    ";
310 CALL KEY(0,K,S)
320 IF S=0 THEN 310
330 CALL CLEAR
340 DATA 0000000000000103,0000000000F8F
    CFE,070F1F3F3F3F3F3F,FFFFFFFFFFFFFF
    FF,3F3F3F3F3F3F3F3F,0080C0E0E0E0E0E
    0
350 DATA 3F1F1F0F0701,FFFFFFFFFFFF,FFFF
    FFFFFEF8,E0C080,000000000F0F0F0F,00
    000000FFFFFFFF,00000000F0F0F0F0
360 DATA 0F0F0F0F,FFFFC0C0,FFFF070F1F3F
    7FFF,F0F0F0F0F0E0C080,0103070F1F3F7
    FFF,FFFEFCF8F0E0C080,0103070F
370 DATA FFFEFCF8,00,00,00,3F3F3F3F3F3F
    3F3F,FCFCFCFCFCFCFCFC,0000000000000
    001,0000001E3E7EE0C0,03070E1C3870E0
    C0
380 DATA 80,FFFFFFFFFFFFFFFF,00,0000000
    1030F3F7F,00000080C0F0FCFE,FFFFFFFF
    FFFFFFFF,0103070707,FFFFFFFFFF0303
390 DATA FFFFFFFFFFC0C0,80C0E0E0E0,00,0
    00103070F1F3F7F,0080C0E0F0F8FCFE,00
    0103070E0E1E1E,FFFFFFFFE8EBEBEB
400 DATA FFFFFFFF2FE7EAED,0080C0E0B030B
    8B8,1E1E0E0E06070301,0BEBEBEBE8FFFF
    FF,EDEFEFEF2FFFFFFF,B8B8B8B0B0E0C08
    0
410 DATA FF03070F1F3F,FFC0E0F0F8FC,FEFC
    F8F0F0F0F0F0,7F3F1F0F0F0F0F0F,F0F0F
    0F0F0F8FCFE,0F0F0F0F1F3F7F
420 DATA 000000000000030F3F,0000000000C0F
    0FC,0003070F1F3FFFFF,FFFFFFFFFFFFFF
    FF,00C0E0F0F8FCFFFF,FFFF3F1F0F0703
430 DATA FFFFFCF8F0E0C0,3F0F03,FCF0C0
440 FOR I=88 TO 152
450 READ A$
460 CALL CHAR(I,A$)
470 NEXT I
480 FOR I=1 TO 7
490 CALL COLOR(I,1,1)
500 NEXT I
510 CALL COLOR(8,9,16)
520 CALL COLOR(9,9,16)
530 CALL COLOR(10,9,16)
540 CALL COLOR(11,2,16)
550 CALL COLOR(12,5,16)
560 CALL COLOR(13,13,16)
570 CALL COLOR(14,13,16)
580 CALL COLOR(15,11,16)
590 CALL COLOR(16,11,16)
600 CALL CHAR(33,"FFFF0000000000FF")
610 CALL CHAR(35,"C1C1C1C1C1C1C1C1")
620 CALL CHAR(36,"8383838383838383")
630 CALL CHAR(37,"FF0000000000FFFF")
640 CALL CHAR(39,"000000FFFF000000")
650 CALL CHAR(40,"FFFFE0D0C8C4C2C1")
660 CALL CHAR(41,"C1C2C4C8D0E0FFFF")
670 CALL CHAR(43,"834323130B07FFFF")
680 CALL CHAR(47,"FFFF070B13234383")
690 CALL CHAR(59,"FFFFFFFF1C6B6B1B")
700 CALL CHAR(60,"FFFFFFFFC75B5B57")
710 CALL CHAR(62,"686B1BFFFFFFFFFF")
720 CALL CHAR(63,"575B5DFFFFFFFFFF")
730 CALL CHAR(64,"0044442810101010")
740 PRINT TAB(6);"(!!!!/(!!!!/(!!!!/"
750 FOR I=1 TO 4
760 PRINT TAB(6);"#    $#    $#    $"
770 NEXT I
780 PRINT TAB(6);")%%%%+)%%%%+)%%%%+":T
    AB(4);"(!!!!!!!!!!!!!!!!!!!!/"
790 PRINT TAB(4);"#MATCH:           PTS
    $":TAB(4);"#*ALL THREE.......100$":
    TAB(4);"#*FIRST TWO:          $"
800 REM USE @ FOR LETTER Y
810 PRINT TAB(4);"#   ACE,SEVEN,BAR.. 50
    $":TAB(4);"#   BELL,CHR@,LMN.. 30$":
    TAB(4);"#*FIRST & THIRD... 15$"
```



TI-99/4A

```
820 PRINT TAB(4);"#*LAST TWO.........10
    $":TAB(4);"#*NO MATCH..........-10$":
    TAB(4);"#,,,,,,,,,,,,,,,,,,,,,,,$"
830 PRINT TAB(4);"#POINTS SCORED:
    $":TAB(4);"#";TAB(25);"$":TAB(4);"#
    TOTAL  POINTS:          $"
840 REM  USE @ FOR LETTER Y
850 PRINT TAB(4);"#";TAB(25);"$":TAB(4)
    ;"#NO. OF  PLA@S:        $":TAB(4);"#
    ";TAB(25);"$"
860 REM  USE @ FOR LETTER Y
870 PRINT TAB(4);")%%%%%%%%%%%%%%%%%%%%
    +":"P=PLA@, N=NEW PLA@ER, S=STOP";
880 CALL SCREEN(11)
890 FOR I=1 TO 5
900 CALL VCHAR(1,I,147,23)
910 CALL VCHAR(1,I+27,147,23)
920 NEXT I
930 CALL VCHAR(1,6,147,6)
940 CALL VCHAR(1,7,147,6)
950 CALL VCHAR(1,26,147,6)
960 CALL VCHAR(1,27,147,6)
970 CALL VCHAR(10,29,36,10)
980 CALL VCHAR(10,30,118,12)
990 CALL HCHAR(20,28,37)
1000 CALL HCHAR(21,28,118,2)
1010 CALL HCHAR(20,29,43)
1020 CALL HCHAR(8,28,150)
1030 CALL HCHAR(8,29,109,2)
1040 CALL HCHAR(8,31,149)
1050 CALL HCHAR(9,28,148)
1060 CALL HCHAR(9,29,109,2)
1070 CALL HCHAR(9,31,146)
1080 FOR I=1 TO 7
1090 CALL COLOR(I,2,8)
1100 NEXT I
1110 GOTO 1940
1120 TOTAL=0
1130 PLAYS=0
1140 CALL HCHAR(19,23,32,4)
1150 CALL HCHAR(21,23,32,4)
1160 RANDOMIZE
1170 FOR J=9 TO 21 STEP 6
1180 FOR I=2 TO 5
1190 CALL HCHAR(I,J,32,4)
1200 NEXT I
1210 NEXT J
1220 CALL HCHAR(17,23,32,4)
1230 PLAYS=PLAYS+1
1240 C=9
1250 FOR I=1 TO 3
1260 SYM(I)=INT(6*RND)+1
1270 ON SYM(I)GOSUB 2010,2050,2090,2130,
     2170,2210
1280 C=C+6
1290 NEXT I
1300 REM   POINTS
1310 IF SYM(1)<>SYM(2)THEN 1330
1320 IF SYM(2)=SYM(3)THEN 1710 ELSE 1350
1330 IF SYM(1)=SYM(3)THEN 1520
1340 IF SYM(2)=SYM(3)THEN 1650 ELSE 1590
1350 IF SYM(1)/2<>INT(SYM(1)/2)THEN 1440
1360 TOTAL=TOTAL+50
1370 FOR I=1 TO 3
1380 CALL SOUND(200,550,2)
1390 CALL SOUND(200,330,2)
1400 NEXT I
```

TI-99/4A

```
1410 CALL HCHAR(17,23,53)
1420 CALL HCHAR(17,24,48)
1430 GOTO 1870
1440 TOTAL=TOTAL+30
1450 CALL SOUND(200,440,2)
1460 CALL SOUND(200,550,2)
1470 CALL SOUND(200,660,2)
1480 CALL SOUND(200,440,2)
1490 CALL HCHAR(17,23,51)
1500 CALL HCHAR(17,24,48)
1510 GOTO 1870
1520 TOTAL=TOTAL+15
1530 CALL SOUND(150,440,2)
1540 CALL SOUND(150,550,2)
1550 CALL SOUND(150,660,2)
1560 CALL HCHAR(17,23,49)
1570 CALL HCHAR(17,24,53)
1580 GOTO 1870
1590 TOTAL=TOTAL-10
1600 CALL SOUND(550,110,0)
1610 CALL HCHAR(17,23,45)
1620 CALL HCHAR(17,24,49)
1630 CALL HCHAR(17,25,48)
1640 GOTO 1870
1650 TOTAL=TOTAL+10
1660 CALL SOUND(200,660,2)
1670 CALL SOUND(200,770,2)
1680 CALL HCHAR(17,23,49)
1690 CALL HCHAR(17,24,48)
1700 GOTO 1870
1710 TOTAL=TOTAL+100
1720 FOR I=1 TO 5
1730 CALL SOUND(400,800,1)
1740 CALL SOUND(400,550,4)
1750 NEXT I
1760 FOR I=1 TO 12
1770 CALL COLOR(1,2,14)
1780 CALL COLOR(1,2,7)
1790 CALL COLOR(1,2,4)
1800 CALL COLOR(1,2,11)
1810 CALL COLOR(1,2,16)
1820 CALL COLOR(1,2,8)
1830 NEXT I
1840 CALL HCHAR(17,23,49)
1850 CALL HCHAR(17,24,48)
1860 CALL HCHAR(17,25,48)
1870 CALL HCHAR(19,23,32,4)
1880 FOR I=1 TO LEN(STR$(TOTAL))
1890 CALL HCHAR(19,22+I,ASC(SEG$(STR$(TO
     TAL),I,1)))
1900 NEXT I
1910 FOR I=1 TO LEN(STR$(PLAYS))
1920 CALL HCHAR(21,22+I,ASC(SEG$(STR$(PL
     AYS),I,1)))
1930 NEXT I
1940 CALL KEY(3,KEY,STATUS)
1950 IF STATUS=0 THEN 1940
1960 IF KEY=80 THEN 1160
1970 IF KEY=78 THEN 1120
1980 IF KEY<>83 THEN 1940
1990 CALL CLEAR
2000 END
2010 REM    CHERRY
2020 RESTORE 2040
2030 GOTO 2240
2040 DATA 88,89,114,115,90,91,116,117,92
     ,91,91,93,94,95,96,97
```

Continued on p. 192

Index to Advertisers

Slots . . . from p. 191

TI-99/4A

```
2050 REM      SEVEN
2060 RESTORE 2080
2070 GOTO 2240
2080 DATA 98,99,99,100,101,102,103,104,1
     09,105,106,109,107,108,109,109
2090 REM      LEMON
2100 RESTORE 2120
2110 GOTO 2240
2120 DATA 109,144,145,109,146,147,147,14
     8,149,147,147,150,109,151,152,109
2130 REM      ACE
2140 RESTORE 2160
2150 GOTO 2240
2160 DATA 109,128,129,109,130,131,132,13
     3,134,135,136,137,109,138,139,109
2170 REM     BELL
2180 RESTORE 2200
```

TI-99/4A

```
2190 GOTO 2240
2200 DATA 109,120,121,109,109,122,122,10
     9,109,122,122,109,123,124,125,126
2210 REM     BAR
2220 RESTORE 2230
2230 DATA 109,109,109,109,112,59,60,113,
     112,62,63,113,109,109,109,109
2240 FOR R=2 TO 5
2250 READ R1,R2,R3,R4
2260 CALL HCHAR(R,C,R1)
2270 CALL HCHAR(R,C+1,R2)
2280 CALL HCHAR(R,C+2,R3)
2290 CALL HCHAR(R,C+3,R4)
2300 NEXT R
2310 CALL SOUND(30,400,1)
2320 CALL SOUND(30,800,1)
2330 RETURN
```

HCM

however, the appropriate software for the other machines does not exist, although it is possible to "fix" the IBM PC for accuracy by compiling the BASIC program using the BASIC compiler and running the compiled program from DOS. [On the IBM PCjr, Cartridge BASIC provides 16-digit accuracy.—Ed.]

Necessary Precision

Just how much accuracy is required of the home computer for scientific and engineering work? This is a difficult question because the answer depends on the type of computations that have to be done. Ten to fifteen years ago, I used a slide rule for all my personal computing. But I didn't use the slide rule to integrate rocket trajectories or invert 6-order matrices. These computations were saved for the IBM 704 or the Univac 1103. Then along came the HP 35, SR 52, HP 65, TI 59, HP 42, etc. These hand-held calculators changed the work habits of many scientists and engineers who came to expect the same accuracy from them as they got from the large IBM or Univac scientific machines. But these calculators were slow and limited in their capacity to hold large programs. Hence, they did not lend themselves to numerical integration or Monte Carlo simulation using random numbers. Now we have home computers—and we expect them to do nearly everything that the large computers do.

"Most computer users do not require any more accuracy than 6 or 7 significant figures."

Most scientific constants are known to 6 significant figures, and this sets the minimum standard for accuracy in computation. There is also some justification for extending this standard of precision to 8 digits since some physical measurements (e.g., angular velocity of the earth) are known to this accuracy. Computational errors continually arise from incorrectly entered data or faulty program algorithms. In addition to these obvious sources of error, two other conditions can lead to loss of accuracy in computing. These situations call for a standard of accuracy higher than 6 or 8 significant figures.

"Accurate" Numbers

The first problem results from the finite size of the computer's *accumulator* (a part of the Central Processing Unit where numbers are processed). The accumulator is only 8 decimal figures or 5 bytes (40 binary bits). Accuracy is lost whenever a large accurate number is added to (or subtracted from) a small accurate number in an accumulator. (An accurate number uses all the computer's significant digits.) Accuracy is also lost when two accurate numbers that are nearly equal are subtracted. As an example, try the following computation on your calculator or home computer:

(1E8+1.23456789) − 1E8 = ?

IBM's response is 1.2345678, TI's is 1.2346, and Apple and Commodore return 1.25. If the result you get is 1.23456, then your computer's addition/subtraction is very accurate. If you get zero, you should be aware that you must program your calculations to minimize this loss in accuracy. In some cases, it is possible to evaluate the above result using a power series expansion that saves several terms which can be accurately evaluated. Fortunately, this problem arises only during addition and subtraction and is of little concern when the input numbers are of the same magnitude and precision.

Byte Noise

Extended "number crunching" can also lead to a loss of accuracy in computing. Errors in the internal math functions (SQR, SIN, EXP, LOG, ATN, etc.) and the accumulation of inaccurate numbers lead to computation "noise." In estimating the effect of noise on computation accuracy we shall assume that the two noise sources are approximately equal. Then the computation error may be regarded as a noise error, which increases as the square root of the number of operations, N, and:

$$E_f = N \times E_I$$

where E_f and E_I denote final and initial errors.

You can get a feeling for the number of computer operations involved by considering a typical ten-minute powered flight trajectory integration. This involves about 100 computations for the integration step of 6 variables and an additional 100 computations for printout. If the step size is 2 sec, then approximately 60,000 operations or computations (+, −, ×, ÷, SQR, SIN, ATN, etc.) are required.

Now, if the computer has 8 significant figures and the interval math functions are evaluated to this accuracy, then the likely initial noise error is:

$$E_I = 10^{-8}$$

If we now run the trajectory integration program of 60,000 operations, then the error in the result E_f is estimated as:

$$E_f \approx 2.4 \times 10^{-6}$$

This computation assumes that all numbers are about the same size so that we don't have the loss in precision of the first type discussed.

The TI-99/4A can perform 1 million computations (operations) and still provide 10-digit accuracy. Many of the available computers will provide only 6-digit accuracy after 100 computations. But even with the TI-99/4A, it is occasionally necessary to program within the limits of the central processor. For example, evaluate:

$$B(n,K) = \left(\frac{1}{2^n}\right)\left(\frac{n!}{(n-K)!K!}\right)$$

for n = 300, K = 200

The term n! is pronounced "n factorial," and it is equal to:

$$1 \times 2 \times 3 \times \ldots \times (n-1) \times (n).$$

A TI computer is able to evaluate n! only up to n=69. IBM, Commodore and Apple can only go to n=33.

"Don't be so awed by your machine that you accept its every response with unquestioning trust simply because it is a computer."

Most computer users do not require any more accuracy than 6 or 7 significant figures. This is sufficient for most business, real estate, word processing or spreadsheet applications. Accountants may require 9 or 10 significant figures along with round-off protection because an error of even one cent is not acceptable. Serious engineering and scientific calculations clearly need more than 6 or 7 significant figures and round-off protection.

This discussion should motivate you to re-evaluate the degrees of accuracy you require and to investigate the capabilities of your own machines. Ideally, you should face these questions before you purchase your computer; otherwise you may find yourself constantly burdened with special programming techniques to avoid losses in accuracy. Of course other factors, such as sound capability, graphics features, and cost, must also be considered when selecting a computer. Just remember that all computers have limitations. Don't be so awed by your machine that you accept its every response with unquestioning trust simply because it is a computer. HCM

Future Is Now . . . from p. 167

The Future

As this hardware/software combination catches on in the marketplace, we can soon expect to see new software designed for Apples with the *SuperSprite* board in slot 7. These products will be state-of-the-art in every way and equal to or better than the graphics you see on any system.

One of the truly appealing features of *SuperSprite* is that the old Apple graphics do not become totally obsolete. They are compatible with the improved system. This new development in Apple graphics has accomplished a quantum leap to enhance an already fine system. Finally, we have a way to update, rather than replace, our Apple graphics and end up with arcade-quality results, with very natural-sounding speech thrown into the bargain. HCM

Introducing
The Chambers Of Vocab.

Goony, Zany, Goofy, Brainy The Tricky,

Word Maze Game.

The Chambers of Vocab™ is a word game where all the action takes place in a monstrous maze. It's got great color graphics—as good as any game. But with a difference. The Chambers of Vocab is more than just adventure and entertainment.

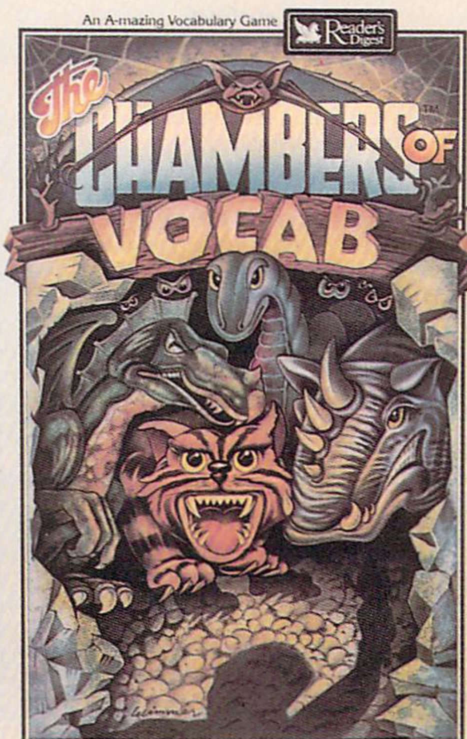
It will improve your vocabulary while you try to escape from the monsters of the maze by avoiding mistakes. Once you've tamed the Gondras in the first maze, you can squirm past Bliterates, Sylamanders and Chimeras in the second, third and fourth. They keep getting tougher.

And when the whole program's over, it's not over at all. The disk has 300 words. When you're finished with them, you can add your own words to create new challenges for your new skills.

The Chambers of Vocab is a brain game that grows as you do. A veritable labyrinthian labor with longevity! If you don't know what that means now, become a maze master and you will.

See your local dealer, or Call Toll-Free: 800/431-8800. (NY, AK, HI: 914/241-5727).

Warranty information available upon request by writing to: Reader's Digest Services, Inc., Microcomputer Software Division, Pleasantville, New York 10570.



This disk is designed for use on the APPLE II* or APPLE II Plus* with 48K and 1 disk drive

Reader's Digest®

HOME COMPUTER MAGAZINE'S SENSATIONAL SOFTWARE GIVEAWAY

YOUR CHOICE OF THE PROGRAMS FROM THIS ISSUE AT THE UNBELIEVABLE LOW PRICE OF ONLY $3.95!

PROOF OF PURCHASE NECESSARY

To participate in our monthly Software Giveaway, you need to be a bona-fide purchaser of *Home Computer Magazine* and follow the directions on this form.

PLEASE PRINT

NAME ______________________

ADDRESS ______________________

CITY ______________ STATE ______ ZIP ______

Have you taken advantage of our Software Giveaway before? ☐ Yes ☐ No

Indicate your choice of media: ☐ ON TAPE™ ☐ ON DISK™

Indicate which machine the media is for: (Pick one)
☐ Apple II Series ☐ C-64 ☐ VIC-20 ☐ IBM PC ☐ PCjr ☐ TI-99/4A

Fill in this magazine's *Volume No.* ____ and *Issue No.* ____. You will receive all programs (which have versions for your selected machine) whose listings appear in this issue. Please also list, in order of preference, these same programs—helping us determine future offerings.

1. ______________ 4. ______________
2. ______________ 5. ______________
3. ______________ 6. ______________

IMPORTANT: Complete the questionnaire on the reverse side. Tear out this complete page and enclose in an envelope along with $3.95. Payment must be made by check, money order or VISA/MasterCard. Proof of purchase (subscriber label/number, sales receipt or reasonable facsimile thereof) must also accompany this form.

☐ Questionnaire is completed.
☐ Enclosed is check or money order for $3.95.
☐ Enclosed is my proof of purchase.
☐ Bill my ☐ VISA ☐ MasterCard

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
Account No.

OFFER GOOD ONLY IN USA
(Canada & Mexico please inquire)

Signature ______________ ______________ Exp. Date

Mail this page to:
Software Giveaway
Home Computer Magazine
P.O. Box 5537
Eugene, OR 97405

TO PROPERLY HANDLE THE VOLUME OF SOFTWARE REQUESTS, ORDERS MUST BE SENT IN ON THIS FORM, NO PHOTO COPIES. SORRY, WE CANNOT ACCEPT TELEPHONE ORDERS FOR THIS SERVICE.

SG/2-84

ON TAPE / ON DISK—$3.95 each issue.

ABSOLUTELY FREE!
2-6 MONTHS OF MAGAZINE SOFTWARE ON TAPE™ OR ON DISK™ ARE FREE WITH YOUR PAID SUBSCRIPTION

With a 1-year subscription to *Home Computer Magazine* you get 2 FREE months of ON TAPE or ON DISK. Subscribe for 2 years and receive 4 months. And with a 3-year subscription we'll give you 6 full months of this convenient software on cassette tape or floppy disk.

NEW SUBSCRIBER OFFER
(Valid in USA only)

PLEASE PRINT

Name ______________________

Address ______________________

City ______________ State ______ Zip ______

☐ Check or Money Order Enclosed

MUST BE US FUNDS DRAWN ON A US BANK

Bill my ☐ VISA ☐ MasterCard

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐
Account No.

Signature ______________ Exp. Date ______

SATISFACTION GUARANTEED—or the unfilled portion of your subscription will be refunded, less the cost of any premium items you have received.

Indicate your choice of media: ☐ ON TAPE ☐ ON DISK

Indicate which computer media is for: (Pick one)
☐ Apple II Series ☐ C-64 ☐ VIC-20 ☐ IBM PC ☐ PCjr ☐ TI-99/4A

Choose desired subscription term:
1-yr ☐ $25—12 issues + 2 months ON TAPE / ON DISK
2-yr ☐ $45—24 issues + 4 months ON TAPE / ON DISK
3-yr ☐ $63—36 issues + 6 months ON TAPE / ON DISK

Enclose payment or credit card information and mail with completed form to:

Home Computer Magazine
P.O. Box 5537 • Eugene, OR 97405

Or use our TOLL-FREE ORDER LINE*
FOR VISA/MasterCard orders only: 1-800-828-2212
In Oregon, Alaska, Hawaii Tel. (503)485-8796
*Please make reference to this ad when placing phone orders.
Allow 6-8 weeks for your first issue.

SGSO/2-84

HOME COMPUTER magazine™

QUESTIONNAIRE

Complete and mail to: Home Computer Magazine • P.O. Box 5537 • Eugene, Oregon 97405

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+/IIe ☐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 IIe ☐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+/IIe ☐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

SUBSCRIBE TODAY

Please Print **NEW SUBSCRIBER OFFER**

Name ____________

Address ____________

City ____________ State ________ Zip ______

Select your choice of media: ☐ ON TAPE ☐ ON DISK

Indicate which machine the media is for: (Pick one)

☐ Apple II Series ☐ C-64 ☐ VIC-20 ☐ IBM PC ☐ PCjr ☐ TI-99/4A

Choose desired subscription term:

1-yr ☐ $25—12 issues + 2 months ON TAPE / ON DISK

2-yr ☐ $45—24 issues + 4 months ON TAPE / ON DISK

3-yr ☐ $63—36 issues + 6 months ON TAPE / ON DISK

Canada add $7; Foreign Surface add $18 for 1-yr magazine subscription only. Free software not available outside USA.

☐ Check or Money Order Enclosed. Must be in US funds drawn on a US bank.

Bill my ☐ VISA ☐ MasterCard

Account No. ____________ Allow 6-8 weeks for your first issue.

Signature ____________ Exp. Date ______

SATISFACTION GUARANTEED—or the unfilled portion of your subscription will be refunded, less the cost of any premium items you have received.

MSO/2-84

PLACE FIRST CLASS POSTAGE STAMP (CANADIANS PLEASE AFFIX 37¢ POSTAGE

HOME COMPUTER™ magazine
P.O. Box 5537
Eugene, OR 97405

Subscribe Today

SAVE OVER 30%

Off The Single-Copy Price

AND RECEIVE

FREE SOFTWARE TO BOOT!

(See reverse side for details)

PLACE FIRST CLASS POSTAGE STAMP (CANADIANS PLEASE AFFIX 37¢ POSTAGE

HOME COMPUTER™ magazine
P.O. Box 5537
Eugene, OR 97405

Come Discover A Truly Unique Magazine

Inside You will find:

- EDUCATION
- RECREATION
- INFORMATION
- ILLUMINATION

AND ABOVE ALL

- COMPU-PRESTIDIGITATION*

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