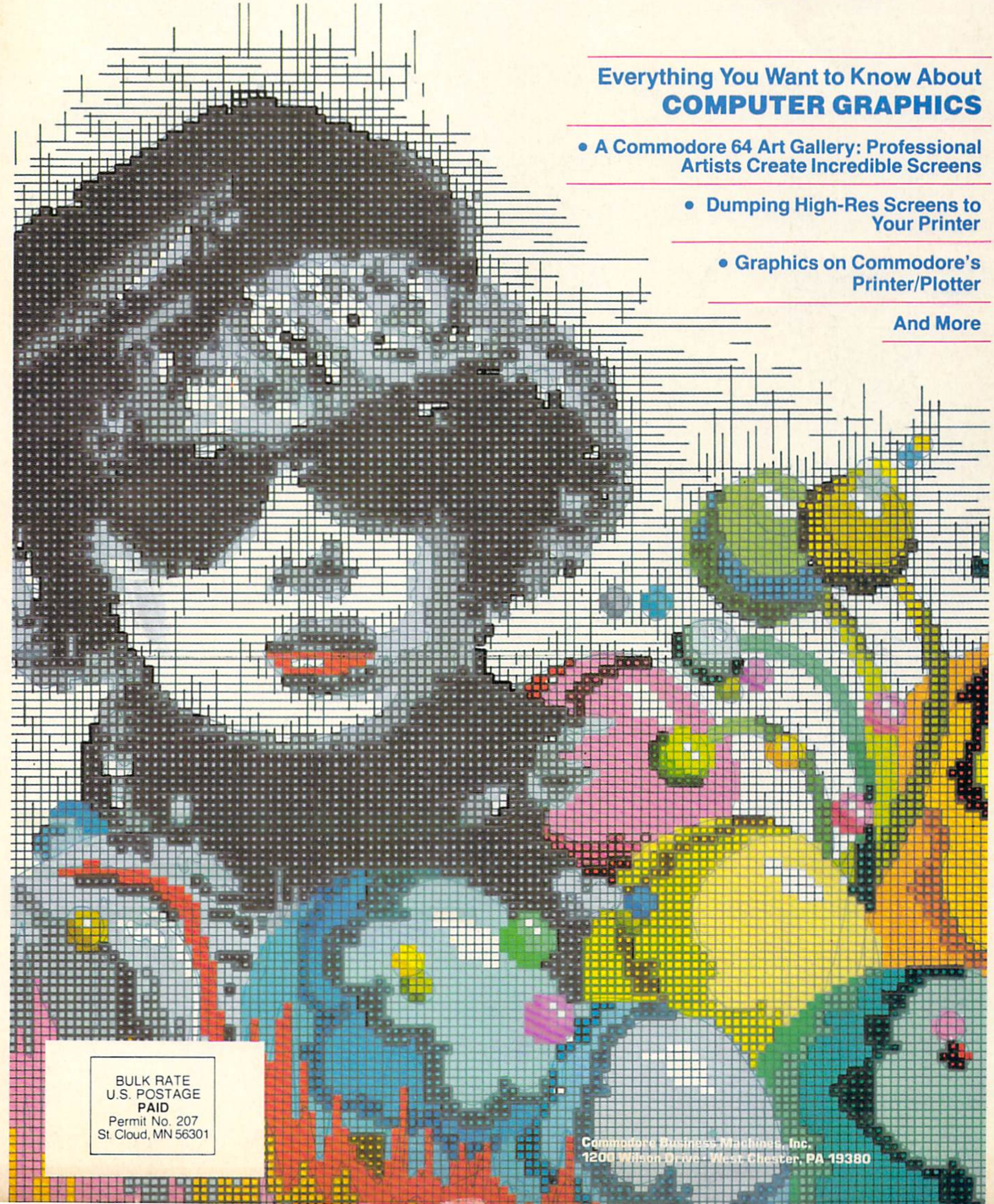
commodore
the microcomputer magazine

Volume 5, Number 1, Issue 28
$2.50 U.S.
$3.50 Canada
ISSN 0744-8724

Everything You Want to Know About
COMPUTER GRAPHICS

- A Commodore 64 Art Gallery: Professional Artists Create Incredible Screens
- Dumping High-Res Screens to Your Printer
- Graphics on Commodore's Printer/Plotter

And More



Commodore Business Machines, Inc.
1200 Wilson Drive · West Chester, PA 19380

COMMODORE BOOKWARE

COMMODORE SOFTWARE ENCYCLOPEDIA
commodore COMPUTER
STIMULATING SIMULATIONS FOR THE VIC
VIC REVEALED
VIC-20 PROGRAMMER'S REFERENCE GUIDE
MASTERING YOUR VIC-20 THROUGH EIGHT BASIC PROJECTS
commodore
VIC GRAPHICS
COMMODORE 64 PROGRAMMER'S REFERENCE GUIDE
VIC GAMES
HAYDEN

Hardware... Software... BOOKWARE

Commodore offers you complete support for your home computer. To complement your Commodore hardware and software, we have a full range of "teach yourself" programming and instructional books. From "create your own graphics and music" to extensive computer language programming, Commodore meets your computer needs.

The Commodore Software Encyclopedia is an indispensable guide for worldwide Commodore software.

From beginner to business professional Commodore Books are valuable additions to your computer library.

commodore
COMPUTERS

First In Quality Software

Be confident because your PowerType Daisywheel correspondence becomes you.

When you can't personally be there, only the clearest typed correspondence should be your substitute. With a PowerType Daisywheel printer your documents look highly professional. And so do you!

PowerType. It's "typewriter friendly." Using a simple drop-in ribbon cassette, it bi-directionally types executive quality correspondence at 18 cps with a print wheel that holds 96 flawless characters.

Designed for personal or business applications, PowerType's carriage accepts paper that ranges from letter to legal size, from fanfold to roll to cut sheet. You can set right and left margins, vertical and horizontal tabs.

Plus, of course, PowerType has both serial and parallel interfaces to enable it to connect to just about any personal or business computer.

So the next time you're going face to face through the mail, rely on PowerType. It will help you make a professional impression. And that's always very becoming.

star™
MICRONICS • INC

THE POWER BEHIND THE PRINTED WORD.

Computer Peripherals Division
P.O. Box 612186, Dallas/Ft. Worth Airport, TX 75261 (214) 456-0052

star
MICRONICS • INC

Dear Mickey,

Believe it or not, this letter is being typed by our new daisywheel! I'm thrilled to announce that the quality of its 96 characters has met our highest expectations.

She's now called PowerType, and she's boxed and ready to go!

As you certainly know, this has been a pet project of mine and it's so exciting to see the actual finished product live up to my imagination. It sounds corny, but I guess I almost feel like a proud father.

Send regards to Alan and John and tell them to make room on their shelves! We'll be sending PowerType and literature early next week.

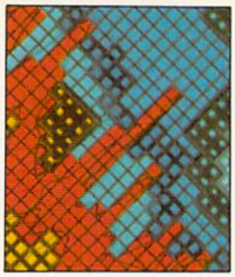
And remember, with PowerType, dot's not the way it goes. (Sorry, I'd better stick to designing.)

All my best,

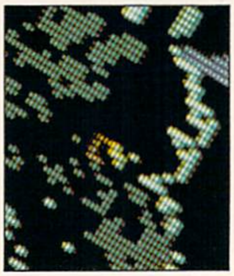
the microcomputer magazine

features

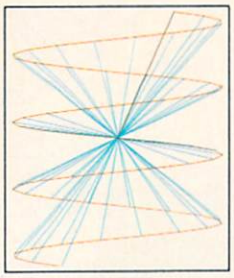
Volume 5, Number 1, Issue 28



Art Gallery



Michaelangelo

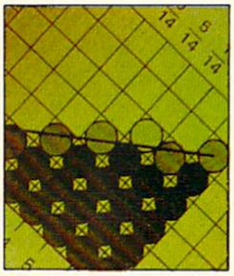


Commodore 1520 Printer/Plotter

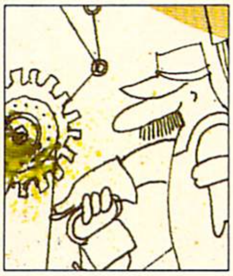
departments

Dumping High-Res Screens

commodore
the microcomputer magazine



Antialiasing Lines and Polygons



Shrinking Data

staff

Director—Publishing
Neil Harris

Editor
Diane LeBold

Technical Editor
Jim Gracely

Staff Writers
David Berezowski
Matt Blais
Brooks Cooley
Andy Finkel
Steve Finkel
Stephen Murri
Howard Rotenberg

Contributing Writers
Gail Austin
Robert W. Baker
Dick Barnes
Elizabeth Deal
Andy Gamble
Donald Hassler
Craig R. Hessel
Eddie Johnson
Stephen S. Leven
Len Lindsay
Rita McCann
Joe Rotello
Ronald Suddendorf
Herbert Swartz
Mark Zimmermann

Technical Staff
Barbara Karpinski
April Koppenhaver
Cyndie Merten
Sarah Meyer
Stephen Murri

Circulation Manager
John O'Brien

Circulation Assistant
Kathy Reigel

Advertising Coordinator
Sharon Steinhofer

Graphic Design
Neumann Greenberg Schlenker
King of Prussia, Pennsylvania

Cover
Greg Purdon
(from a screen by Steve Murri)

Typography
Associates International, Inc.
Wilmington, Delaware

Printing
Volkmuth Printers
St. Cloud, Minnesota

ISBN 0-88731-006-0

VIC 20™, Commodore 64™ and SuperPET™ are trademarks of Commodore Electronics Ltd. PET® is a registered trademark of Commodore Business Machines, Inc. CBM® is a registered trademark of Commodore Electronics Ltd.

Commodore: The Microcomputer Magazine is published by the Computer Systems Division, Commodore Business Machines, Inc., 1200 Wilson Drive, West Chester, Pennsylvania 19380. Copyright 1984© by Commodore Electronics Ltd. All rights reserved. No material may be reprinted without permission. Volume 5, Number 1, Issue 28: February/March 1984.

Subscription Orders: Phone 800-345-8112 (In Pennsylvania 800-662-2444).

The Best Ever in 1984!

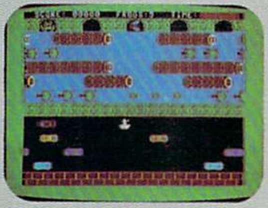
Power/Play, Spring: Look for more from the Commodore Kids, the mysterious Dr. Curriefavor, games, learning at home, and a wealth of programs to type and save.

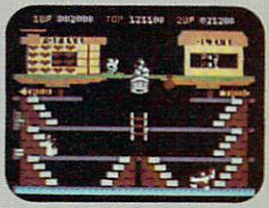
Commodore, Issue 29: Computer Literacy is our feature topic for April/May. How and where to learn about computers and computing, for educators, parents and anyone interested in knowing more about the wonders of this amazing tool. **C**

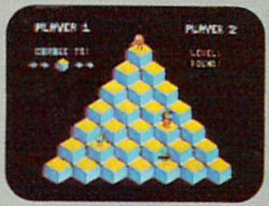
Key to Entering Program Listings

```
"[F1,F2,F3,F4,F5,F6,F7,F8]":F1,F2,F3,F4,
  F5,F6, F7 AND F8
"[POUND]":ENGLISH POUND
"[PI]"PI SYMBOL
"^":UP ARROW
"[HOME]":UNSHIFTED CLR/HOME
"[CLEAR]":SHIFTED CLR/HOME
"[RVS]":REVERSE ON
"[RVOFF]":REVERSE OFF
"[BLACK,WHITE,RED,CYAN,MAGENTA,GREEN,BLUE,
  YELLOW]" THE 8 CTRL KEY COLORS
"[ORANGE,BROWN,L. RED,GRAY 1,GRAY 2,L.
  GREEN,L. BLUE, GRAY 3]":THE 8
  COMMODORE KEY COLORS (ONLY ON THE 64)
GRAPHIC SYMBOLS WILL BE REPRESENTED AS
  EITHER THE LETTERS SHFT (SHIFT KEY) AND
  A KEY:"[SHFT Q,SHFT K,SHFT V,SHFT T,
  SHFT L]"
  OR THE LETTERS CMDR (COMMODORE KEY) AND
  A KEY:"[CMDR Q,CMDR H,CMDR S,CMDR N,
  CMDR O]"
IF A SYMBOL IS REPEATED, THE NUMBER OF
  REPITITIONS WILL BE DIRECTLY AFTER THE
  KEY AND BEFORE THE COMMA:"[SPACE3,
  SHFT S4,CMDR M2]"
```

GIVE YOUR COMMODORE 64™ A LITTLE CHARACTER.







Your Commodore 64 is a great computer system. And one of the things that's great about it is it can play three of the greatest Arcade Action games ever. Frogger,*™ Popeye,® and Q*bert,™ from Parker Brothers.

The award-winning FROGGER is one of the top selling Arcade Action games of all time. With graphics that are nothing less than ribbitting and game play that gets tougher as you get better.

And POPEYE has you running through three screens of non-stop action, where you try to capture Olive Oyl's heart while avoiding untold dangers, including Brutus and the Sea Hag.

As for Q*BERT, he's irresistible. Jumping from cube to cube, trying to avoid an army of nasty critters, he's jumped into the hearts and minds of millions.

Frogger, Popeye, and Q*bert, from Parker Brothers' Arcade Action Series. They make your Commodore 64 computer feel as close to the arcade as you can get.

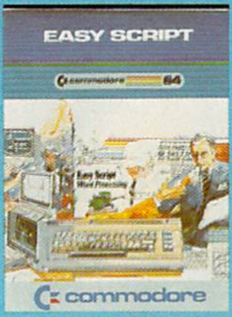
®PARKER BROTHERS

commodore 64

INDISPENSABLE SOFTWARE

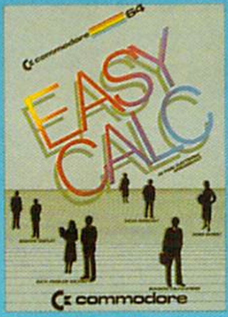
For Your Most Important Computing Needs

Commodore is your best value in practical software—just take a look at the programs shown here—we've got everything from wordprocessing to business accounting, from electronic spreadsheets to computer graphics. Use the Software Selection Guide to find the programs which best meet your needs, then see your Commodore dealer!

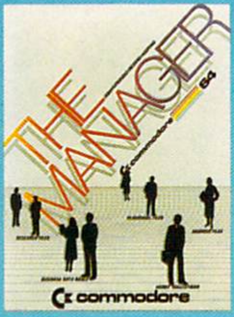


EasyScript 64
Displays 764 lines × 240 characters. Prints to 130 columns. Works with EasySpell 64.

EasySpell 64
20,000 word Master Dictionary and automatic spelling checker. Works with EasyScript 64.



EasyCalc 64
Multiple electronic spreadsheet with color bar graph feature. 63 columns × 254 rows.



The Manager
Sophisticated database system with 4 built-in applications, or design your own. Text, formulas, graphics.

SuperExpander 64
21 special commands. Combine text with high resolution graphics. Music and game sounds.

Easy Finance I—Loan Analysis
12 loan functions. Bar graph forecasting as well as calculation.

Easy Finance II—Basic Investment Analysis
16 stock investment functions. Investment bar graph.

Easy Finance III—Advanced Investment Analysis
16 capital investment functions. Bar graphs.

Easy Finance IV—Business Management
21 business management features. Bar graphs.

Easy Finance V—Statistics and Forecasting
Assess present/future sales trends with 9 statistics and forecasting functions.

Accounts Payable/ Checkwriting
11 functions. Automatic billing. 50 vendors/disk.

Accounts Receivable/Billing
11 billing functions. Printed statements.

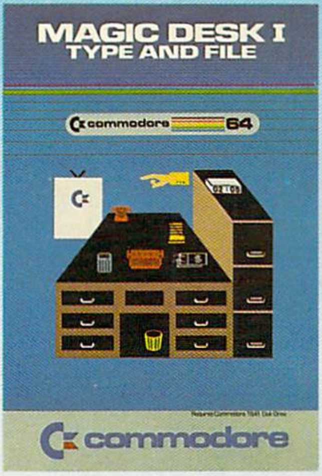
General Ledger
8 general ledger options. Custom income statement, trial balances, reports.

Inventory Management
1000 inventory items. Full reports.

Payroll
24 different payroll functions. Integrated with G/L system.

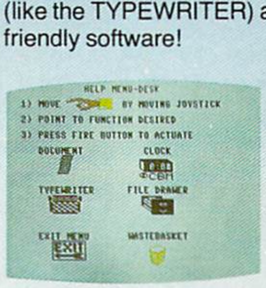
SOFTWARE SELECTION GUIDE

APPLICATION	SOFTWARE
Budget/Calculation	EASYCALC 64
Business Accounting	ACCOUNTS PAYABLE/CHECKWRITING, ACCOUNTS RECEIVABLE/BILLING, GENERAL LEDGER, INVENTORY MANAGEMENT, PAYROLL
Business Management	EASYFINANCE IV—BUSINESS MANAGEMENT
Children's Programming	ZORTEK & THE MICROCHIPS
Cooking/Recipes	MICRO COOKBOOK
Data Base Management	THE MANAGER
Electronic Spreadsheet	EASYCALC 64
Filing/Recordkeeping	MAGIC DESK, THE MANAGER, INVENTORY MANAGEMENT
Financial Investments	EASYFINANCE II—BASIC INVESTMENT ANALYSIS, EASYFINANCE III—ADVANCED INVESTMENT ANALYSIS, FINANCIAL ADVISOR
Graphics/Sound	SUPEREXPANDER 64
Learn Programming	INTRODUCTION TO BASIC—PART 1
Loans/Mortgages	EASYFINANCE I—LOAN ANALYSIS, FINANCIAL ADVISOR
Mailing List	EASYMAIL 64
Music	MUSIC COMPOSER, MUSIC MACHINE
Programming Aids	SUPEREXPANDER 64, SCREEN EDITOR, ASSEMBLER 64
Reference Books	PROGRAMMERS REFERENCE GUIDE, SOFTWARE ENCYCLOPEDIA
Spelling Dictionary	EASYSPELL 64 (for use with EASYSCRIPT 64)
Statistics/Forecasting	EASYFINANCE V—STATISTICS & FORECASTING, EASYFINANCE IV—BUSINESS MANAGEMENT
Teacher's Aids	EASYLESSON/EASYQUIZ, LOGO, PILOT
Telecommunications	VICMODEM, AUTOMODEM, TERM 20/64, RS232 INTERFACE
Wordprocessing	EASYSCRIPT 64, MAGIC DESK, WORD MACHINE/NAME MACHINE



MAGIC DESK I-TYPE & FILE

Only Commodore brings you the magic of MAGIC DESK... the next generation of "user-friendly" software! Imagine using your computer to type, file and edit personal letters and papers—without learning any special commands! All MAGIC DESK commands are PICTURES. Just move the animated hand to the picture of the feature you want to use (like the TYPEWRITER) and you're ready to go. MAGIC DESK is the "ultimate" in friendly software!



```
          HELP MENU-DESK
1) MOVE            BY MOVING JOYSTICK
2) POINT TO FUNCTION DESIRED
3) PRESS FIRE BUTTON TO ACTUATE
   DOCUMENT          CLOCK
                     CBM
   TYPEWRITER      FILE DRAWER

   EXIT MENU       WASTEBASKET
    EXIT
```

Special "Help" Menus

Not only is MAGIC DESK easy to use... it's hard to make a mistake! Just press the COMMODORE key and one of several "help menus" appears to tell you exactly what to do next.

commodore
COMPUTERS

First In Quality Software

commodore 64

THE BEST GAMES IN TOWN

Arcade Action Games

Pinball Spectacular: Real pinball action and thrills. Sound you won't believe. Chutes, lights, bumpers and more.

Supersmash: Raquetball arcade classic. 3 games in 1. Many skill levels keep the challenge alive.

Tooth Invaders: Reviewed by American Dental Association. Arcade action teaches good dental care. Beat D.K. at all 9 play levels.

NEW **Star Post:** Protect the Star Post from waves of invaders. 3 levels of skill. 99 levels of action.

NEW **Avenger:** Destroy attacking aliens with laser cannons. Classic arcade action. Multi-speed attacks.

Jupiter Lander: Space landing simulation. Horizontal/vertical thrust. Soft-land scoring. Wow! animation.

Radar Rat Race: Beat the maze. Eat all the cheese. Beware deadly cats/rats. Cartoon action fun for all ages.

NEW **Lemans:** Multi-obstacle road racing at its best. Arcade action and graphics. Night, water and divided highway hazards.

NEW **Star Ranger:** Fight your way through hoards of space enemies. Avoid asteroids and land safely. Superb graphics and space action.

NEW **Frogmaster:** Unique sports challenge. Train animals to play football and rugby. Over 100 variations. Play against, computer, friend or yourself.

Children's Series

Introduction to Basic I: Simple step-by-step instructions. Modular design. Practical BASIC applications as you learn.

Zortek and the Microchips: Award winning program teaches children BASIC through games, graphics and stories.

Easy Lesson/Easy Quiz: Take the drudgery out of writing tests and quizzes. Answer keys provided. 7 categories per test.

NEW **Number Nabber/Shape Grabber:** 2 Learning games in 1. Build both math and object identification skills. Lively graphic and sound effects.

Visible Solar System: Fly the solar system. Land on planets. Calculate age and weight. Astronomy for home and school. Award winner.

Speed/Bingo Math: 2 games in 1 teach children 4 to 10 basic math skills. Beat the clock or your friends.

SPEED/BINGO MATH
VISIBLE SOLAR SYSTEM
EASY LESSON EASY QUIZ
PROGRAMMING WITH ZORTEK AND THE MICROCHIPS
INTRODUCTION TO BASIC/PART I
NUMBER NABBER SHAPE GRABBER
commodore

Bally Midway

NEW **Gorf:** 4 Space action games in 1. Fly your fighter defeat "The Empire". Multi-skill levels. IT TALKS! (with Magic Voice)

NEW **Wizard of WOR:** Fight your way through 30+ mazes. Defeat the Wizard and Worriors. Multi-skill. IT TALKS! (with Magic Voice.) Award winning conversion.

Seawolf: The classic battle at sea. Destroy PT Boats and Destroyers. Great graphics and sound.

Omega Race: Fast space race action. Many skill levels. Avoid deadly mines as you eliminate droid forces.

Clowns: Amazing action under the "Big Top". Help clowns "pop" balloons. Colorful acrobatics. Fun for all.

Kickman: Ride the unicycle and catch falling objects. Multi-skill levels. Tuneful sound. Watch out! Don't fall!

NEW **Blueprint:** Help J.J. build the "Ammo Machine". Parts are stored in a colorful maze of houses. Multi-skill and difficulty levels.

NEW **Lazarian:** 4 different screens. Multi-skill level space action. Rescue, evade obstacles and destroy a one-eyed leviathan.

Adventure Games

Zork I: Fantasy adventure in a dungeon. Find all the treasure and escape alive.

Zork II: This dungeon adventure dares you to find treasure and secret places and still survive.

Zork III: The ultimate dungeon test. Discover the Dungeon Master's secret purpose and come out alive.

Suspended: Awake in 500 years. Solve varied real and original puzzles to save the planet from total destruction.

Starcross: Travel through the mystery ship. Meet aliens friend and foe. Face the challenge of your destiny. Map of galaxy included.

Deadline: Find the murderer and solve the mystery all in 12 hours. Inspector casebook and evidence included.

Music Series

Music Machine: Play piano or organ melodies and percussion rhythms together. Music staff shows notes on screen. Vibrato, tempo and pitch controls.

Music Composer: Create, play and save your tunes easily. Simulates up to 9 instruments. Notes appear on screen. Play your keyboard like a piano.

commodore
COMPUTERS

First In Quality Software

commodore

VIC 20

SUPER SOFTWARE SAVINGS

Bally Midway

Gorf: 4 Space action games in 1. Fly your fighter defeat "The Empire". Multi-skill level.

Seawolf: The classic battle at sea. Destroy PT Boats and Destroyers. Great graphics and sound.

Omega Race: Fast space race action. Many skill levels. Avoid deadly mines as you eliminate droid forces.

Clowns: Amazing action under the "Big Top". Help clowns "pop" balloons.

Lifestyle Series

Quizmaster: Write and give your own quizzes. Teach, revise, test and entertain.

Know Your Child's I.Q.: 3 Comprehensive tests. 100 questions. Auto and tamperproof scoring. Improve school test performance.

Know Your Own I.Q.: 4 I.Q. tests. 160 problems. Auto and tamperproof scoring. For hours of entertainment.

Know Your Personality: 3 In-depth personality tests. 450 questions. Auto scoring. Find your friends true feelings. For entertainment only.

Robert Carrier's Menu Planner: 120 meals and 20 wines start your menu data-base. Add your own recipes.

Children's Games

The Sky is Falling: Pre-school and elementary age children help Chicken Little. Builds hand-eye coordination.

Mole Attack: Bop the nasty moles as they stick heads out of burrows. Cartoon graphics. Multi-speed action.

Home Babysitter: Building blocks teach the alphabet. Common objects teach numbers to 20. Plus funny face maker.

Visible Solar System: Fly the solar system. Land on planets. Calculate age and weight. Astronomy for home and school. Award winner.

Speed/Bingo Math: 2 games in 1 teach children 4 to 10 basic math skills.

Business and Financial

Personal Finance: Four programs in one. Track expenses. Spending analysis. Budgets and deductibles.

Simplicalc: Electronic spreadsheet. 1200 entries. Design/repeat formulas and worksheets.

VIC Writer: Wordprocessing made simple. From 45 lines unexpanded to 1207 lines of text with 16K RAM PACK.

Money Decisions I: 7 Loan analysis functions. Principle, regular/last payment. Balance. Time period. Interest. Variable rate loan.

Money Decisions II: 9 investment functions. Future/Initial/Minimum Investment. Regular deposit/withdraw. Interest. Annuity Continuous compounding.

Educational Programs

Introduction to Basic I & II: Simple step-by-step instructions. Practical BASIC applications.

Zortek and the Microchips: Award winning program teaches children BASIC through games, graphics and stories.

Waterloo Basic: The original course in VIC BASIC.

Chopper Math: Challenging helicopter landing game that teaches math basics.

Easy Type: Learn touch-typing the easy way.

VIC-20
AN INTRODUCTION TO BASIC - PART 2
WATERLOO STRUCTURED BASIC
VIC 20
ZORTEK AND THE MICROCHIPS
commodore

Adventure Games

Adventureland: Fantasy adventures challenge you to get all the treasure and escape alive.

Pirate Cove: Find the long lost treasure of pirate John Silver. Uncover clues while battling foes.

Atomic Mission: Save the nuclear powerplant from destruction. Piece clues together—solve the mystery.

The Count: Make your way through the dungeon, collect treasure and kill Count Dracula.

Voodoo Castle: Find the Count of Monte Cristo and remove the deadly curse.

Arcade Action Games

VIC Avenger: Destroy attacking aliens with laser cannons. Classic arcade action. Multi-speed attacks.

Super Alien: Trapped in an alien maze, your only defense is an alien buster. Hi-speed action.

Superslot: Vegas and Atlantic City casinos come home. Real slot machine action, graphics and sound.

Jupiter Lander: Space landing simulation. Horizontal/vertical thrust. Soft-land scoring.

Draw Poker: Casino style action. Betting. Sound effects.

Road Race: Night driving challenges you to the max. 4-speed shift. Stay on course. Don't overheat.

Radar Rat Race: Beat the maze. Eat all the cheese. Beware deadly cats/rats.

Raid on Ft. Knox: Sneak gold bars past deadly panthers and back to the hideout before time is up.

Pinball Spectacular: Space action and pinball thrills combined. Lights, bumpers, and special skill bonuses.

Sargon II Chess: Challenging chess strategy classic. Multi-skill levels from beginner to advanced.

Supersmash: Raquetball arcade classic. 3 games in 1. Many skill levels keep the challenge alive.

Cosmic Cruncher: Make your way through the Milky Way. 11 levels of play. Over 300 color/maze combinations.

Money Wars: Grab the money and run. 3 brick barricades are your protection as you dodge deadly bullets.

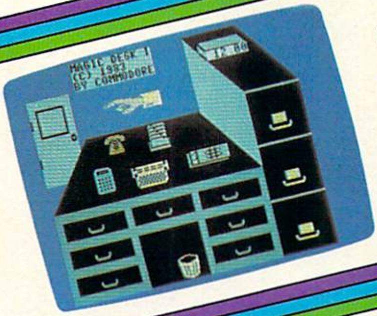
Tooth Invaders: Arcade action teaches good dental care. Beat D.K. at all 9 play levels.

Star Post: Protect the Star Post from waves of invaders. 3 levels of skill. 99 levels of action.

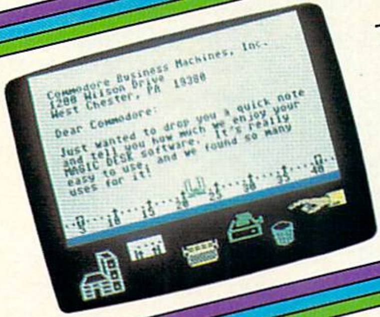
commodore
COMPUTERS

First In Quality Software

commodore 64
MAGIC DESK I
TYPE AND FILE
1200
$
Only from Commodore—
the excitement
and simplicity of
Magic Desk



Only Commodore brings you the magic of MAGIC DESK . . . the next generation of "user friendly" software! Imagine using your computer to type, file and edit personal letters and papers *without learning any special commands!* All MAGIC DESK commands are PICTURES. Just move the animated hand to the picture of the feature you want to use (like the TYPEWRITER) and you're ready to go.

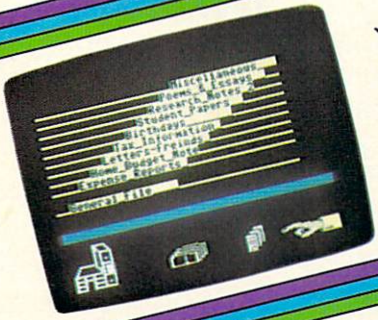


Commodore Business Machines, Inc.
1200 Wilson Drive
West Chester, PA 19380

Dear Commodore:

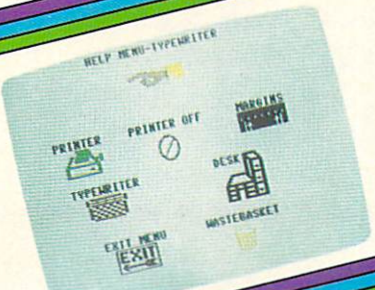
Just wanted to drop you a quick note and tell you how much we enjoy your MAGIC DESK software. It's really easy to use, and we found so many uses for it!

The MAGIC DESK Typewriter works just like a real ELECTRIC TYPEWRITER . . . and it's COMPUTERIZED. All the filing is *electronic.* Excellent sound effects and screen animation make typing fun, whether you're typing letters, reports or memos . . . and the built-in filing feature makes MAGIC DESK useful for keeping names and addresses, home inventory lists, insurance information and more.

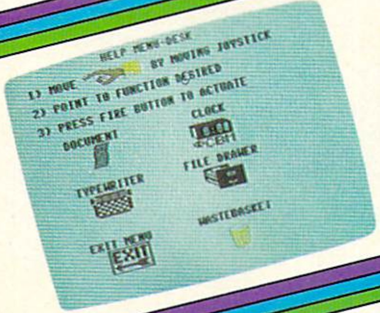


Miscellaneous
Poems & Essays
Research Notes 2
Research Papers
Student Papers
Birthdays
Tax Information
Letters-freinds
Home Budget Note
Home Reports
Expense Reports
General file

Your COMMODORE 64, COMMODORE DISK DRIVE and MAGIC DESK are an unbeatable combination. Filing operations are automatically linked to your Commodore disk drive—but you don't have to know any commands—just "file" the pages you type in the file cabinet and your text is automatically saved on diskette. There are 3 file drawers with 10 file folders in each drawer and 10 pages in each folder.



To PRINT a page you've typed, just "point" at the picture of the printer and your pages are automatically printed on your COMMODORE PRINTER or PRINTER/PLOTTER. If you want to erase what you've typed, the WASTEBASKET under the desk lets you "throw away" pages. There's even a DIGITAL CLOCK which helps you keep track of time while you're typing.



HELP MENU-DESK
1) MOVE BY MOVING JOYSTICK
2) POINT TO FUNCTION DESIRED
3) PRESS FIRE BUTTON TO ACTUATE

Not only is MAGIC DESK easy to use . . . it's hard to make a mistake! Just press the COMMODORE key and one of several "help" menus" appears to tell you exactly what to do next. Special messages show you how the various picture commands work and help you when you make a mistake. Help messages also show you how to use the printer, filing cabinet, digital clock and wastebasket.

commodore
COMPUTERS

First In Quality Software

But Is It Art?

This issue is devoted to a topic that has everyone around here very excited—computer graphics. I remember the days when the words "computer" and "art" repelled each other like the same poles of two magnets. There was just no way to bring them together. We all know that's changed, but to see just how much it's changed take a look at our Commodore 64 Art Gallery on page 34. (My special thanks to Steve Murri, who collected and photographed those screens.)

Of course, it's pretty easy for us to get programming-type information about graphics, since programming is what people do around here at Commodore. That's why we were glad when Albuquerque artist Eddie Johnson said he'd like to talk about computers as a medium of expression—from the artist's point of view. I think you'll find his perspective interesting and his writing enjoyable.

For all our readers who have been patiently waiting for more information on PET/CBM computers, we're finally starting to regain our footing. I have to admit that the great popularity of the Commodore 64 temporarily knocked us a little off balance, but now we're making a comeback thanks to people like Joe Rotello, who appears in this issue with an article on creating windows on your 8032 screen, as well as more of his "PETSpeed Tips". We'll keep working on maintaining that crucial balance among our various computers, so no one gets left out. Meanwhile, keep those letters and suggestions for articles coming in. They're a great help to us.

As we begin 1984, I thought some of you might be interested in the history of Commodore's magazines. We started about four years ago as the *PET User Club Newsletter,* were known briefly as *Interface,* and then finally settled into the somewhat unwieldy title *Commodore: The Microcomputer Magazine* in mid-1981, under the editorship of my stodgy pal Paul Fleming. (*Power/Play* didn't come on the scene until mid-1982.) I remember what a big decision it was to add a second color to our pages, which up until the middle of 1982 had been strictly black and white. And look at us now.

We owe it all to you. Thanks. **C**

—Diane LeBold
Editor

Improving on Trace

To the Editor:

After typing in the listing included in the article on debugging in Volume 4, Number 3, Issue 24 I was pleased that it worked as well as it did. There was only one flaw that made it hard to deal with long programs. The problem is that at the default speed long commands were hard to follow but if a slower speed was POKEd in, the process of tracing was unbearably slow. As I am learning to program in machine language I decided to change the program to use F7 to move along in the trace. The listing below is how I solved the problem.

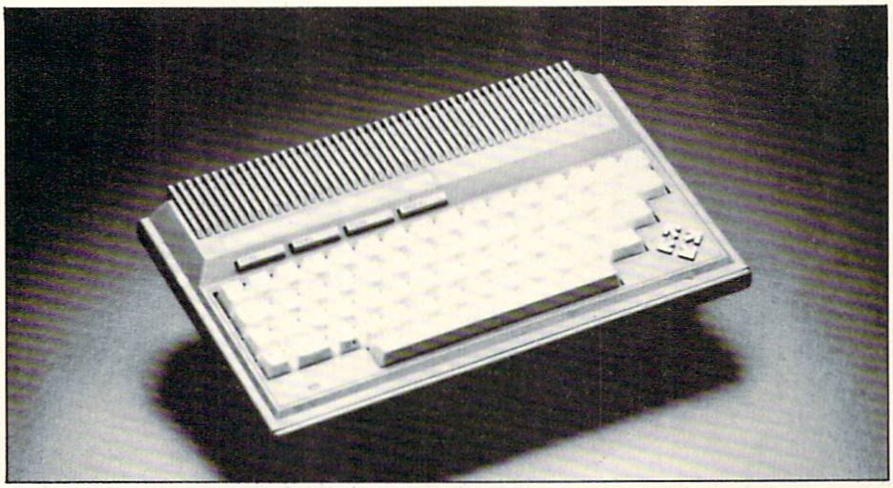
In using this change all the reader has to do is add this listing to the original listing and the changes will be made when the program is run. When tracing a program use the SYS command to enable the program, then hit F7 to start and continue the program at your own pace. If your program has an INPUT statement remember to complete it with a RETURN before hitting F7 and if it has a GET statement with a loop that waits for a key to be pressed, then hit F7 and the correct key for your program after the IF $=" part of the command appears at the top of the screen.

I hope that this will make the program more useful for other readers as it has for me. **C**

Laurie Orr
Flagstaff, Arizona

```
530 B=52776
540 FORP=BTOB+14
550 READX:POKEP,X
560 NEXTP
570 DATA 32,228,255,
    201,136,208,249,
    165,00,170,168,
    234,234,234,234
580 REM ADDITIONS ARE
    LINES 530 TO 570
    WITH ORIGINAL 530
    REMO VED"
600 END
```

Commodore Introduces New Computer at CES



Commodore introduced an exciting new home computer at the Consumer Electronics Show in Las Vegas in January. Dubbed the Commodore 264, the new computer is aimed at a broad market, including business, education and the home. The 264's many built-in features make it a truly exceptional buy for any user. They include:

- 64K RAM (60K available for BASIC programming)
- Full typewriter-style keyboard Optional built-in software
- Screen window capability
- HELP key
- Eight programmed, reprogrammable function keys
- Four separate cursor keys
- Compatibility with most Commodore 64 and VIC 20 peripherals
- 128 colors (16 primary colors and eight luminance levels)
- Over 75 BASIC commands
- High resolution graphics plotting
- Split-screen text with high-res graphics
- Graphic character set on keyboard
- Keyboard color controls
- 320 × 200 pixel screen resolution
- Reverse and flashing characters
- Two tone generators
- Built-in machine language monitor

Of course, the key to any computer is software and Commodore will have a full line of software available from your Commodore dealer. Specific applications include word processing, financial calculations, learning activities, mail lists, project planning, record keeping, home budget, investment analysis and much more. As usual, software will be available on cartridge, disk or tape. In addition, many models of the 264 have software built into the machine. As in all Commodore computers, BASIC is built-in, and other popular computer languages, such as LOGO, will be available as well.

One of the most outstanding features of the 264 is the additional set of BASIC commands available for programming, including full graphics plotting and program editing commands. Exercises are provided in the user manual to help programmers learn to use the new commands.

The 264 is compatible with Commodore's color monitor (or your TV), Commodore's datassette tape recorder and disk drive, Commodore printers or 1520 printer/plotter, the Commodore VICMODEM or AUTOMODEM and Commodore joysticks. **C**

Introducing the B-Series: Part 1

by Howard Rotenberg

The long awaited B-Series computer has finally arrived! This is the same computer that people have referred to as the P-500, B-700 and various other names. The product is officially called the B-128 or the B-256 according to the amount of memory that the computer incorporates. Incorporated into the new B-Series are an extended BASIC, extended memory support, error trapping, RS232 interface, advanced editing, advanced monitor commands with DOS support, special function keys and many other new advancements. The processor used in this computer is the 6509, which lets you address up to one megabyte or one million bytes of memory. It is accessed as 16 individual banks of 64K bytes. The speed of the computer's processing and I/O has improved since the 8032, which has usually been considered our business computer. The B-Series features an option to run CP/M 86, which will be available for those people interested in having that option.

A New Streamlined Look

The B-Series has a new, slick, streamlined look of the future. It is a low-profile computer molded in an eye-pleasing beige casing. It has ninety-three keys (ninety-four if we include the space bar). The keys have that rich feel that was available only on larger systems before.

The keypad is laid out for quick and accurate entry of numbers. It also incorporates arithmetic operators for ease of mathematical operations. It has an "enter" key and one key that has two zeros on it. This key will literally print two zeros when used, which is really handy when you are using large numbers. The "enter" key has the same function as the return key but its closer proximity to the keypad makes it more useful.

The key that I use the most and find extremely useful is the CE or "clear entry" key. This key will delete the last entry that you have made on the computer. Suppose, for instance, you had entered 456 + 120 and made a mistake on the second number. Normally if you want to change the 120 to another number, you have to use the delete key three times. Now with the CE key, you just need to press it once to have the 120 cleared in one stroke and the cursor left sitting over the position that the one occupied. The parser for the CE key is smart enough that even if you enter a calculation such as 456+120 with no spaces and then press the CE key, it will only delete the 120. If you press it once more it deletes the plus sign and when pressed a third time it clears the 456.

The main keyboard contains all the alphanumerics and symbols such as brackets, percent sign, etc. There is an escape key that is used in conjunction with any of the 26 alphabetic keys for a wide variety of functions. To perform one of these functions, the ESC key is hit followed by the appropriate letter. The functions that are provided allow some very good and interesting editing, some of which was available only on word processors prior to this.

It was nice to see that the keys once again had the graphic symbols printed on them for easy recognition and access. They are all available directly from the keyboard. The CTRL key will slow down scrolling while being held and the Commodore key will stop scrolling completely until another key is pressed. There are some format keys that allow you to switch between normal and graphics mode with just a key press. As usual, there is also a key to switch on reverse or normal graphics mode and the regular editing keys such as cursor movement and CLR/HOME are all present.

A refreshing and useful new feature is the programmable function keys. There are ten keys and by using the shift you have access to twenty different modes. The first ten keys are already preset upon power-up but can be re-programmed. To list the contents of these keys you just need to enter the word KEY.

The use of these keys is limited only to your imagination and 160 characters. With the proper syntax, you may have a key defined to do any batch of commands that you may want that may normally be executed in immediate mode.

Finally we come to the back of the computer. The ports available are an IEEE, audio out, cartridge, cassette (although not implemented), video out and a true RS-232 port. There is an internal user port that may be accessed if you need it.

Now that you have a good grasp on the new design and keyboard functions of the B-Series, we will go on to explore the extended BASIC commands that are available.

Extended BASIC

There has been a number of extensions to the BASIC interpreter. Some of these commands will allow a more structured program while others will do error trapping and generally allow easier programming. All the BASIC 4.0 commands are present in this new BASIC so I will just deal with the new ones or any that have changed. The following is a list of the new commands with a quick explanation and example of their use.

BANK: In the B-Series computers, there are either two or four 64K memory banks for the 128 and 256 series respectively. The bank command sets the bank for the POKE, PEEK and SYS commands.

BLOAD/BSAVE: Once again this command is used to reference the different 64K banks in the computer. This time they are for loading or saving a binary file to or from any location in memory.

POKE/PEEK: This command has only changed in the respect that you may now use it to look at or change locations in other banks. The default is bank 1, which is the basic text bank, or, to rephrase that, the bank that your written program resides in.

SYS/USR: These commands once again have not changed much, except that the SYS command used in conjunction with the bank command and the USR command calls a machine language routine with its starting address stored at locations three and four of bank 15. An expression in brackets is stored in the floating point accumulator prior to entering the subroutine.

FRE: This command will return the number of free bytes in the bank or segment specified by a parameter in brackets. An invalid parameter returns a zero.

DELETE: This command will delete a range of line numbers from the user's BASIC program. The parameters are the same as those used in list. NOTE: When you are deleting the entire program, it actually erases all of memory. It does not just reset the pointers and place three zeros at the start of the program. This means that it is absolutely irreversible.

DIRECTORY: This will display the directory of the desired disk. If the screen is about to scroll, it stops the display and prompts you with (MORE). Pressing any key except STOP will resume the display. The directory command will take an optional file name or a * as a wild card and search for those occurrences.

INSTR: This command will match the occurrence of b$ in a$ with an optional starting position of n. It will return the starting position or zero if not found.

IF-THEN-ELSE: This command structure will allow you to use more structured statements within your programs. If a condition is true then the following commands are executed until a branch or the end of the line is reached. If it is false then the commands following the else will be executed.

KEY: This is used to define the function keys that I previously discussed. The contents of the defined keys may be seen by typing key (CR).

TRAP: This command will disable the BASIC error-handling routines and let you handle the error yourself. This command is usually used with the reserved variables EL, ER, ERR$ and RESUME which will be discussed shortly.

RESUME: This statement is used in conjunction with TRAP. It specifies where execution will continue after an error has been trapped. If no options are specified it will try to re-execute the statement in error. You may use the NEXT option which will cause it to resume execution after the statement in error. If a line number is used, then it will proceed from there.

DISPOSE: This command will allow a user to terminate a FOR-NEXT loop or jump out of a subroutine without leaving garbage on the stack. An ideal use would be during an error trapping routine.

ERR$/EL/ER: These are new reserved variables. ERR$ will return an error message determined by a parameter. The valid parameters are zero to 42. When the variables EL and ER are used with TRAP, EL will hold the line number with the error, while ER will contain the correct error number used by ERR$.

PRINT USING: This is the command that will probably be most appreciated by the majority of users. It will provide formatted printing to the screen or a logical device. The format symbols that may be used are: #,+,−,.,comma,$,= and the exponential sign. The way to print to a file would be "print(#file), using clause; print list". For example, "print#4, using"$####.##";1000" will result in the figure $1000.00 being sent to that file. The format clause may be put into a string such as A$ = "$####.##".

PUDEF: This will allow the user to redefine certain symbols in the PRINT USING statements.

DCLEAR: This command will simply initialize the drive specified.

TI$: Although this is not a command, it has changed in that it now displays the time up to tenths of a second. It may be set by using seven figures or just six to make it compatible with an older program.

This winds up the new and revamped commands in the extended BASIC. Having the use of these new commands will definitely allow for easier programming and more structured programming.

Next issue we'll look at the B-Series in a little more depth. A good place to start is the machine language monitor. The new monitor has all the old commands plus six new ones, although the syntax of some of the old commands has changed. We'll take a look at the new commands as we continue to get a feel for Commodore's new computers. **C**

Income Tax Helper

by Stephen S. Leven

"Income Tax V/64" won't pay your taxes for you! But it can make figuring your federal income tax a bit easier. This program will work on the VIC 20 with 24K memory expansion or the Commodore 64, and allows data storage and retrieval to/from tape and/ or disk, as well as printed output on the VIC 1525 printer. Since the program is very long, we have not included the listing here. If you would like a copy, send a formatted disk or tape, self-addressed, stamped return mailer and $2.00 to Stephen at: 6 Hillery Court, Randallstown, Maryland 21133.

If you're like most Americans, buried somewhere under a pile of papers on your desk is your federal income tax packet for 1983. What a depressing thought—spending all that time working on your taxes when you could be playing (working ??) with your computer, instead. Being a bit lazy when it comes to repetitive arithmetic operations (but not when it comes to something as interesting and creative as programming), I wrote a program to help with some of the drudgery of the process.

"Income Tax V/64" is a menu-driven program that allows the user to input all his/her federal income tax data for Form 1040, Schedule A, and Schedule B. It is patterned after the 1982 federal income tax forms, so some minor modifications might be necessary to be fully compatible with the forms to be used for 1983. Since this program is written 100% in BASIC, you should be able to add the modifications yourself.

This program requires either a VIC 20 with 24K expansion, or a Commodore 64. The program includes a large number of REM statements to help identify various parts of the program. However, eliminating these REM's still will not allow the program to fit in a VIC with 16K expansion, so you might as well include them when you type the program. If you eliminate the sections relating to the printout (Lines 3510 through 4570), and change Lines 1140 and 1180 to read:

```
1140 PRINTSPC(RR)RQ$"7
  DISPLAY THE DATA"
1180 ONVAL(G$)GOTO1270,
  3090, 2860, 1270, 860, 1080,
  4610, 1230
```

the program will fit within a 16K expanded VIC (21K total, or 19967 free bytes on power-up).

What "Income Tax V/64" will *not* do for you is, of course, pay your taxes! In addition, the tax tables are not programmed into the program. "Income Tax V/64" will tell you what your adjusted gross income is, ask you to look in the appropriate table to find the amount of taxes you owe, then wait for you to enter that amount.

Full Featured Program

Income Tax V/64 is a full featured program, similar in scope to professional quality software. Let's look at what the program has to offer.

After the program is LOADed (it takes over six minutes to LOAD the tape version and about one minute to LOAD the disk version) the user is asked for the last two digits of the year. For this year's taxes, enter 83. It is extremely important to enter the correct year, since the year becomes a part of the name of the data file which is found and read from tape or disk. The program will not read data on the tape or disk for year 1982 if you have entered 1983 as the year. I suggest using the year which shows on your current tax forms (1983) rather than the current calendar year (1984). After entry of the year's last two digits, it is not necessary to press RETURN.

Next, you are asked to enter three initials of the taxpayer. Again, it is important to enter the correct initials since these, too, become part of the data file name. This feature is to provide separate file names in case you also do someone else's taxes. The main menu is then displayed, showing all of the features and options of the program. These include:

1 ENTER DATA-KEYBOARD.
New income tax data can be entered, one item at a time, from the VIC or 64 keyboard.

2 ENTER DATA DISK/TAPE.
Data which has been previously stored on disk or tape can be re-entered from the disk drive or datassette.

3 STORE DATA DISK/TAPE.
Data which is in the computer's memory, which you have entered from either the keyboard, disk or tape, or which has been changed, can now be stored on disk or tape using the disk drive or datassette.

4 CORRECT THE DATA.
Data which has been entered, from either the keyboard, disk or tape can be corrected.

5 CORRECT TAXPY'R/YEAR.
If either the year or the taxpayer's initials, which were entered before the menu appeared and are displayed at the top of the menu, is incorrect or otherwise needs to be changed, this can be accomplished without losing any of the other data in the computer's memory.

6 PRINT THE DATA.
This option outputs the data to the VIC 1525 printer. The data is output in a concise manner that produces two 8½" by 11" pages of printout. The first page contains all the entries for sides one and two of Form 1040. The second page contains all the entries for Schedule A and Schedule B.

7 DISPLAY THE DATA.
The data is displayed, up to eight entries at a time, on the screen. After each "page" of data, the computer pauses until the user is ready to see the next page. Since the display of data takes 21 pages, and therefore can be time consuming, the user can return to the menu after viewing any page.

8 END THE PROGRAM.
The screen clears and the message "GOTO1080" is displayed, with the cursor flashing on that line. Since none of the data is destroyed by ENDing the program, simply pressing RETURN will take you back to the menu. If you wish to do something other than return to the menu, just move the cursor off of that line before continuing. Typing GOTO1080 will always return you to the main menu.

Program Operation

In the ENTER DATA-KEYBOARD mode, the program walks you through all the possible entries for Form 1040, Schedule A (whether or not you think you have enough deductions to itemize) and Schedule B. Schedule B is first. You are asked to enter data for interest income from an individual mortgage loan. Enter your actual amount. If the amount is zero, you may simply hit RETURN.

Next, you are asked to enter interest income from banks and savings institutions. You may enter data for up to ten accounts, including identifying data such as bank name or account number, and, of course, the amount of interest income. If you have entered all your regular interest data, simply type END when asked for the name of the next account.

You will then be asked to enter income from allsavers certificates. Again, you may enter data for up to ten accounts. Type END when you are done. You will be prompted for your allsaver exclusion. Enter the proper amount. You will then be asked to enter your dividends in a manner similar to the interest and allsavers certificates. You again have up to ten entries, and type END when finished.

The program then asks you for the amount to be included on each of the lines for the remainder of Schedule B, Form 1040 Side 1, Schedule A, and Form 1040 Side 2, in that order. Note that only those items which require original input from the user are asked. All data which are merely additions, subtractions, or percentages of previous lines are not asked for, but are calculated as the program runs. As mentioned earlier, if the dollar amount for a given entry is zero you may simply hit RETURN.

Of special note are the following:

For number of exemptions, enter the total number, such as four for a "standard" family of two parents under 65 and two children. The computer will calculate the dollar amount of your exemptions.

For general sales tax, the computer displays your adjusted gross income, added together with those entries which are allowed in using the optional state sales tax tables. You use this dollar amount, together with the "family size", to look up the applicable sales tax deduction for your state. Of course, if your records show that you paid more in sales tax than the table lists, you can deduct the higher amount. In any case, enter the amount of sales tax deduction.

If you don't have enough deductions to itemize, the program will tell you. You don't have to do anything special—the standard deduction is automatically figured for you. A special charitable contribution deduction for nonitemizers is calculated for you, if applicable, and will appear in the printout or display.

Your total taxable income is displayed, and you are asked to enter your tax due. Look up this information in the tax tables and enter it. If the amount is zero, you must enter a zero, in this case.

Continue to answer the questions as before. At the end of the series of questions, your refund (hopefully!!) or balance due will be displayed. The next stroke of any key will return you to the main menu.

In the CORRECT THE DATA mode, the program walks through each data item just as in the ENTER DATA-KEYBOARD mode. However, this time the existing information in the computer's memory is displayed. You have the option to change any data item(s) simply by entering the correct or new data at the prompt. If you wish to leave the data item the same, simply type RETURN, except in the case of income tax due, where you must enter the correct dollar amount, even if it is the same as the one displayed.

At the end of the trip through the tax form, you will again be returned to the main menu.

Data Storage/Retrieval

In the DATA DISK/TAPE mode, you are first asked whether your data is on tape or disk. Simply type the number of the correct answer. As always, you also have an option to abort and return to the main menu.

If your data is on disk, the disk drive will immediately begin searching for your data. Once the data is loaded, the menu will reappear. If there is a disk error in searching for your data file or in reading the data, the error message will be displayed on the screen.

If your data is on cassette tape, you are asked to put the correct tape, containing the previously stored data for this tax year, into the datassette and rewind/fast forward the tape to the proper position. If you change your mind about what you are doing at this point, you can instantly go back to the menu by typing a C for "cancel" or M for "menu".

In the DATA STORAGE DISK/TAPE mode, you are also asked if you want to store your data on tape or disk. If you choose disk, the storage will begin immediately. If you choose tape, you are asked to put the tape on which you want the data into the datassette and position it to the proper location. In this mode, too, you can change your mind and return to the main menu by simply typing a C or M.

The CORRECT TAXPY'R/YEAR mode will allow you to change the year or taxpayer initials without losing any of the financial data. This option is primarily useful when you find that you cannot retrieve previously stored data because the year or initials do not match those in the name of the file you wish to access, or you wish to create a new file of the same data.

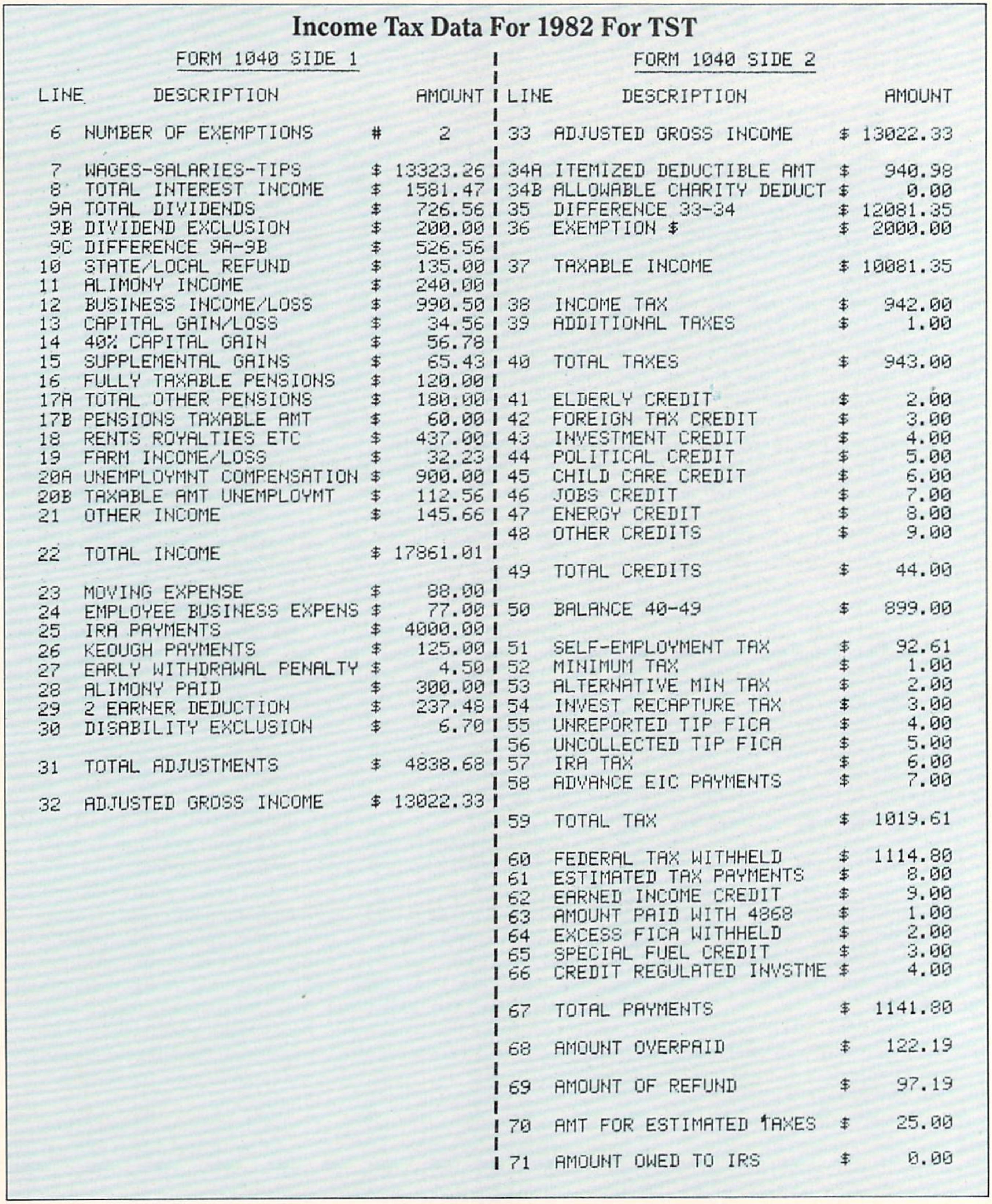
Data Output

No program would be complete without some means of data output, and this one is no exception. The data can be displayed on the screen or printed on the VIC 1525 printer, so that the dollar amounts can be transferred to your income tax forms.

In the PRINT THE DATA mode, the program asks you to make sure that your printer is set to top-of-form. This is done by advancing the printer paper until the seam or fold of the paper is positioned even with the top of the printer head. See Figure 1. It is important to set the printer to top-of-form because the printout is a full two pages long, and this can be filed with your income tax records for the year.

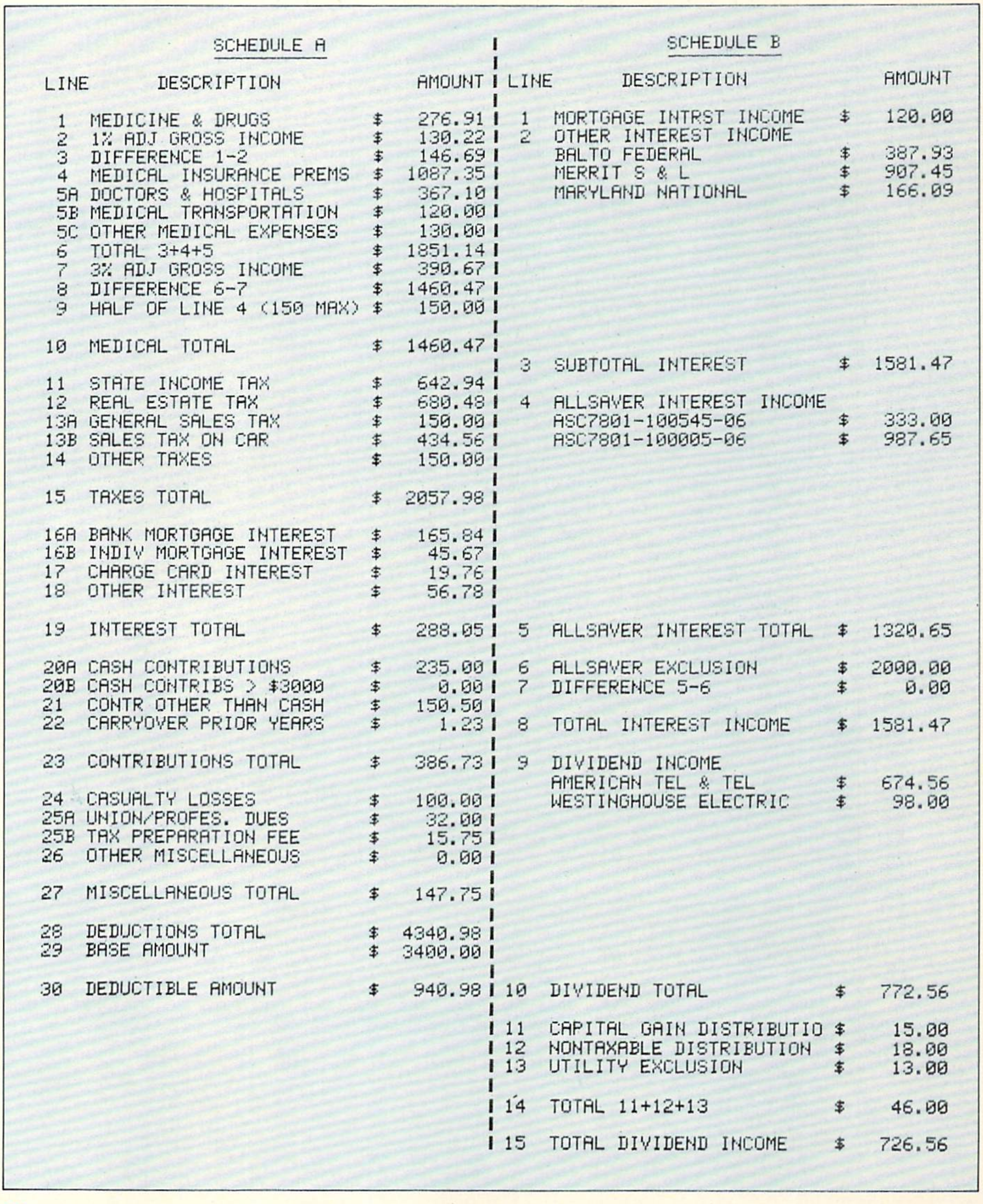
At this point, you may cancel the printout and return to the main menu by typing M for "menu". To begin the printout, press any other

Sample Output: Income Tax V/64



Income Tax Data For 1982 For TST

```
         FORM 1040 SIDE 1                     I           FORM 1040 SIDE 2
                                              I
LINE      DESCRIPTION                  AMOUNT I LINE      DESCRIPTION                   AMOUNT
                                              I
  6  NUMBER OF EXEMPTIONS         #        2  I 33  ADJUSTED GROSS INCOME         $ 13022.33
                                              I
  7  WAGES-SALARIES-TIPS          $ 13323.26  I 34A ITEMIZED DEDUCTIBLE AMT       $   940.98
  8  TOTAL INTEREST INCOME        $  1581.47  I 34B ALLOWABLE CHARITY DEDUCT      $     0.00
 9A  TOTAL DIVIDENDS              $   726.56  I 35  DIFFERENCE 33-34              $ 12081.35
 9B  DIVIDEND EXCLUSION           $   200.00  I 36  EXEMPTION $                   $  2000.00
 9C  DIFFERENCE 9A-9B             $   526.56  I
10   STATE/LOCAL REFUND           $   135.00  I 37  TAXABLE INCOME                $ 10081.35
11   ALIMONY INCOME               $   240.00  I
12   BUSINESS INCOME/LOSS         $   990.50  I 38  INCOME TAX                    $   942.00
13   CAPITAL GAIN/LOSS            $    34.56  I 39  ADDITIONAL TAXES              $     1.00
14   40% CAPITAL GAIN             $    56.78  I
15   SUPPLEMENTAL GAINS           $    65.43  I 40  TOTAL TAXES                   $   943.00
16   FULLY TAXABLE PENSIONS       $   120.00  I
17A  TOTAL OTHER PENSIONS         $   180.00  I 41  ELDERLY CREDIT                $     2.00
17B  PENSIONS TAXABLE AMT         $    60.00  I 42  FOREIGN TAX CREDIT            $     3.00
18   RENTS ROYALTIES ETC          $   437.00  I 43  INVESTMENT CREDIT             $     4.00
19   FARM INCOME/LOSS             $    32.23  I 44  POLITICAL CREDIT              $     5.00
20A  UNEMPLOYMNT COMPENSATION     $   900.00  I 45  CHILD CARE CREDIT             $     6.00
20B  TAXABLE AMT UNEMPLOYMT       $   112.56  I 46  JOBS CREDIT                   $     7.00
21   OTHER INCOME                 $   145.66  I 47  ENERGY CREDIT                 $     8.00
                                              I 48  OTHER CREDITS                 $     9.00
22   TOTAL INCOME                 $ 17861.01  I
                                              I 49  TOTAL CREDITS                 $    44.00
23   MOVING EXPENSE               $    88.00  I
24   EMPLOYEE BUSINESS EXPENS     $    77.00  I 50  BALANCE 40-49                 $   899.00
25   IRA PAYMENTS                 $  4000.00  I
26   KEOUGH PAYMENTS              $   125.00  I 51  SELF-EMPLOYMENT TAX           $    92.61
27   EARLY WITHDRAWAL PENALTY     $     4.50  I 52  MINIMUM TAX                   $     1.00
28   ALIMONY PAID                 $   300.00  I 53  ALTERNATIVE MIN TAX           $     2.00
29   2 EARNER DEDUCTION           $   237.48  I 54  INVEST RECAPTURE TAX          $     3.00
30   DISABILITY EXCLUSION         $     6.70  I 55  UNREPORTED TIP FICA           $     4.00
                                              I 56  UNCOLLECTED TIP FICA          $     5.00
31   TOTAL ADJUSTMENTS            $  4838.68  I 57  IRA TAX                       $     6.00
                                              I 58  ADVANCE EIC PAYMENTS          $     7.00
32   ADJUSTED GROSS INCOME        $ 13022.33  I
                                              I 59  TOTAL TAX                     $  1019.61
                                              I
                                              I 60  FEDERAL TAX WITHHELD          $  1114.80
                                              I 61  ESTIMATED TAX PAYMENTS        $     8.00
                                              I 62  EARNED INCOME CREDIT          $     9.00
                                              I 63  AMOUNT PAID WITH 4868         $     1.00
                                              I 64  EXCESS FICA WITHHELD          $     2.00
                                              I 65  SPECIAL FUEL CREDIT           $     3.00
                                              I 66  CREDIT REGULATED INVSTME      $     4.00
                                              I
                                              I 67  TOTAL PAYMENTS                $  1141.80
                                              I
                                              I 68  AMOUNT OVERPAID               $   122.19
                                              I
                                              I 69  AMOUNT OF REFUND              $    97.19
                                              I
                                              I 70  AMT FOR ESTIMATED TAXES       $    25.00
                                              I
                                              I 71  AMOUNT OWED TO IRS            $     0.00
```



```
           SCHEDULE A                       I               SCHEDULE B
                                            I
LINE       DESCRIPTION               AMOUNT I LINE       DESCRIPTION                AMOUNT
                                            I
  1  MEDICINE & DRUGS          $     276.91 I  1   MORTGAGE INTRST INCOME   $      120.00
  2  1% ADJ GROSS INCOME       $     130.22 I  2   OTHER INTEREST INCOME
  3  DIFFERENCE 1-2            $     146.69 I      BALTO FEDERAL            $      387.93
  4  MEDICAL INSURANCE PREMS   $    1087.35 I      MERRIT S & L             $      907.45
 5A  DOCTORS & HOSPITALS       $     367.10 I      MARYLAND NATIONAL        $      166.09
 5B  MEDICAL TRANSPORTATION    $     120.00 I
 5C  OTHER MEDICAL EXPENSES    $     130.00 I
  6  TOTAL 3+4+5               $    1851.14 I
  7  3% ADJ GROSS INCOME       $     390.67 I
  8  DIFFERENCE 6-7            $    1460.47 I
  9  HALF OF LINE 4 (150 MAX)  $     150.00 I
                                            I
 10  MEDICAL TOTAL             $    1460.47 I
                                            I  3   SUBTOTAL INTEREST        $     1581.47
 11  STATE INCOME TAX          $     642.94 I
 12  REAL ESTATE TAX           $     680.48 I  4   ALLSAVER INTEREST INCOME
13A  GENERAL SALES TAX         $     150.00 I      ASC7801-100545-06        $      333.00
13B  SALES TAX ON CAR          $     434.56 I      ASC7801-100005-06        $      987.65
 14  OTHER TAXES               $     150.00 I
                                            I
 15  TAXES TOTAL               $    2057.98 I
                                            I
16A  BANK MORTGAGE INTEREST    $     165.84 I
16B  INDIV MORTGAGE INTEREST   $      45.67 I
 17  CHARGE CARD INTEREST      $      19.76 I
 18  OTHER INTEREST            $      56.78 I
                                            I
 19  INTEREST TOTAL            $     288.05 I  5   ALLSAVER INTEREST TOTAL  $     1320.65
                                            I
20A  CASH CONTRIBUTIONS        $     235.00 I  6   ALLSAVER EXCLUSION       $     2000.00
20B  CASH CONTRIBS > $3000     $       0.00 I  7   DIFFERENCE 5-6           $        0.00
 21  CONTR OTHER THAN CASH     $     150.50 I
 22  CARRYOVER PRIOR YEARS     $       1.23 I  8   TOTAL INTEREST INCOME    $     1581.47
                                            I
 23  CONTRIBUTIONS TOTAL       $     386.73 I  9   DIVIDEND INCOME
                                            I      AMERICAN TEL & TEL       $      674.56
 24  CASUALTY LOSSES           $     100.00 I      WESTINGHOUSE ELECTRIC    $       98.00
25A  UNION/PROFES. DUES        $      32.00 I
25B  TAX PREPARATION FEE       $      15.75 I
 26  OTHER MISCELLANEOUS       $       0.00 I
                                            I
 27  MISCELLANEOUS TOTAL       $     147.75 I
                                            I
 28  DEDUCTIONS TOTAL          $    4340.98 I
 29  BASE AMOUNT               $    3400.00 I
                                            I
 30  DEDUCTIBLE AMOUNT         $     940.98 I 10   DIVIDEND TOTAL           $      772.56
                                            I
                                            I 11   CAPITAL GAIN DISTRIBUTIO $       15.00
                                            I 12   NONTAXABLE DISTRIBUTION  $       18.00
                                            I 13   UTILITY EXCLUSION        $       13.00
                                            I
                                            I 14   TOTAL 11+12+13           $       46.00
                                            I
                                            I 15   TOTAL DIVIDEND INCOME    $      726.56
```

key, as instructed. If your printer is not switched on or is otherwise improperly connected, you will get an error message and be "dumped" out of the program. To get back to the menu without losing any of your data immediately type: GO TO 1080.

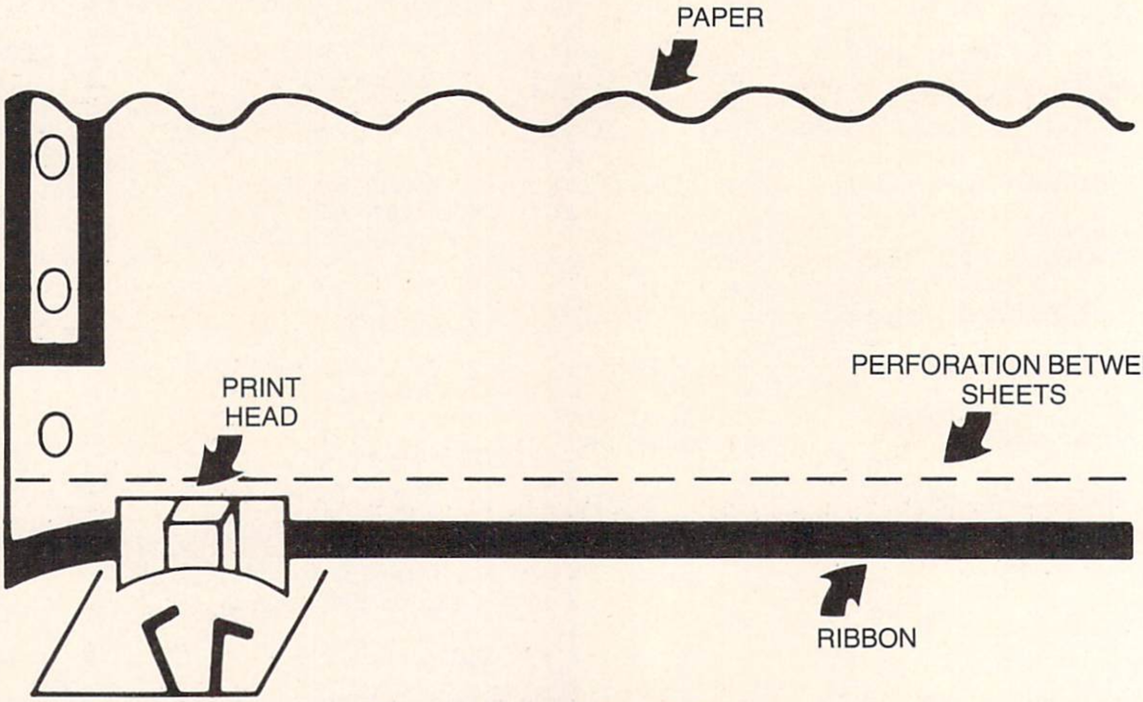
When the printout is finished, you will be returned to the main menu.

DISPLAY THE DATA will do just that. The display consists of a heading that shows which form is being displayed and which page of the 21-page display is being displayed. Since the VIC is limited to a 22-character screen width, each entry uses two lines. The first line contains the description of the entry, and the second line lists the line number from the income tax form, as well as the appropriate amount. The Commodore 64, with a 40-character display, will list all the data on one line, but the lines will be double spaced for clarity in reading. Up to eight lines appear on a page. Pressing M will end the display and return you to the main menu. Pressing any other key will cause the next page to be displayed.

During both the data display and printout, all data items will be displayed or printed, including those that were not entered by the program user but were calculated by the program.

END THE PROGRAM clears the screen and ends the program. The statement GOTO1080 is typed at the top of the screen. If you END THE PROGRAM by mistake, simply hit RETURN to get back to the main menu, since the data is not CLeaRed from memory. If you really intended to END THE PROGRAM, just move the cursor from that line before going ahead. **C**

FIGURE 1. Setting Printer to Top-of-Form



Print Head Aligned with Paper Perforations

Software Copyrights: How and When Do They Apply?

by Herbert Swartz

This month we begin a series of articles by copyright lawyer Herbert Swartz concerning the struggle to maintain the copyright "balance" for computer software—a struggle in which Congress, the courts and the computer industry are joined.

Some 500 years ago, Johannes Gutenberg invented the printing press. Then in 1710, England passed the first Copyright Act, the Statute of Anne. Thus began the copyright scheme, the process of furthering knowledge by encouraging authors to create.

That encouragement has taken two forms: Authors are given "exclusive" control over the right to copy their works and they receive the economic rewards from the sale of their works. Such dual monopoly, however, is limited. It exists not forever but for a limited time—today, the life of the copyright owner plus 50 years. Furthermore, the public is given free access to the use of the work; only copying is prohibited. In short, the author's limited monopoly extends to his expression. Ideas—which cannot be copyrighted—remain in the public domain.

A recent "Note" in the *Harvard Law Review* summarized the proposition of our copyright scheme:

> The law of copyright is designed to advance social welfare by maximizing the public availability of literature, music, and the arts. Because a regime under which all creative works were in the public domain would discourage creative effort, copyright seeks to guarantee some economic return to the copyright owner by granting the owner a limited monopoly over his work. Copyright thus achieves its goal of maximizing availability by balancing the desire of the public for uninhibited access to intellectual works against the desire of the creators of those works for financial reward.

In the United States, this "balance" is founded in the Constitution, Article I, Section 8, Clause 8: "Congress shall have Power... To promote the Progress of Science and the useful Arts by securing for limited Times to Authors and inventors the exclusive Right to their Writings and Discoveries."

But in the mandate to Congress lies what Professor Leon Seltzer of Stanford Law School, author of *Exemptions and Fair Use in Copyright,* describes as a "built-in tension." The public's need for knowledge is always threatening to encompass more and more of the author's "exclusive rights." When this happens, we have "imbalance."

Today, with the new technology, copying—indeed exact duplication, what the law calls "naked copying"—is easier than ever before. In addition, the public's need for access to the new technology is immense. These two factors have brought us to the edge of "imbalance." As a result, according to our most celebrated copyright scholar, Professor Melville Nimmer of UCLA Law School, author of *Nimmer on Copyright,* we have reached "the twilight of the Age of Gutenberg."

Nowhere is the threat of "imbalance" more pressing than in the area of copyright protection for microcomputer software, especially since only copyright law provides any real protection for the rights of software authors.

Understanding the copyright scheme presently in existence, creators of software can protect their rights and gain the financial rewards due to them. In turn, the computer industry and society prosper.

And understanding has another benefit. As you might expect, any legal scheme almost 300 years old is far from perfect in many ways for computer software. The new technologies are simply too new. The dangers, the gaps and the errors that lurk within copyright protection for computer software unfortunately can never be remedied until they are recognized.

Copyright Today

Copyright law today is governed by the Copyright Act of 1976 which became law on January 1, 1978.

Several times since the Constitution, Congress has passed copyright statutes. What is significant—and different—about the 1976 Act, though, is that for the first time we have a copyright system that is "automatic" and "unitary."

It is "automatic" because copyright attaches upon creation. As long as a copyrightable work—and Congress and courts have now clearly said that computer programs are a category of copyrightable works—was created after January 1, 1978, it is automatically copyrighted as soon as it is "fixed" (*e.g.*, written on paper) so that it can "be perceived, reproduced, or otherwise communicated for more than a transitory duration." However, as the House Report on the 1976 Act emphasized, "purely evanescent or transitory expressions such as those... captured momentarily in the running of a computer" would *not* be automatically copyrighted, since they are not "fixed"—although once a program is saved on disk or tape, it then becomes fixed and thus copyrighted.

And our copyright scheme is "unitary" because as of January 1, 1978, state common-law copyright vanished. Congress preempted the field from the states, as it is allowed to do under the Supremacy Clause of the Constitution. Formerly we had a dual copyright system in the United States—state and federal. This is no longer the case, and thus unless the niceties of federal copyright law are followed, no copyright protection is available for writers of computer software, or any other authors.

Today, therefore, copyrighting a work is no longer a question. Creating a work in a copyrightable category, and computer programs are such a category, creates a copyrighted work. Nothing more is at issue.

What Are Your Rights?

In spite of the "automatic" nature of copyright, however, owning a copyright provides no rights or remedies unto itself. Congress provides these rights and remedies by statute, and in order for the statute—and a copyright owner's rights and remedies to come into force—a copyright must be registered, and properly. But, assuming due registration (we shall detail this later), what does the copyright owner then have?

Precisely what the copyright owner might expect to have: the "exclusive right" to make copies and preclude others from doing so. Nor is the copyright owner's "bundle of rights" limited to copying. The author controls as well the licensing, manufacturing, sale and use of the copyrighted software. The program is used by the public as purchasers or licensees because the copyright owner has allowed such events to come to pass. So, authors might have licensed software publishers to license their programs. Or, they can split the copyright so that one publisher can market programs on one computer and others on different computers. The copyright owner can give, either as a gift or by will, the copyright to someone else—just as with the ownership rights of any property. The owner controls, in sum, the distribution of copyright, and it is this right that provides the legal tool to combat piracy.

Remember, however, that the copyright does not protect ideas, algorithms, program names or company names. *Only the expression is protected.* Naturally, as at least one Supreme Court justice has declared, the separation of ideas from the expression of the ideas can get a bit metaphysical at times, and certainly computer programs, especially those embedded on ROM, will prove that point once again. But for the moment, it is enough to note that what Congress says is unprotected—*e.g.*, ideas—and therefore in the public domain, remains so until Congress speaks to the contrary. No state law is possible to make private what Congress says is public.

Achieving the "balance" between the public's need for access to computer software and the author's rights to make money on their creations is quite a noble and prescient undertaking. For example, consider "expression" versus "idea" in the case of, say, a word-processing program that incorporates a unique feature. The feature itself, even though "original," as all copyrightable works must be, is *not* subject to copyright. The actual program lines, though, that cause the feature to execute *are* copyrightable once "fixed." They can in turn be registered as a copyrighted work. The author has his "bundle of rights." No one can copy the work without his permission.

Nevertheless, as long as some user does not copy the actual lines, that user is free to write a program that does the *same thing* as the first program without incurring copyright infringement. Rather, a second copyrighted work has been created—and this, for the world of computer software, is how the balance between public and private, expression and idea, is presently designed to function.

As noted, copyrightability of the "expression" in computer programs extends to all original computer programs as soon as they are "fixed," even those embedded in ROM—a development highlighted by the recent decision in *Apple v. Franklin.* The point is not that the idea of the program be brilliant or novel but that it simply be "original" and "fixed" in a tangible medium of expression—in writing on or a floppy disk or tape.

Protecting Your Rights

However, what Congress has chosen to give as part of the public-private balance, Congress can taketh away. Copyright that is given "automatically" upon creation is *lost* if "publication" takes place without appropriate copyright notice. Obviously, "publication" is a key legal issue.

The statute defines the term as "the distribution of copies of a work to the public by sale or other transfer of ownership, or by rental, lease, or lending." No one can dispute that "publication" takes place when an article or program appears in this or any other magazine, or when a software publisher begins to take a program to market. But an author of a program may have "published" in other ways, not so readily apparent, without realizing the consequences of his acts. For example, people often lend to a friend the source code of a program they have written. Legally, this would probably trigger "publication" and what it entails. So a good rule of thumb—when in doubt assume "publication" has taken place and affix a copyright notice. This could save heavy legal bills, a long trial, and possible disappointment at a later date. Nothing is lost if a notice is missing from an "unpublished" work, but why take the chance.

Fortunately, Congress has provided some escapes from the dire consequences of "publication without notice." If a work is registered with the United States Copyright Office before the alleged infringement took place, the copyright notice will be deemed *not* to have been "omitted"—and the suit for infringement can be maintained, since the work will still be considered to be owned rather than to have slipped into the public domain. Nor will the work, though published, be deemed public domain if the notice was "omitted from no more than a relatively small number of copies... distributed to the public." And in any event, if the "omission" is considered by a court to be unintentional, registration can be made within five years after the "publication without notice", as long as "a reasonable effort is made to add notice to all copies... distributed within the United States after the omission has been discovered." (A few lower federal courts have been willing to extend the five-year period to cover intentional omissions as well, but most courts do not, so where the omission is intentional this "out" should not be counted on.) The important point to note is that this escape mechanism is dual: you may protect your copyright if you add notice as soon as you discover the omission AND you act to register within five years.

Registration thus may become vital if you have omitted notice for a published work. Regardless, it is vital for all works in order to enforce rights and remedies under copyright law. At its simplest—no registration equals no suit for copyright infringement. Fortunately, however, registration is not difficult and usually costs but $10. (The 1976 Act changed the function of the Copyright Office. Whereas formerly that office existed as a place to issue copyrights, since January 1, 1978, it has been an office of record—the place where copyrights can be registered, and the place where the record of registration is retained.)

According to the statutory scheme, registration is permissive but not mandatory for all works, pub-

lished or unpublished. You don't forfeit your rights if you fail to register. However, you must be registered if you intend to bring suit for infringement.

Registration also determines whether or not all available remedies for copyright infringement are available. For if registration takes place before infringement occurs—before someone violates one or more of the copyright owner's "bundle of rights" —the full arsenal of remedies provided by Congress is available. These include injunctions, impoundment, disposition of infringing articles, lost profits, other damages and recovery of the infringer's profits. Two other noteworthy remedies are recovery of attorney's fees and statutory damages. But if registration occurs after the date of the claimed infringement, these last two remedies are gone. (However, owners of published works can register three months after first publication and still retain all remedies for infringement.)

Eliminating these last two remedies in cases of late registration represents Congressional policy to induce people to register, and promptly. Statutory damages, for instance, are especially quite commonly relevant to suits for copyright infringement. "Damages" are often speculative and hard to prove. *Statutory* damages, however,—running from a low of $100 per offense to a high of $50,000 per offense—are in the nature of penalties for the offense rather than compensation for the damages suffered. They are thus an excellent "inducement" to settle as well as a simple way of estimating the amount to be recovered in a suit.

As for innocent infringers, when a copyright notice has been omitted—regardless of whether it has been corrected by registration or by affixing the notice—here too the remedies of attorney's fees and statutory damages are lost for infringements prior to notice to the offender. However, a court may require the infringer to disgorge profits made from the use of the copyrighted work during the preceding period.

For purposes of registration, computer programs are classified as literary works by the Copyright Office. The registration provisions to follow are threefold—a completed application form, a copy (deposit) of the work with appropriate notice (the issues of deposit of computer programs and appropriate notice for them will be the subject of our next article) and the fee. Forms and a fee schedule are available free of charge from the Copyright Office, Information and Publications Section, LM-45, Library of Congress, Washington, D.C. 20559. There is as well a hotline number for questions—(202) 287-8700—open after five o'clock EST. **C**

Getting the Most From Your Business Computer:
Word Processing

by Donald E. Hassler
Phoenix, Arizona

A word processing program can help you save money—and save you from tragedies like Jimmy Retailer's.

It's 4 p.m. on June 29th and Jimmy Retailer is sweating to get his first big school bid ready. It's going to be more than $50,000. He's been courting Mr. Bigspender, the purchasing agent for District 13, for two years. The final spec was written and sent out this June for Jimmy's specific products. Wow!

Jimmy fought hard to get this business away from Amalgamated School Supply, the supply house that has been successfully serving the district for 22 years. Now, with this entree, his business will take a 40% increase. At last he'll be able to do that 2000 square-foot expansion he's dreamed of since he opened five years ago. Everything is going wonderfully... or is it?? Just ahead is a major catastrophe for Jimmy.

Unknown to him, that little flu bug has been working on Martha, Jimmy's main assistant and only typist. She has all the material in her head and plans to spend all day tomorrow typing this big bid for Jimmy. Alas, it is not to be. Martha will call in sick with a 103-degree fever. No typing, no bid, all that work shot... and no big expansion this year. Jimmy should never have been caught in this bind.

How, you might ask, could he have avoided it? The answer is sitting right there in the office, unused except at the end of the month for general ledger and payroll—Jimmy Retailer's Commodore computer.

I'm writing this article from my own experience as a small businessman using a microcomputer and specifically a Commodore computer. I will suggest a way you can avoid Jimmy Retailer's problem and explain how you can get more out of your Commodore business computer, using it in one more productive way.

If you are a small-business person with a Commodore computer and not using it for typing, letters, proposals, mailings and the like you're missing a $10,000 per year helper. That's how much you'd have to pay (or more) to put a skilled typist on your payroll to do only a part of what a word processing program can do for you. And the Commodore computer doesn't have vacations, sick leave, problems with the spouse and kids or quit right at five to go home. It sits there patiently, 24 hours a day, waiting for you or one of your assistants to give it words to process.

Up to now I have written about cash register sales programs, general business accounting and spread sheet programs. The next big item to exercise your Commodore computer on is a word processing program. Be sure you discuss the available programs with your local Commodore dealer and follow his advice on how to get started. A word of caution for first-time users: buying a word processing program by mail may be asking for trouble. A word processing system requires explanation and training from someone you can count on to help quickly and personally when you get in a bind.

Lesson number one: Buy a word processing system that will come with at least four hours training from the dealer. Unless one of your people has direct experience with word processing it is unlikely anyone in your company will sit down and learn it without help. Furthermore, installation of programs with certain computer/printer combinations will require the help of an expert (the computer dealer) who has been through the grief that you will avoid. Have the system demonstrated to everyone and have the training aimed at several persons, not just secretaries or clerks.

Lesson number two: More people should be using word processing than just typists. That's the secret to enhanced productivity.

The more people that can get on your system and do simple work, the more output you'll get.

The most obvious uses you'll try immediately are simple letters and bulletins. They're easy to do. And if it's work that is recurring, just store it on disk to be brought up in the future for revision and new output. Try preparing an employee name and address list. If it's no more than a page or two, better to keep it on your WordPro than in some complicated database.

Typing multi-page bids and proposals is the workhorse benefit of any word processing program. All the material may be prepared in stages (sub-paragraphs, sections, etc.) and stored. As it builds you have all the parts there, ready to go. If the typist misses "bid day", no problem. Everything is already there. Just punch "print" and out it comes, quickly and letter-perfect.

This is lesson number three: You don't have to be a good typist (or any kind of typist) to get perfect work from word processing. The editing is already done by you on the screen. When everything, including format and spacing, is ready to go you just print it!!

The next thing is to expand the use of your word processing system with personalized letters. Doing this is very easy with the right program. All you need is a letter with blanks (called "variable blocks") where the personal information goes, plus a separate list of the actual names, addresses, etc. Then you simply start the letters. They can be printed on individual sheets of letterhead or on continuous stationery if you want more than 20 or 30 at a time.

Finally, to reach the most sophisticated level with your system, add a spelling program. When the letter, bid or proposal is finished just run the spelling package. It will find words it doesn't recognize and give you a chance to correct the spelling or to leave the word as is. You can also add that word to the dictionary if it's one that is used a lot. Technical and special trade terms are examples of words that need to be added to any spelling program.

The final lesson is: If your Commodore business computer is sitting there, only used part-time, get it on your payroll now as a free typist. There is a world of wordpower out there waiting for you. **C**

Text Analysis

by Andy Gamble

A program for the PET that helps educators determine the level of difficulty of textbooks and other writing.

> "He writes the worst English that I have ever encountered. It reminds me of a string of wet sponges."
> *H.L. Mencken, on W.G. Harding.*
>
> "Sometimes these meandering words would actually capture a straggling thought and bear it triumphantly a prisoner in their midst. . . ."
> *Senator McAdoo, on W.G. Harding.*
>
> "I would like the government to do all it can to mitigate, then, in understanding, in mutuality of interest, in concern for the common good, our tasks will be solved."
> *W.G. Harding*

I teach in a college that caters to foreign students, to whom English is a second language. Textbooks for such students have to be chosen very carefully, matching the reading difficulty to their appropriate grasp of English. Until recently the grading of texts was done "by hand", according to various well-established principles, often involving lots of coffee.

There are many methods available to assess the reading level of a text; all involve some counting of words, sentences and sometimes syllables, and all are tedious to perform by hand. This is exactly the sort of thing that machines do so well. Text-analysis programs are therefore of a great deal of interest to us. These programs take a sample of text input by the user and analyze it as to grade level.

The manner generally considered the most accurate is known as the Dale-Chall method. Briefly, this procedure involves comparing the given text with 3000 "familiar" words, and as such is beyond the scope of a small program such as this. The two methods presented here do not use vocabulary searches.

Gunning's Fog Index (it measures the clarity of a text!) counts the total number of words, including those considered difficult to read, and the average sentence length. In this program, difficult words are those with more than nine letters or more than three syllables. It would be extremely hard for a program to count syllables, however, and so a count is taken of all single vowels except an "e" at the end of a word. For all practical purposes this gives the same result. The formulae can be found in line 350. The Fog Index was originally designed to work with business letters and reports, and therefore gives a better result with report-like material such as science texts.

The second method used here is the Fry Index. A count is taken of syllables, words and sentences (lines 390 to 400). When this method is used manually, these results would be compared to a standard graph from which the reading level is derived. In this program the graph lines have been transferred to statements 410 to 520, which give the reading level directly.

There are various considerations needed for programs like this one. As experienced users of the PET know, it is all too easy to crash a program (but not fatally) by pressing "return" with no input (technically returning a null string). Since a program such as this one would be used by novice users (read: "English teachers"), I did not look forward to being summoned at all hours of the day and night to rescue the program.

A second consideration is that the actual analysis should take place during the typing of the text. If the analysis took place after all the text was typed in, it would leave one staring at an inactive computer for a time. Again, an inexperienced user might imagine that something had gone wrong.

Further, unnecessary restrictions should not be placed upon the mode of typing. If INPUT statements were used, commas would be forbidden. It is annoying to have to remember to delete all punctuation except at the end of the sentence.

The following program alleviates these problems. It's written for the PET/CBM, but converting to run

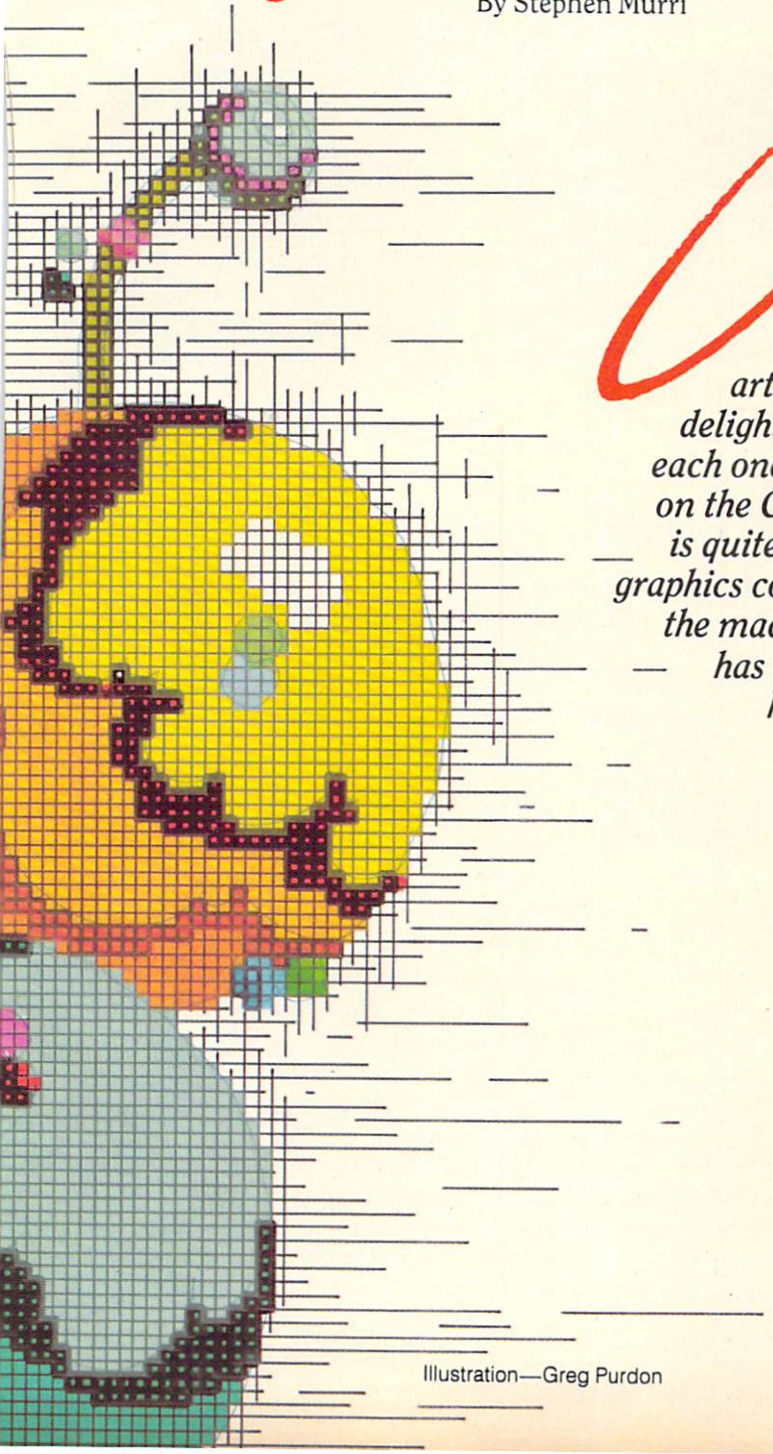
(continued on page 64)

Art

Commodore 64 Gallery

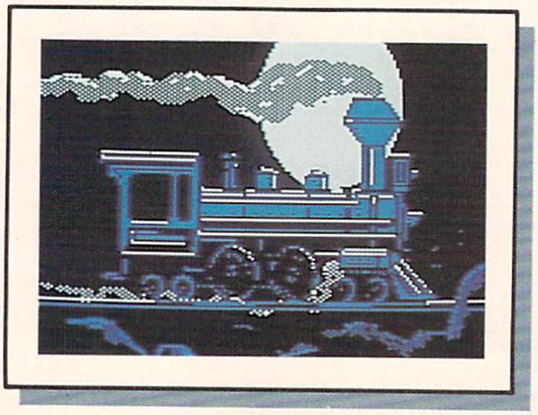
By Stephen Murri

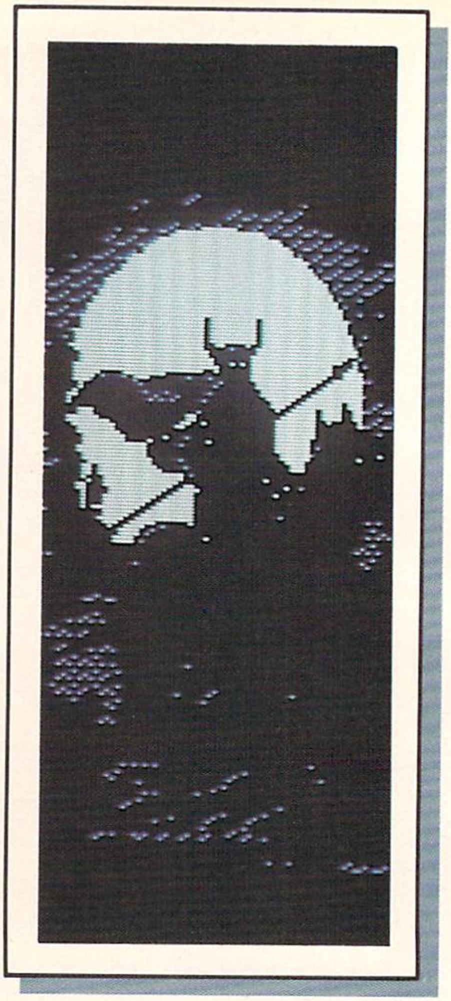
From the looks of the ever-increasing amount of imagery being developed on the Commodore 64, there appears to be no doubt that the machine has become a definite hit with graphic designers and computer artists. It seems that every day we are delighted with another dazzling graphic, each one better than the last. Art is blooming on the Commodore 64. The reason, of course, is quite simple. The instant appeal of computer graphics coupled with the sophisticated capabilities of the machine, make it an ideal medium which has been tempting even the most hard-core traditionalists.

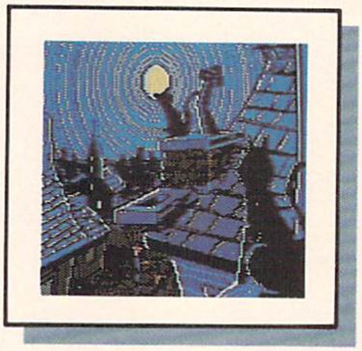


Since this issue of *Commodore* magazine is devoted to graphics, we thought it would be nice if we could give you a sampling of the various types of graphics being created on the Commodore 64. We selected the works of five highly imaginative artists, each with varying combinations of techniques and approaches. Some of it is hybrid and some is exotic, but all of it represents the beginning of a new and unique artform which transcends the relationship between the artist and his or her work. So, without further ado, here is the Commodore 64 Art Gallery.

Illustration—Greg Purdon

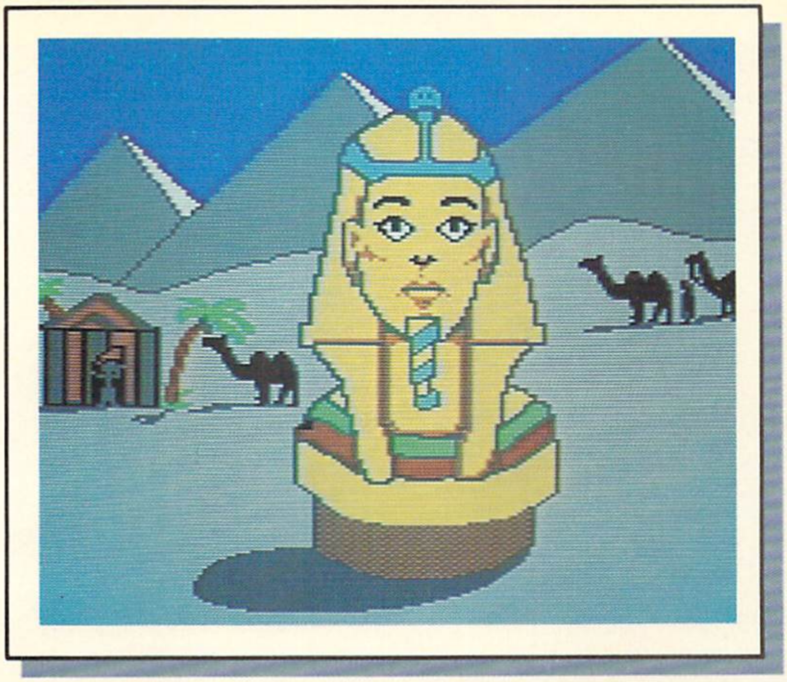


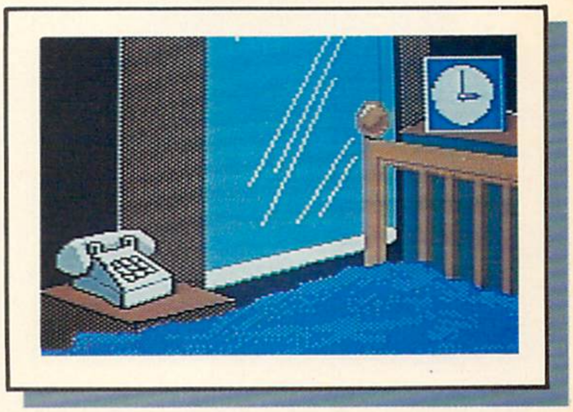


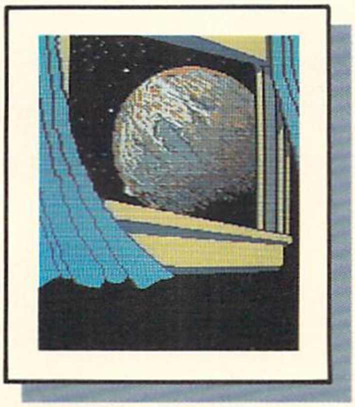


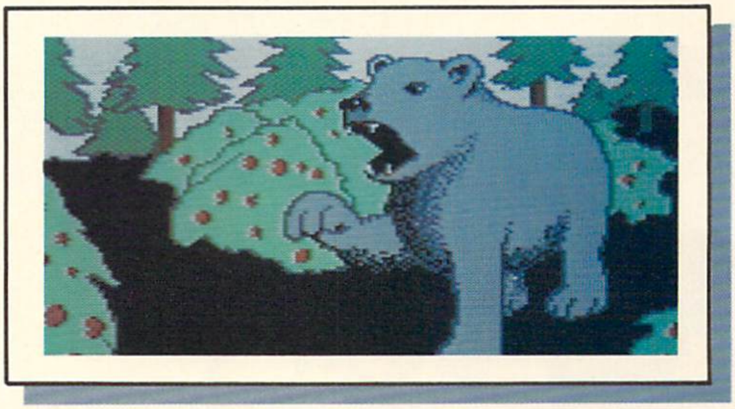
Art Huff

Art Huff is currently the art director at Datamost in Chatsworth, California. He oversees packaging, advertising, and is also a game designer and computer artist. Huff has been "playing with art" all his life in a variety of stages and mediums. His background includes a number of years at Walt Disney Studios, where he worked as an artist and model maker. Huff purchased his first computer in 1978 and has been hooked ever since. He has tried everything that is possible on a small computer, ranging from graphic cartoon-like images to dramatic, illusionary abstracts. A sample of Huff's most recent works are shown below. Most images were created with custom software packages interfaced to a graphics tablet.



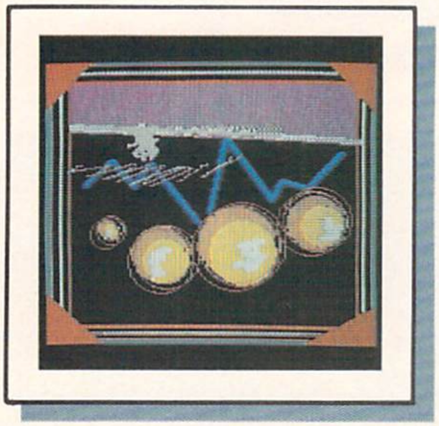




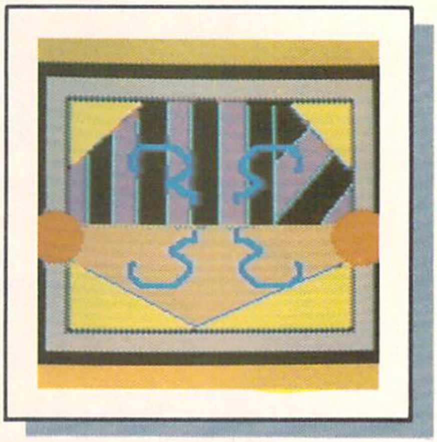


Camelot

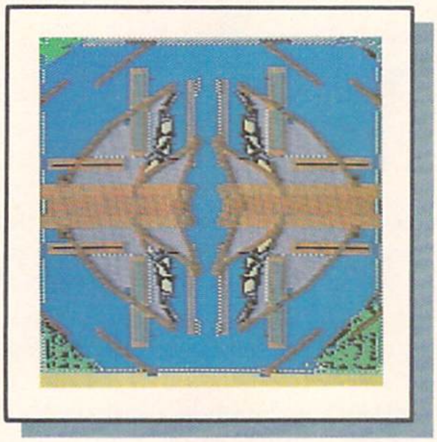
Camelot is a software development firm specializing in original software for the Commodore 64. About two years ago, Camelot decided to attack the problem of developing a system for effective computer graphics. Their solution was a team approach. They began training experienced artists to work in association with experienced programmers. These efforts have culminated in finished products which are truly state-of-the-art. Commodore will soon be releasing two products developed by Camelot, *Viduzzles* and *Birthday Game*. The graphics on this page were designed by the artists at Camelot under the direction of Ralph McGeehan, Video Art Director.

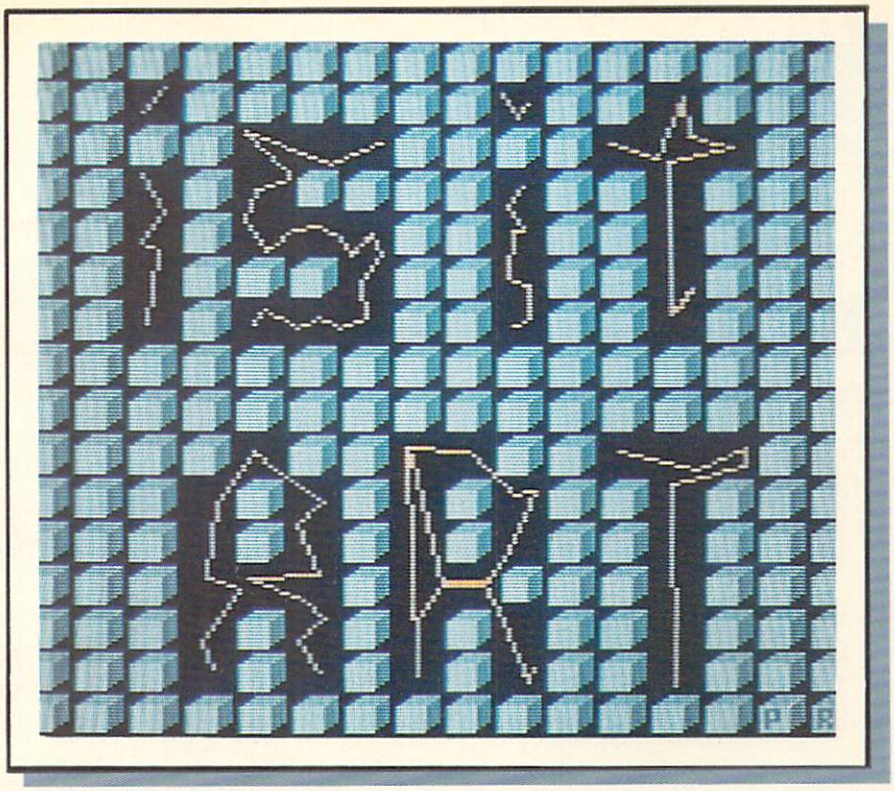


Trip Denton

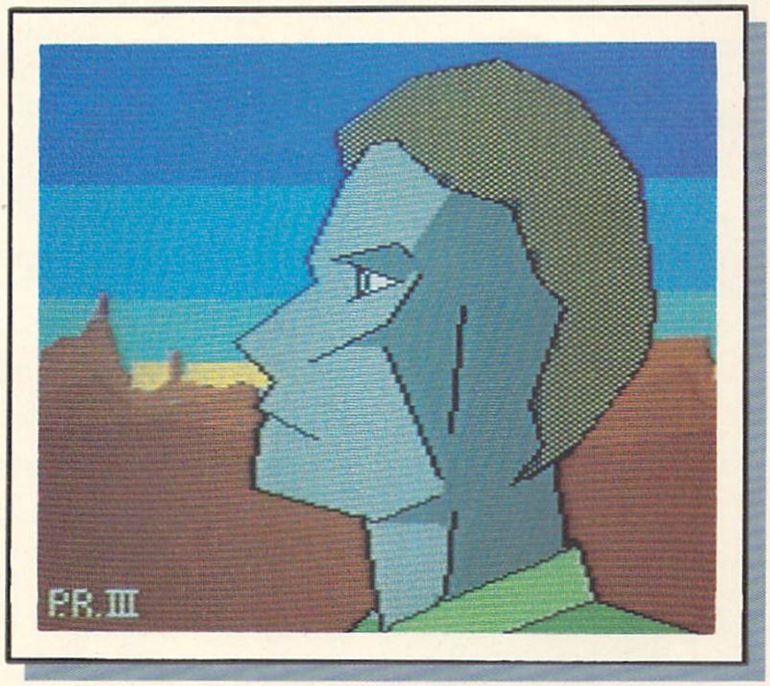


Trip Denton is a painter-turned-computerist who has recently become known for his success in combining the often opposing forces of fine art and technical science. Denton has a degree from the Pennsylvania Academy of Fine Arts and teaches computer graphics at Philadelphia College of Art. His work has appeared in a number of exhibits in the Philadelphia-New York area. He is best known for his experimentation in abstract, dynamic art, where an initial image is constructed, then gradually changes either through random incoherence or actual environmental influence. Denton initially pioneered his work on the VIC 20 but has more recently switched over to the Commodore 64. His most recent works, created with the Koala Pad, are shown here.



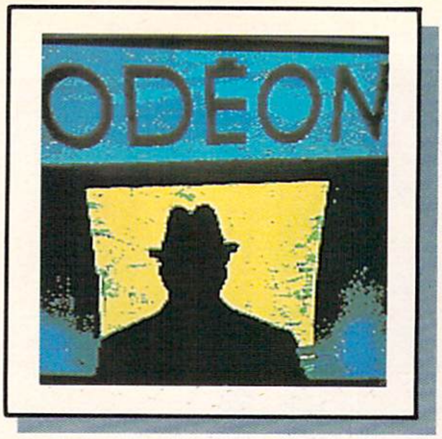


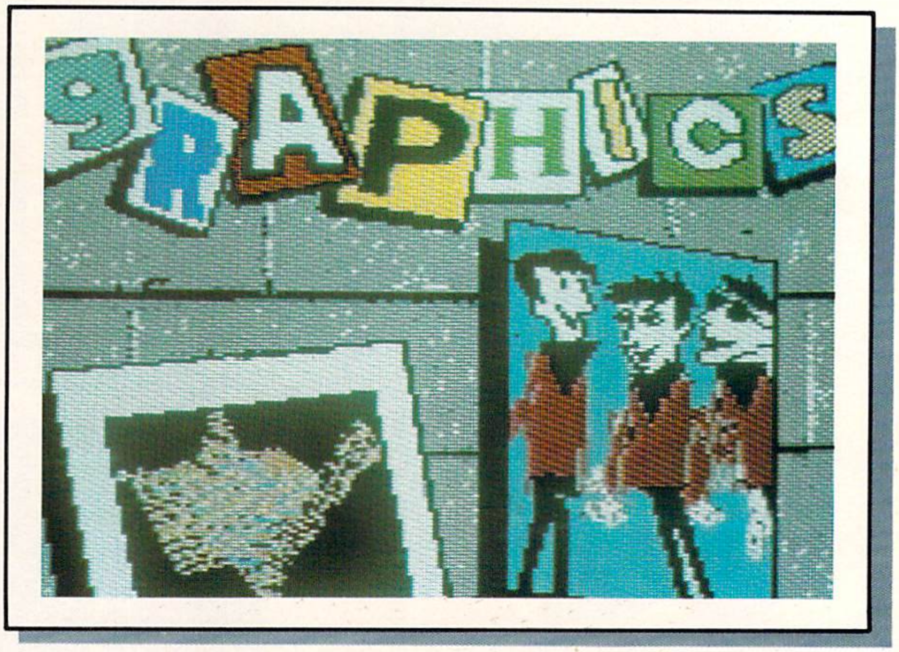
PAUL REICHE

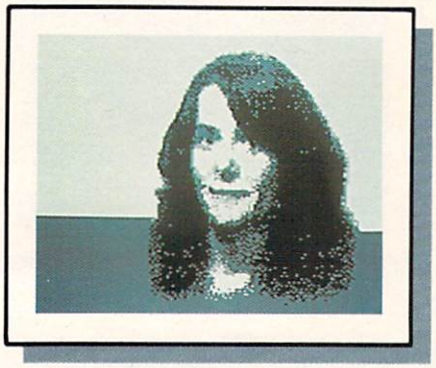


Paul Reiche

Paul Reiche is primarily a game designer and computer artist. Reiche has studied modern art and electronics at the University of California, Berkeley. It is probably this background that has contributed many of the space-like images that Reiche is known for. Reiche is currently working with Electronic Arts. His most recent accomplishment is the video game *Archon*. Reiche formerly worked for Automated Simulations and TSR Hobbies. The following are three screens which were designed using a graphics pad and software designed by Island Graphics. These images were specially designed for this article.







Stephen Murri

Stephen Murri currently works in Commodore's Software Division. His special interest area is computer graphics and he has done much experimentation with graphics on the Commodore 64. Murri is no stranger to the world of graphics. A few years ago, he took a short sabbatical from his computer career to open a commercial photographic studio specializing in color printing and album cover design. The photo-like images on this page were created with a homemade digitizer originally designed by Dr. Frank Covitz for the PET. (See page 41 for a more detailed description of the hardware.) Murri borrowed the digitizer from Covitz and altered the software to directly convert any photographic image to a multi-color high-resolution display on the Commodore 64. The images were then colored, and in some cases, pasted together with a graphics pad. C

Digitizing: How It Can Be Done

By Frank Covitz

On page 14 you'll find a digitized photo of our editor (which greatly improves her looks), created by Steve Murri on the Commodore 64. In addition, on page 40 are several more examples of Steve's digitized work. Here Frank reveals how Steve managed to create these screens.

Imagine you have a pen and ink plotting device that can be controlled by a computer. Next, throw away the pen and mount a photo detector in its place. Imagine that you also have a "black box" that converts light values from the photo detector to a digital value that your computer can read. At this point you have the capability of using your plotter, which is normally a graphic output device, as a graphic input device. The computer can position the "pen" where you want it and then read the light value. What you do with the number so obtained is up to you. The catch, of course, is that you have to develop the programs (and put together the hardware) to implement this wild and crazy idea.

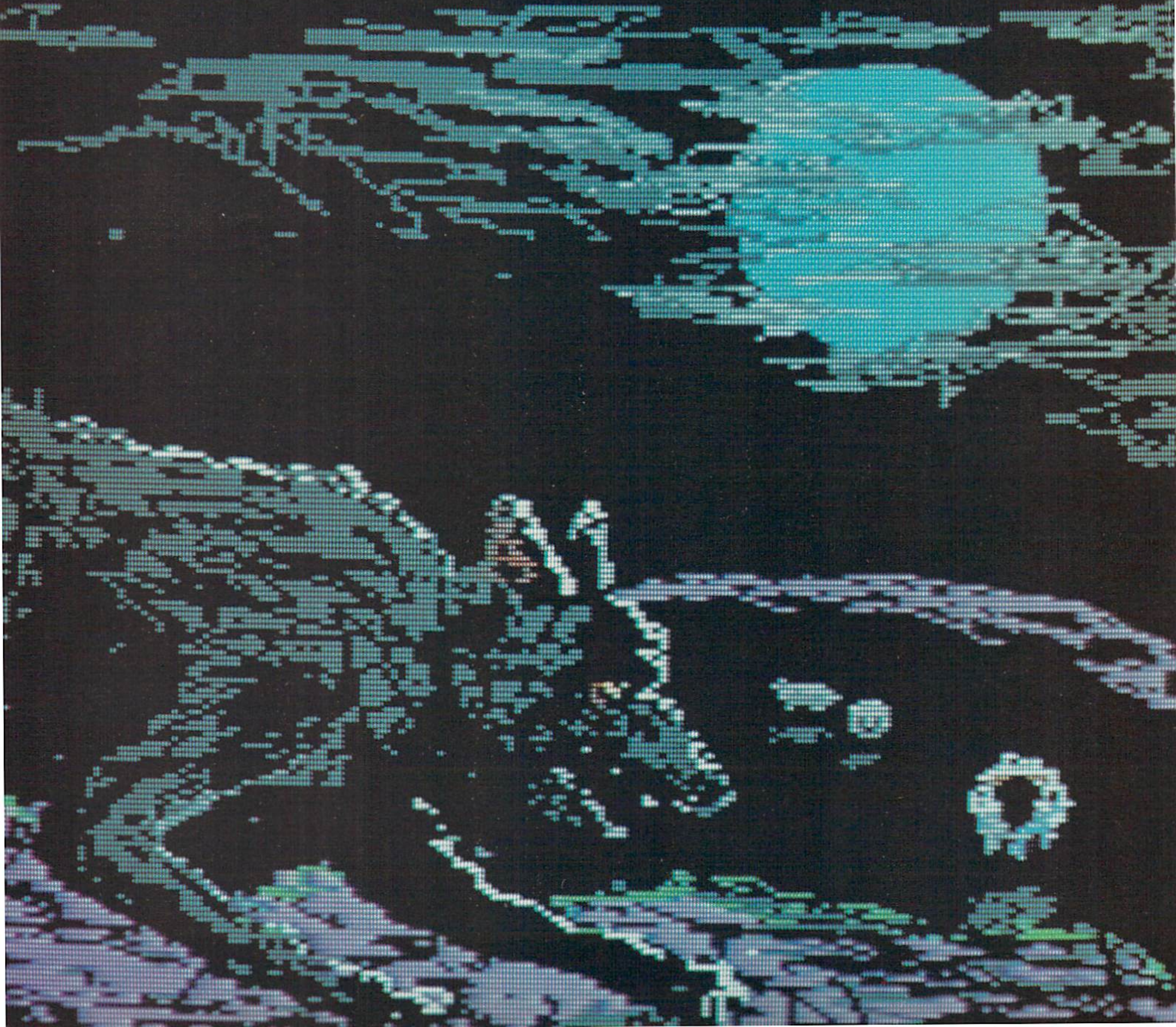
Steve Murri's graphics were obtained using the above technique. The hardware consisted of: 1) A Houston Instruments Hiplot digital incremental plotter (using its RS-232 mode at 4800 baud). 2) A TNW-2000 IEEE-488 to RS-232 device. 3) A Commodore 64 IEEE-488 adapter. 4) A Sharpie fine tipped felt pen with the felt removed. 5) A Radio Shack photo-resistor mounted within the top portion of the pen. 6) A miniature incandescent light bulb (origin unknown) taped to the bottom portion of the pen, to provide constant illumination of the region just below the tip of the pen. 7) A homebrew resistance-to-voltage opamp circuit. 8) An Intel ADC-08 eight-bit A-to-D module (I believe the 64's SID chip A-to-D converter could have been used for items seven and eight).

In use, the software (which was developed first by myself to run on my PET and later modified by Steve and myself to run on the 64) allows the user to set the lower left and upper right corners of the picture to be digitized, so that the appropriate scale factors can be applied. The 64 then drives the plotter in a raster pattern, picking up light values as it goes. The light values are used to plot on the 64 display. I use a straight bit-map technique with a "dithering" algorithm to give a pseudo gray scale. Steve used multi-color with gray levels. Other plotting algorithms, such as edge detection and contouring are also experimented with. The point is that once you have this graphic input capability under essentially complete software control, you have the potential of producing a wild variety of graphic transformations. The resulting "image" can be modified further (as Steve did) with graphics manipulations packages such as Doodle or Koala Pad. C

Speaking of Michaelangelo

By Eddie Johnson

Since most of us are computer types, we don't often get to hear about computer graphics from an artist's perspective. Just how does a computer measure up as a vehicle for artistic expression? Artist Eddie Johnson answers that question for us here.



Since my introduction to computers a little over a year ago, I have been thinking about the impact that the computer has had on my personal artistic philosophy. As any artist would be, I was singularly unimpressed by my first acquaintance with the home computer. Only eight colors? This was before I met the Commodore family, but still, when compared to the virtually infinite range offered by the traditional painting media, even a sixteen-color pallette seemed almost claustrophobic.

Worse even than that was the cumbersome and exacting tedium of entering vast quantities of cryptic coded messages into a machine that was less forgiving of error than the most demanding English teacher I'd ever encountered. All this in order to produce an image that I could have made with a box of wax crayons in less time than it would take to warm up the TV set.

Yet here I am, a dedicated and enthusiastic computer artist, celebrating my VIC 20's first birthday by writing an article on the personal computer as a means of artistic expression. How did this transformation occur? How can I honestly compare the personal computer to the traditional media as a serious artistic tool?

Well, to begin with, I can't compare it to any other artistic medium that I know of (and I think I know all of them) except in the broadest terms. As with all artistic tools, it has its own capabilities and its own limitations, and therefore has its own very individual "personality".

Eddie created these screens for a game-in-progress for the 64 using a KoalaPad.

The computer also has the greatest artistic potential of any medium that I know—a Renaissance instrument for a Renaissance man or woman. This particular tool in the hands of an artist becomes a canvas, an orchestra, a theater, a typewriter, and a movie camera all rolled into one, while being more versatile than any one of them. What other medium allows you to "write" a painting that moves and changes, while at the same time plays music on instruments that you have "invented" yourself?

I think what has kept a majority of artists from plunging into computer art is their perception of the computer—any computer—as a highly technological, mathematical and very linear "left brain" sort of machine. Artists are very "right brain", intuitive people. We even have trouble communicating with "left brain" *people,* let alone a "glorified calculator".

I used to think that a computer could only do things one way—no other—and artists hate that sort of limitation. Yet the *Personal Computing on the VIC 20* book that comes with the computer describes three ways to do simple animation. And I have since discovered at least a half-dozen other ways of moving objects around on the screen. There are commands designed for word processing that can be used to play music—mathematical functions that can manipulate colors—everything the computer can do can be used to create art. The excitement of exploration and discovery is unmatched by any other creative activity that I know.

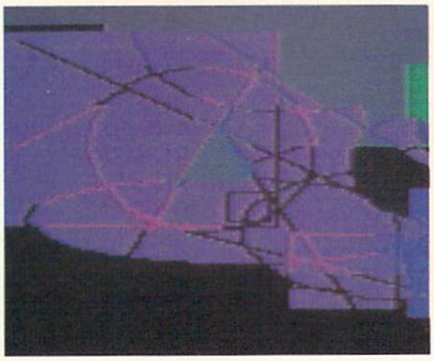
I believe that another reason that the artistic potential of the home computer hasn't been widely exploited as yet is partially due to the fact that any artists who are at all interested in computers are mostly lusting after that nearly impossible dream of getting their paint-stained hands on a multimillion dollar mainframe, and overlooking a central fact in artistic creation—it's the *limitations* of a medium that establish the parameters of creativity. An artist who can create art on a piece of paper with a piece of burnt wood can most assuredly create art on a microcomputer, and it presents as great a challenge and as great a reward.

"Yeah," you say, "we've seen

what can be done with a ten million-dollar CRAY-1 computer, but a VIC 20? Sure, they're fun, but *art*?" Well, it's true there hasn't been, as yet, a J. S. Bach of the home computer (remember, though, that he worked with quill pen and paper!), but I have written a few random graphics programs for small computers (including the VIC 20) that have glued audiences to the screen for periods ranging up to a couple of hours. These are people who will, on occasion, walk out on a multi-million-dollar Hollywood movie. Yeah, there's a lot of power in even that pea-brain five-kilobyte machine!

Of course, it's not necessary for an artist to be a programmer to use the computer as an artistic tool. There are, for instance, graphics utilities that do for the visual artist what a word processor does for the writer. These programs can be operated by the keyboard, joystick, light pen or a whole family of pressure-sensitive drawing pads. My personal favorite, the KoalaPad, falls into this last category, and I will use it as an example.

The programmers at Audio Light, Inc. have written a software program for the KoalaPad and the Commodore 64 that allows the computer to be used by an artist with no more technological skills than those needed to operate a phonograph. This in itself would not be sufficient recommendation, if it were not for the fact that the program allows the artist to create a work of astonishing richness and subtlety. As with the traditional painting media, it permits artists of widely differing styles and philosophies to express their varied ideas in ways suited to their own personal visions.



The DooDah Da Da Museum, a random-design program for the VIC 20.

But why even use a computer when oil paints (or acrylics, etc.) already provide the means to this end? I am reminded of what happened to me when I began working with video a few years ago. As an old-time "underground" filmmaker, I thought the transition to videotape would be no more than learning a new technology, since film and video seem so similar on the surface. Not true! The relationships were there, but the aesthetics were totally different! It's the same with the "Koala Painter"—superficially, you're making a "painting", but as with video as compared with film, the computer allows much greater speed and flexibility, giving the image a peculiar intensity and immediacy.

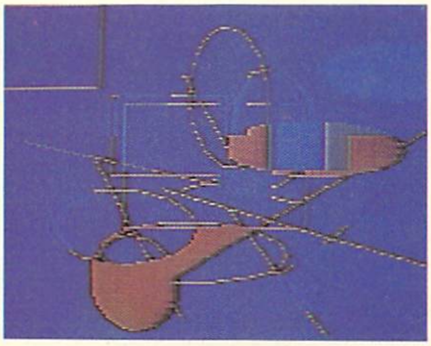
As a "graphics processor", it provides an amazing capability to "edit" a painting in a variety of ways—changing colors at the push of a button, zooming in to edit pixel-by-pixel, erasing mistakes as though they never existed. And perhaps best of all, it makes it possible for the artist to "save" the work at various stages of completion, so there is never the fear of experimentation—going too far and ruining the painting irrevocably. Try that on your watercolor sometime! The risk of failure, which fuels the artist and creates the excitement in the work, can then rightfully be relegated to the risk of failure of the idea, rather than a failure of the technique.

Of course, these "user-friendly" graphics systems are not the only way to create art on the computer. In fact, I think that any artists who overcome their initial hostility towards the intensely "left brain" activity of programming (it took me less than a week), will find the challenge of actually "communicating" with the machine one of the most rewarding activities available to the creative personality. The total involvement of all of one's mental capabilities explains to me quite clearly the addictive quality of computer hacking. There is a very real magic here—I will often find myself looking at a program listing, just marveling at the fact that all those arcane instructions will be wondrously transformed into pictures and sounds when I run the program.

The real clincher for me came when I discovered the random number generator in the computer. Since art comes largely from the random juxtapositions in the human subconscious, it was a great thrill to find an artistic medium that had its own "subconscious"! This means that an artist can write into a program the "happy accidents" that artists have traditionally relied upon for inspiration and growth. This makes it possible to write a program that will produce a virtually infinite number of original art works.

I have created one such program that I call "The Doo Dah DaDa Museum" (Taking screen shots of the program is like trying to convey the feeling of a modern dance piece with a couple of still photographs. Talk about the limitations of a medium!). The randomizing elements have been laboriously constructed with limited parameters, so that, in spite of the unlimited number of pictures and sound sequences, there is an underlying structure proportioning the piece. I have discovered that a totally random piece can be just as boring as one that is totally structured.

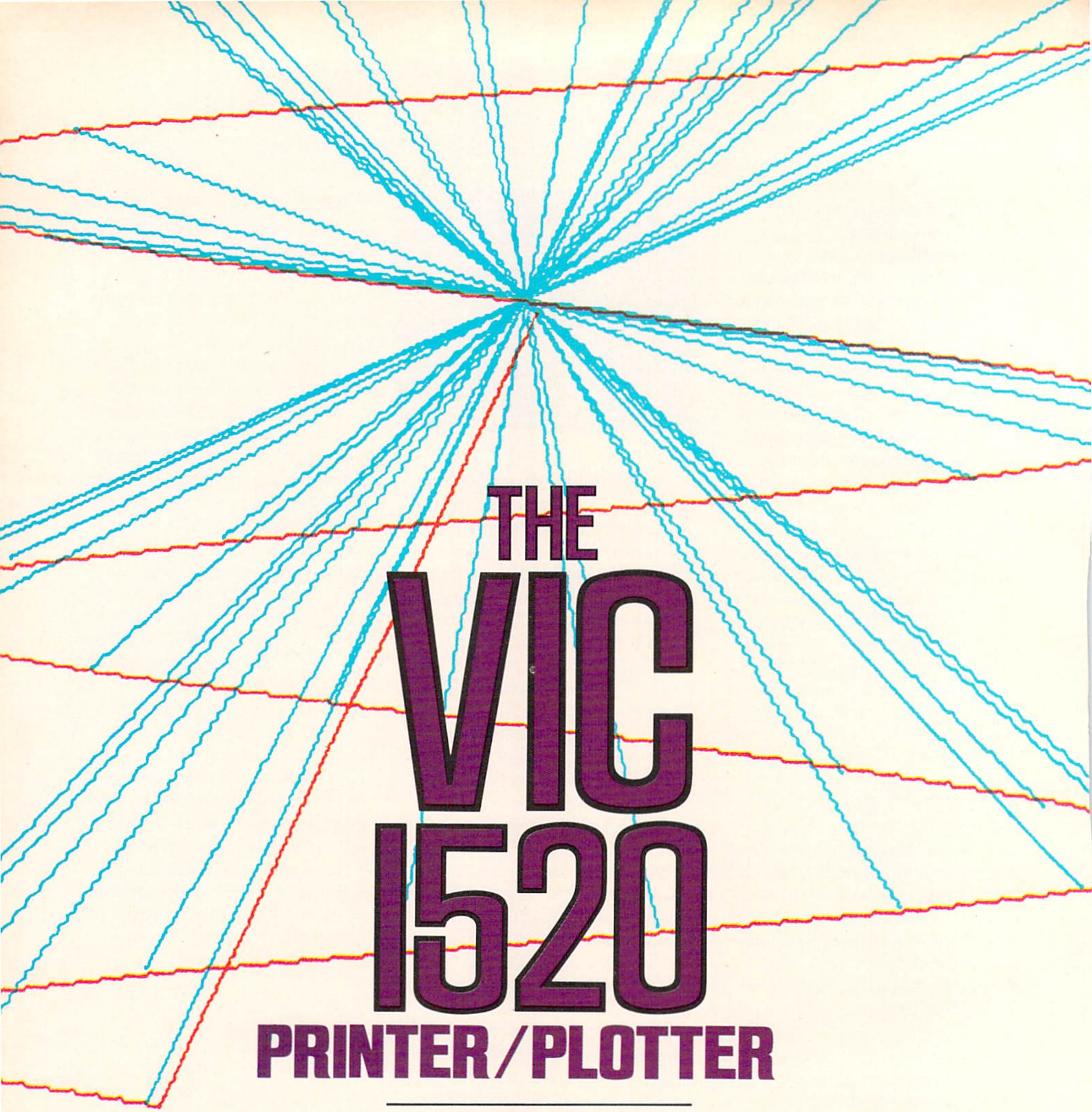
As for computer programming being cut-and-dried and mechanical, I have become aware of individualistic differences in programming styles, so that I can begin to recognize certain programmers the way I can recognize different sculptors or different composers. It is possible to construct a program just like constructing an expressionist painting—one brush stroke at a time, letting each new addition inspire the next move. It is also possible to take a "hard edge" approach—knowing exactly what you want and getting it! In short, computer programming allows all the freedom and demands all the discipline of any of the traditional forms of creative expression.



Expression. Communication. Aside from TV and paperback books, I can think of no other creative tool that can place art in the homes of more people than can the personal computer. It is possible to market a computer program for a tiny fraction of the price of an oil painting, for instance, and it will have an advantage over even a print of the painting, because, in a very real sense, it is an *original* work of art that has not suffered in the slightest the way the painting has by being translated from canvas and paint to paper and ink. Just think—your audience can see your work the way *you* saw it—the way you want them to see it!

I have noticed when reading software reviews that whenever computer graphics are mentioned, the writer is usually referring to some sort of video game. There doesn't seem to be much exploration of the personal computer as a purely artistic tool. I often think about the artists who I believe would have pawned their paint boxes to get their hands on even a little VIC 20—Picasso, Kandinsky, Stuart Davis, Malevich, Mondrian, Duchamp, Da Vinci, the list goes on and on. And those are just the painters. Or would they be writing game programs, too? **C**

Eddie Johnson is a painter, sculptor, photographer, video and film producer—and computer artist—who lives, works and teaches in Albuquerque, New Mexico. He has a Master of Fine Arts degree from Pratt Institute and has taught art at Pratt, Cornell University and Rutgers University.



THE VIC 1520 PRINTER/PLOTTER

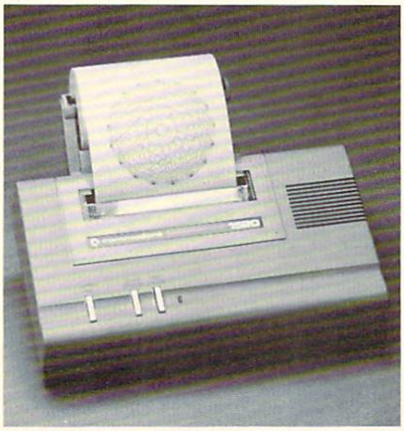
By Jim Gracely

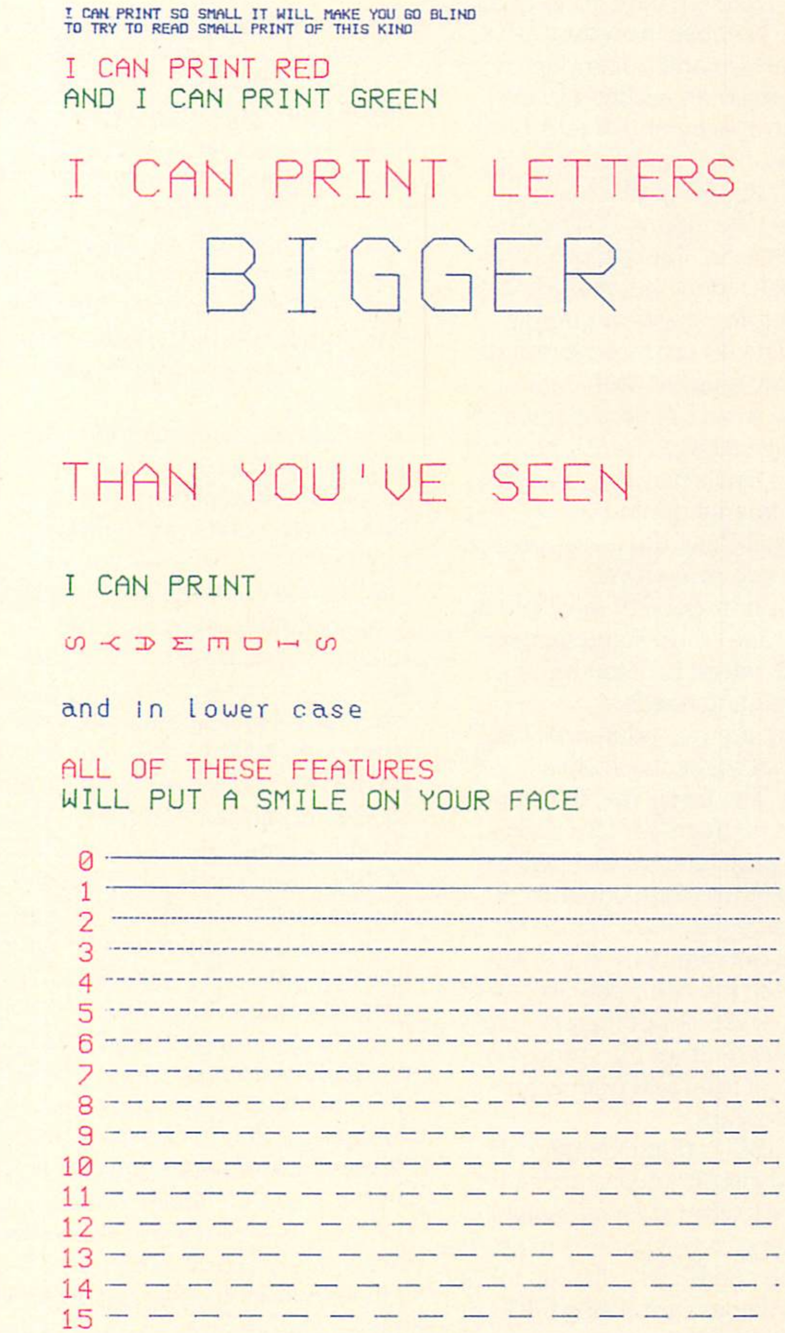
Commodore has a new printer/plotter for use with the VIC 20 and the 64. Although the word "printer" appears first in the name, I only thought of it as a plotter. In fact, it wasn't until after I read the manual that I even realized that it could print! In this article, we'll examine some of the features of the 1520 and present some of its better points.

What Is a Printer/Plotter?

The main difference between a dot matrix printer and a printer/plotter is in the way it prints. A dot matrix printer has a block of tiny hammers (5 × 7 on the 1525) that hit the printer ribbon and make marks on the paper. At the end of each line, the paper is advanced and a new line is started.

The 1520 printer/plotter, on the other hand, has four small pens held in position off of the paper. When something is to be printed, one of the pens is brought in contact with the paper. Because only one pen is used at a time, the paper must be moved up and down to make lines. If you watch a printer/plotter while it prints a word, you will notice that it actually draws each line of each letter.





```
I CAN PRINT SO SMALL IT WILL MAKE YOU GO BLIND
TO TRY TO READ SMALL PRINT OF THIS KIND

I CAN PRINT RED
AND I CAN PRINT GREEN

I CAN PRINT LETTERS
      BIGGER

THAN YOU'VE SEEN

I CAN PRINT

SIDEWAYS

and in lower case

ALL OF THESE FEATURES
WILL PUT A SMILE ON YOUR FACE

 0 ________________________________________
 1 ________________________________________
 2 ........................................
 3 ........................................
 4 ----------------------------------------
 5 ----------------------------------------
 6 - - - - - - - - - - - - - - - - - - - -
 7 - - - - - - - - - - - - - - - - - - - -
 8 - - - - - - - - - - - - - - - - - - - -
 9 -  -  -  -  -  -  -  -  -  -  -  -  -  -
10 -  -  -  -  -  -  -  -  -  -  -  -  -  -
11 -  -  -  -  -  -  -  -  -  -  -  -  -  -
12 -   -   -   -   -   -   -   -   -   -   -
13 -   -   -   -   -   -   -   -   -   -   -
14 -   -   -   -   -   -   -   -   -   -   -
15 -    -    -    -    -    -    -    -    -
```

Figure 1. The 1520 as Printer

The VIC 1520

The 1520 printer/plotter comes all ready to start printing and plotting. Many people think that the 1520 is larger than it really is. It is about 10½" wide, 6" deep (9" including the paper roll) and 2½" high. My trusty assistant has demonstrated to me that the 1520 takes up the same table space as two diskettes laid side-by-side.

In addition to the printer/plotter itself, you get everything else you need to start. You get a serial cable that plugs into the serial port on either the 64 or the 1541 disk drive. You also get one roll of paper (4½" wide) and four pens (one each, blue, black, green and red). The pens are easy to install and the paper is on a roll, mounted to the rear of the printer. In just a few minutes the printer/plotter is ready to go.

There is one major difference between the hardware configuration of the 1520 and all of Commodore's other printers—it is device number six (not four). This is a little confusing at first, but it doesn't take long to get used to.

The 1520 as a Printer

The 1520 is a very good quality printer, because the letters are actually drawn and not impacted on to the paper. In addition to this, you can specify any of four printing sizes. The four printing sizes are: 10, 20, 40 or 80 characters per line (see Figure 1). The ten character-per-line size produces big block letters that make great headings for charts or graphic plots. The 40 character-per-line size is my favorite, with characters that are small but very clear, and fill the line nicely. The 80 character-per-line size is nice if you want a lot of material printed on one line and is still clear, but it is approaching the eye strain level.

The print speed of the 1520 is an average of 12 characters per second, which isn't too bad for most printing needs.

There are two additional features that make the 1520 a good printer. The first is that text can be rotated 90 degrees. This allows you to print letters which will be right-side-up when you turn the paper sideways (see Figure 1). The second feature is the ability to choose the color pen you want to use. The color of the pen can be selected either by a button on the top of the 1520 or through your program.

The 1520 printer/plotter can be used just about as any of Commodore's other printers is used. The listing of programs is the same and you can PRINT directly to the printer if you want (using CMD or PRINT#4). You cannot create custom characters like you can on the 1515 or 1525, but you can program your own characters if you want (it would take a lot of work though. More about this in the plotter section).

The 1520 as a Plotter

In my opinion, the 1520 really begins to shine when the subject becomes plotting. This plotter can draw some really nice graphic designs, and everything can be drawn in four colors with no dots!

The plotter paper is set up like a big grid and you can move or draw from anywhere to anywhere on that grid. The grid has 480 blocks across the paper and a +/− 999-block range on the paper vertically. This is a large working surface. It represents almost 960,000 individual points that each pen can be moved to (from now on I'll refer to them as plotter pixels). The high resolution screen of the 64 is only (only?) 64,000 pixels, so we have 16 times as many pixels to work with.

One of the nicest plotting features is the ability to use both absolute and relative positioning. The position of the pen when the plotting file is opened is the absolute "home" (X=0:Y=0) position. Any absolute moves or draws are relative to this position. You can also set any other point as a relative home position, and any relative moves will be relative to this point. This allows you to move the relative home position from the left side of the paper to the middle of the paper. The hours of grief that this will avoid can only be appreciated by those of us who have worked with high-resolution graphics on the VIC or 64.

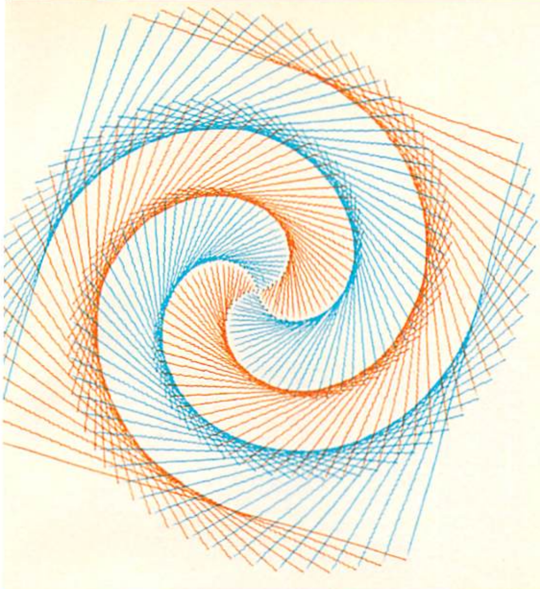
Plotting graphs or graphics on the 1520 is nothing more than drawing lines between points. Anything that you can draw on a piece of graph paper you can plot on the 1520. Of course coming up with things to plot is not the easiest task to accomplish. For those of you who already have the 1520, I have included three of my favorite plot programs to get you started.

In conclusion, I must say that I found the 1520 much more fun to work with than I had first expected. The paper is just the right width for printing out directory listings, and even some machine language code. The plotting capabilities had me pouring over designs I had previously created, translating them to the 1520 (and they were much shorter in every case!). All in all I think that it is a great little printer. **C**

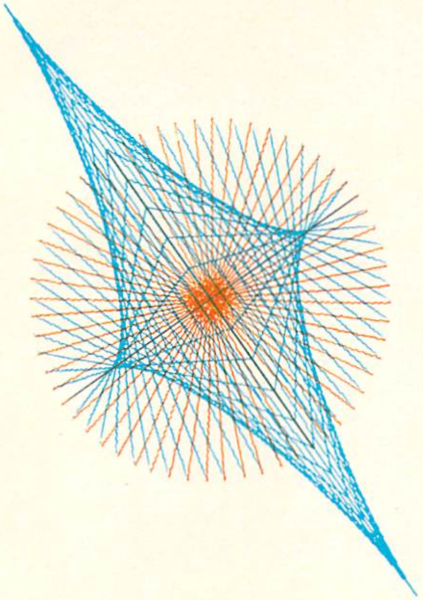
Cubic Spiral

Graphix

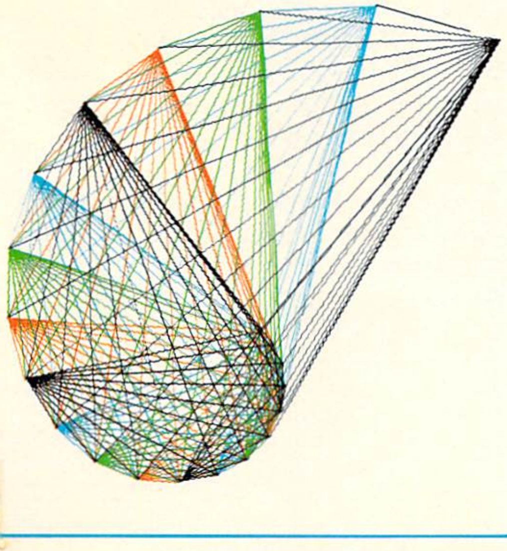
Geosnail



```
5 OPEN 1,6,1:OPEN 2,6,2
6 PRINT#1,"M";240,-240
7 PRINT#1,"I"
10 I=240:A=-5:S=1000:M=S+1:Z=1
12 PRINT#1,"R";0,I
13 A=A+5:T=A*[PI]/180
15 X1=I*SIN(T):Y1=I*COS(T):GOSUB 60
20 X1=-I*COS(T):Y1=I*SIN(T):GOSUB 60
25 X1=-I*SIN(T):Y1=-I*COS(T):GOSUB 60
30 X1=I*COS(T):Y1=-I*SIN(T):GOSUB 60
35 Z=4-Z:PRINT#2,Z
40 IF I>=4 THEN GOTO 13
50 PRINT#1,"R";0,-240:CLOSE 1:OPEN 7,6,7:
   PRINT#7:CLOSE 7:END
60 PRINT#1,"J";X1,Y1
100 I=I-1:X2=X1:Y2=Y1:RETURN
```



```
5 OPEN 1,6,1:OPEN 2,6,2
7 PRINT#2,1
10 R=120:P=0
12 PRINT#1,"M";240,-240
13 PRINT#1,"I"
15 FOR I=0 TO 354 STEP 6
20 X1=0:Y1=0
25 T=I*[PI]/180
30 X2=R*COS(T):Y2=R*SIN(T)
40 PRINT#2,3
50 PRINT#1,"R";X1,Y1
60 PRINT#1,"J";X2,Y2
110 P=1-P:IF P=0 THEN 150
115 PRINT#2,1
120 X3=X1+2*(X1-X2):Y3=Y1
130 X1=X2:Y1=Y2:X2=X3:Y2=Y3
140 GOTO 50
150 NEXT
155 PRINT#1,"R";0,-200
160 OPEN 7,6,7:PRINT#7:CLOSE 7
170 CLOSE 1:CLOSE 2
```



```
1 OPEN 1,6,1:OPEN 2,6,2
2 DIM X(20),Y(20)
3 PRINT#1,"M";350,-200
4 PRINT#1,"I":PRINT#2,1
5 Z=0:A=.34
10 FOR T=3.5*[PI]TO 1.5*[PI]STEP-.3
20 R=EXP(A*T)
30 X(Z)=SIN(T)*R*1.6*5
40 Y(Z)=COS(T)*R*8
50 Z=Z+1:NEXT
100 FOR I=0 TO 19:FOR J=I+1 TO 20
110 PRINT#1,"R";X(I),Y(I):PRINT#1,"J";X(J),Y(J)
115 PRINT#2,I AND 3
120 NEXT:NEXT
130 PRINT#1,"R";0,-300
140 OPEN 7,6,7:PRINT#7:CLOSE 7
150 CLOSE 1:CLOSE 2
```

Antialiasing

By Brooks Cooley

Jagged lines are caused, in part, by the approximations involved in attempting to draw a line on a discrete grid of points. Figure A shows the difference between a line drawn from point A to point B, and an approximation of that line drawn on a discrete grid of points.

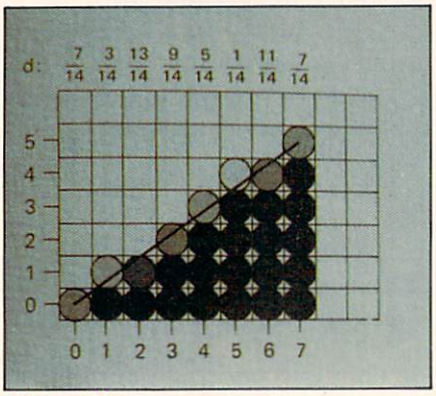


Figure A.

Antialiasing, or dejagging techniques, can be implemented to produce lines that appear to have smooth edges. Antialiasing is quite simple in theory, but can be very time consuming in high resolution environments. For example, doubling the resolution from a typical 512 to 1024 quadruples the number of pixels, and doubles the computation time needed to antialiase a line. In antialiased lines, each pixel overlapped by the line must have an intensity proportional to the area of the pixel covered by the line.

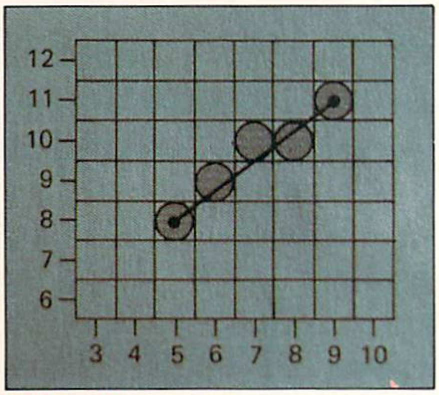


Figure B

Figure B shows a line drawn from (0,0) to (7,5). A straight line has been drawn on the approximated line to show the area of the pixel covered by the line. The resulting area is shown above the graph in Figure B. For example, the line covers 50% of the area of pixels (0,0) and (7,5). Therefore, these pixels would be displayed at 50% of full intensity. Pixel (5,4) would be displayed at 1/14 intensity and so on.

Program 1 shows the basic structure of an algorithm for computing the pixel values of the line y=mx. This algorithm assumes m is the slope of the line, and m is in the range of 0 to 1. This algorithm can be easily generalized to compute the pixel values for any line in the form of y=mx.

Figure C shows the results of lines displayed with and without antialiasing ("a" and "b", respectively).

Lines and Polygons

Program 1.

```
10 REM  * SIMPLE ANTIALIASING ROUTINE *
20 REM  *                             *
30 REM  * WRITTEN BY: BROOKS COOLEY   *
40 REM  * FOR: COMMODORE 64           *
50 REM  * DATE: DECEMBER 1983         *
60 REM  *                             *
70 REM  * THIS PROGRAM CAN EASILY BE  *
80 REM  * ADAPTED TO HIGH-RESOLUTION  *
90 REM  * MULTICOLOR MODE.  THIS      *
100 REM * ILLUSTRATES THE USE OF      *
110 REM * ANTIALIASING ROUTINES FOR   *
120 REM * LINES AND POLYGON EDGES.    *
130 REM
140 PRINT"▓▓":POKE53280,0:POKE53281,0
150 XE=27       :REM  * ENDING X VALUE   *
160 M=5/7       :REM  * SLOPE IS M=Y/X   *
170 REM ** PIXEL COLORS IN ORDER OF **
180 REM ** INCREASING INTENSITY     **
190 CL(0)=0:CL(1)=0:CL(2)=11:CL(3)=12
200 CL(4)=15:CL(5)=1
210 X=0
220 Y=0         :REM  * STARTING Y VALUE *
230 A=1-M
240 D=.5        :REM  * D=PIXEL COVERAGE *
250 GOSUB 430:REM PLOT PIXEL(X,Y,D)
260 REM ** HORIZONTAL MOVE **
270 IF D<A THEN D=D+M :GOTO 320
280 REM ** DIAGONAL MOVE **
290 D=D-A
300 Y=Y+1
310 REM ** INCREMENT X POSITION **
320 X=X+1
330 GOSUB 430:REM PLOT PIXEL(X,Y,D)
340 IF X<>XE THEN GOTO 270
350 PRINT"▓▓▓▓▓PRESS SPACE BAR FOR ";
360 IFCC=1THENPRINT"DE";
370 PRINT"JAGGED LINE"
380 GETA$:IFA$=""THEN380
390 FORZ=1024TO1060:POKEZ,32:NEXTZ
400 CC=ABS(CC-1):GOTO210
410 REM ** COMPUTE NEXT SCREEN **
420 REM ** PIXEL POSITION      **
430 SL=((1984+X)-(Y*40))
440 POKESL,42
450 IFCC=1THENPOKESL+54272,1:RETURN
460 POKESL+54272,CL(INT((D*10+1)/2))
470 RETURN
```

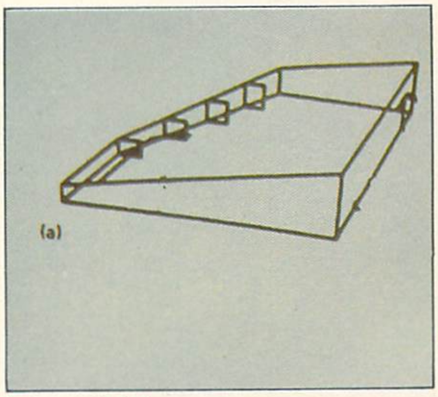


Figure C(a)

Antialiasing of polygons can be generalized from the technique described above. The edge of the polygon determines a unique line. This line can be antialiased using the above algorithm, and mapped onto the edge of the original polygon. Figure B shows a polygon with the edges antialiased using the algorithm above.

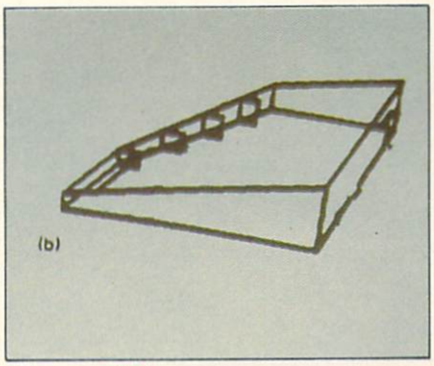
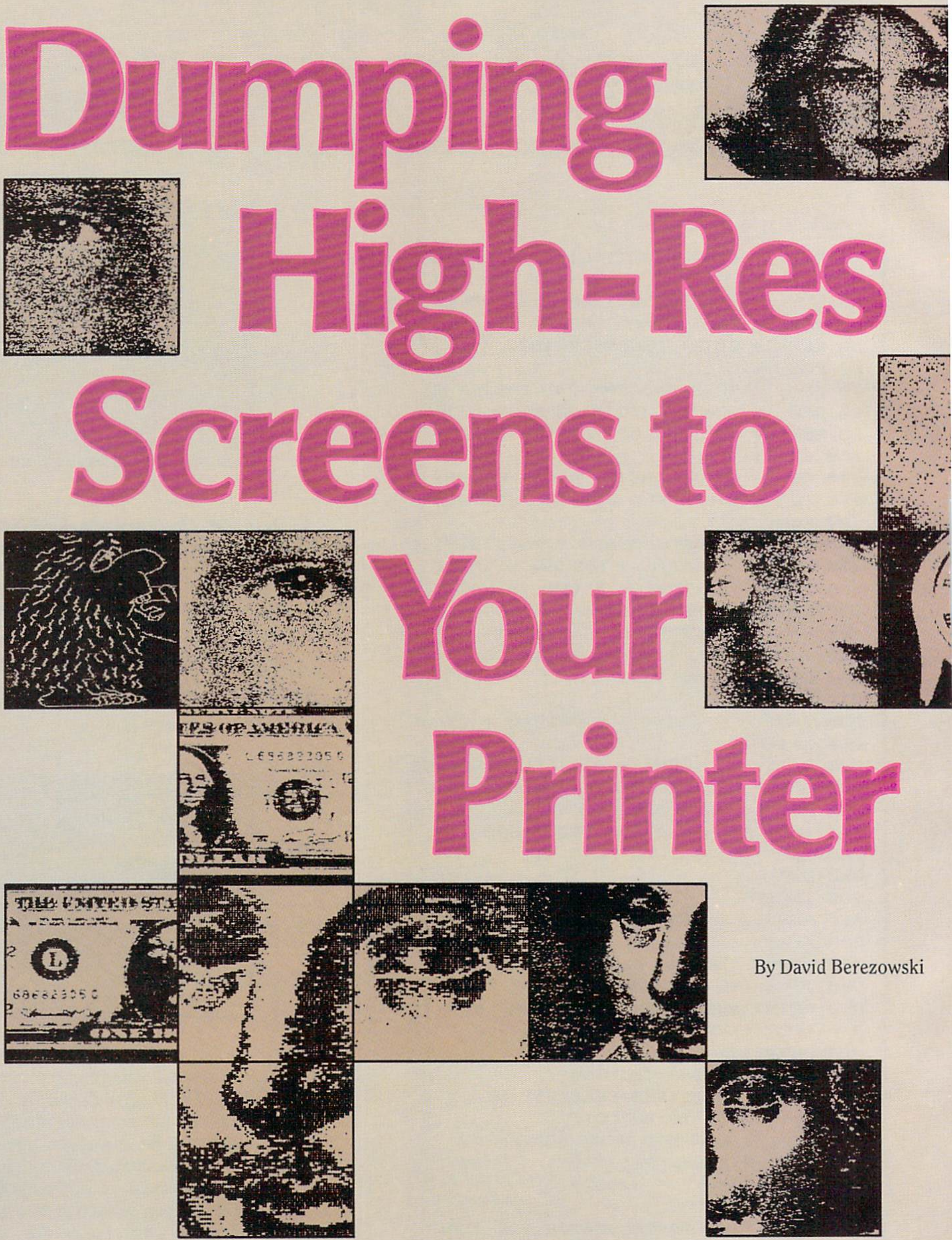


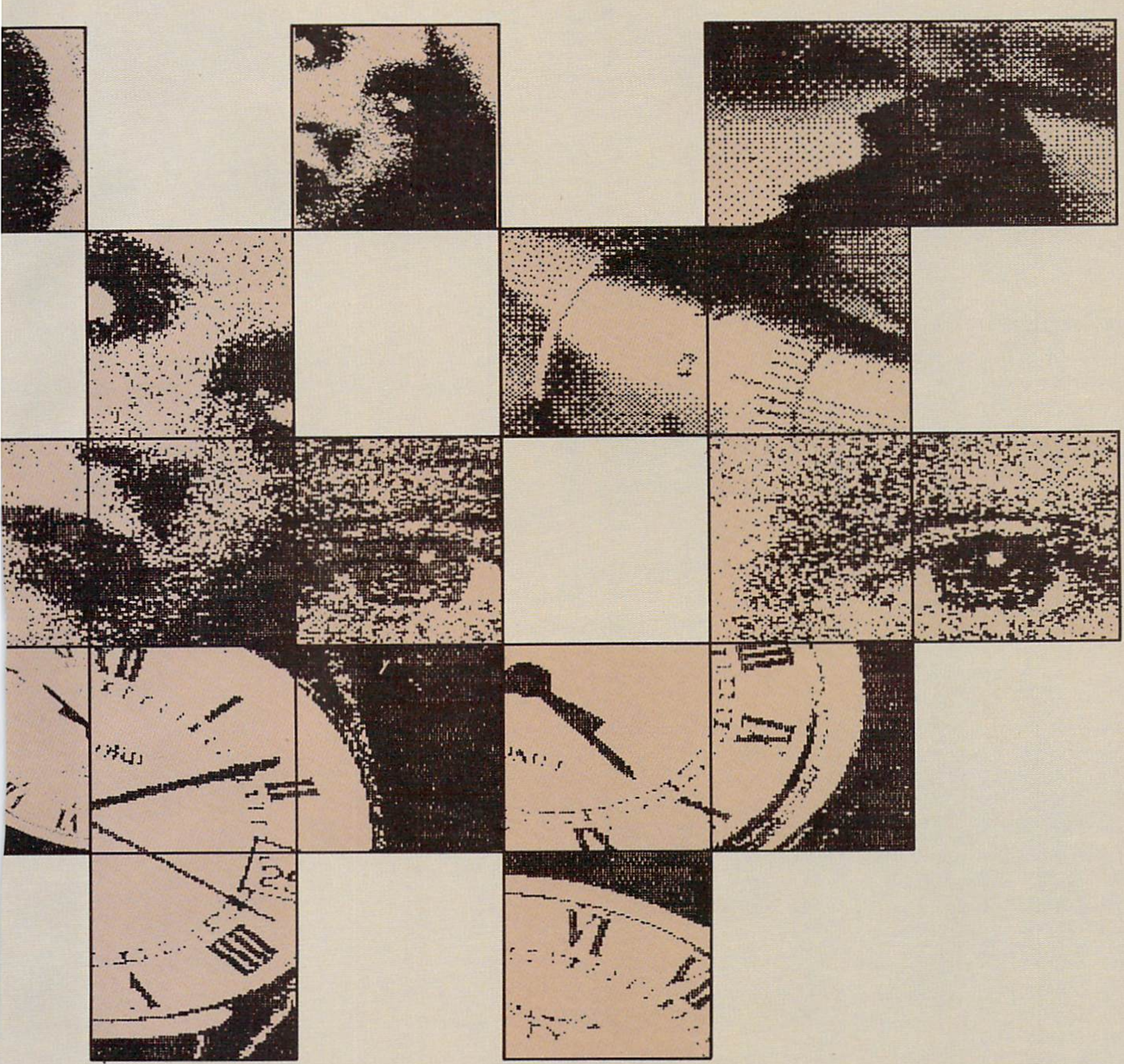
Figure C(b)

Antialiasing has its price. Note that smoothing of the line is achieved at the expense of a slight blurring of the line edges. Even though efficient algorithms have been devised to compute the fractional portion of each pixel overlapped by a line, the process is still too slow for use in general interactive graphic applications. **C**

Dumping High-Res Screens to Your Printer



By David Berezowski



At last you can send printed copies of your favorite high-res screens to your friends and family—if you don't mind sending them in black and white, that is.

This program lets you dump the screens you create on your Commodore 64 from your disk drive to your 8023P printer. It assumes (as you can read in the print statements within the program itself) that a screen is stored as a program file on disk with a load address of $2000 or the screen is in memory from $2000 to $3FFF. It will also work with a PET, VIC 20 or CBM computer. Type in the BASIC loader program and save. Then type in the BASIC program and save under a different name.

To run the program, first load the loader, then load the program itself. You'll notice, beginning with line 360 in the main program, references to a file titled "Karen". This is where you enter the file name of the screen you want to dump. Be sure you type the file name just as it's done in line 360, including the quotation marks: "0:file name".

I've included several examples of the output produced by this program. Each one took about a minute (or less) to print.

Basic Loader

```
10 CHK=0
20 FOR X=7936 TO 8182
30 READ A:CHK=CHK+A:POKE X,A
40 NEXT
50 IF CHK<>27502 THEN PRINT"BAD[SPACE]DATA":END
100 DATA 165, 2, 72, 165, 3, 72, 162, 25, 142, 247
110 DATA 31, 162, 0, 142, 253, 31, 142, 254, 31, 162
120 DATA 40, 142, 248, 31, 162, 0, 142, 251, 31, 142
130 DATA 252, 31, 162, 8, 142, 249, 31, 162, 17, 32
140 DATA 201, 255, 162, 0, 142, 255, 31, 162, 8, 142
150 DATA 250, 31, 162, 0, 134, 2, 162, 32, 134, 3
160 DATA 165, 2, 24, 174, 250, 31, 125, 238, 31, 144
170 DATA 2, 230, 3, 24, 109, 251, 31, 144, 2, 230
180 DATA 3, 24, 109, 253, 31, 144, 2, 230, 3, 133
190 DATA 2, 24, 165, 3, 109, 252, 31, 109, 254, 31
200 DATA 133, 3, 162, 0, 161, 2, 174, 249, 31, 61
210 DATA 222, 31, 240, 13, 173, 255, 31, 24, 174, 250
220 DATA 31, 125, 230, 31, 141, 255, 31, 206, 250, 31
230 DATA 208, 176, 56, 169, 255, 237, 255, 31, 32, 210
240 DATA 255, 206, 249, 31, 208, 152, 173, 251, 31, 24
250 DATA 105, 8, 144, 3, 238, 252, 31, 141, 251, 31
260 DATA 206, 248, 31, 240, 3, 76, 32, 31, 162, 18
270 DATA 32, 201, 255, 169, 13, 32, 210, 255, 32, 204
280 DATA 255, 32, 228, 255, 201, 3, 240, 25, 173, 253
290 DATA 31, 24, 105, 64, 144, 3, 238, 254, 31, 141
300 DATA 253, 31, 238, 254, 31, 206, 247, 31, 240, 3
310 DATA 76, 19, 31, 32, 231, 255, 104, 133, 3, 104
320 DATA 133, 2, 96, 1, 2, 4, 8, 16, 32, 64
330 DATA 128, 1, 2, 4, 8, 16, 32, 64, 128, 7
340 DATA 6, 5, 4, 3, 2, 1, 0
```

Basic Program

```
100 REM C64 HI-RES TO 8023P DUMPER
110 REM DAVID BEREZOWSKI
120 IF PEEK(7936)<>165 THEN PRINT"RUN[SPACE]BASIC[SPACE]
    LOADER[SPACE]PROGRAM[SPACE]FIRST!":END
130 IF C<>0 THEN 390
140 POKE 59468,14:Q$=CHR$(34):PRINT "[CLEAR]C64[SPACE]
    HI-RES[SPACE]TO[SPACE]8023P[SPACE]DUMPER"
150 PRINT"(DAVID[SPACE]BEREZOWSKI[SPACE]-[SPACE]AUG/82)"
```

```
160 PRINT "[DOWN]THIS[SPACE]PROGRAM[SPACE]DUMPS[SPACE]THE
    [SPACE]HI-RES"
170 PRINT "SCREEN[SPACE]OF[SPACE]THE[SPACE]COMMODORE-64
    [SPACE]TO"
180 PRINT "AN[SPACE]8023P[SPACE]PRINTER."
190 PRINT "[DOWN]THIS[SPACE]PROGRAM[SPACE]ASSUMES[SPACE]
    THAT:"
200 PRINT "[DOWN]1)[SPACE]THE[SPACE]HI-RES[SPACE]PAGE
    [SPACE]IS[SPACE]STORED"
210 PRINT "AS[SPACE]A[SPACE]PROGRAM[SPACE]FILE[SPACE]ON
    [SPACE]DISK,"
220 PRINT "WITH[SPACE]A[SPACE]LOAD[SPACE]ADDRESS[SPACE]OF
    [SPACE]$2000."
230 PRINT "[DOWN,SPACE10]OR"
240 PRINT "[DOWN]2)[SPACE]THE[SPACE]HI-RES[SPACE]PAGE
    [SPACE]RESIDES"
250 PRINT "IN[SPACE]MEMORY[SPACE]FROM[SPACE]$2000[SPACE]
    TO[SPACE]$3FFF."
260 PRINT "[DOWN]THIS[SPACE]PROGRAM[SPACE]WILL[SPACE]WORK
    [SPACE]ON[SPACE]ANY
270 PRINT "PET,[SPACE]CBM,[SPACE]VIC,[SPACE]OR[SPACE]C64
    [SPACE]SYSTEM."
280 PRINT "[DOWN,SPACE3,RVS]HIT[SPACE]ANY[SPACE]KEY[SPACE]
    TO[SPACE]CONTINUE"
290 GET A$:IF A$="" THEN 290
300 POKE 59468,14:PRINT "[CLEAR]C64[SPACE]HI-RES[SPACE]TO
    [SPACE]8023P[SPACE]DUMPER"
310 PRINT"(DAVID[SPACE]BEREZOWSKI[SPACE]-[SPACE]AUG/82)"
320 INPUT "[DOWN,RVS]E[RVOFF]XIT[SPACE]OR[SPACE]DUMP
    [SPACE]FROM[SPACE,RVS]D[RVOFF]ISK[SPACE]OR[SPACE,RVS]
    M[RVOFF]EMORY";X$
330 IF X$<>"D" AND X$<>"E" AND X$<>"M" THEN 320
340 IF X$="M" THEN 390
350 IF X$="E" THEN END
360 PRINT "[DOWN]ENTER[SPACE]FILE[SPACE]NAME[SPACE]:"
    :FL$="0:KAREN"
365 PRINT "(IE.[SPACE]"Q$"0:KAREN"Q$")[SPACE]";:INPUT FL$
370 PRINT "[DOWN]LOADING[SPACE]FILE..."
380 C=1:LOAD FL$,8,1
390 OPEN 6,4,6:PRINT#6,CHR$(8):CLOSE 6
400 OPEN 17,4,17:OPEN 18,4,18
410 SYS 7936:REM $1F00
420 OPEN 6,4,6:PRINT#6,CHR$(12):CLOSE 6
430 GOTO 300
```

C

The Incredible Shrinking Data

By Matt Blais

Wherein you will discover a whiz of a technique for cramming

"Hey, that's a really neat design," remarked someone glancing over my shoulder at the screen in front of me. I smiled proudly and admired the latest creation of my new graphics program. I pressed "S" to save the design, but to my dismay, the disk drive gurgled and my program offered the message, "NO DISK SPACE."

I groaned; this was my last disk, and I would have to delete something to make room for my newest masterpiece. A disk directory revealed that my disk was filled by only a handful of programs and ten designs that I had saved.

"Wow, those pictures take up a lot of space," remarked my uninvited friend, who had noticed that each picture took up 37 blocks, or more than 9K bytes of disk space. That added up quickly, amounting to only 18 pictures per 170K diskette, if no programs at all were stored on it.

"There must be something I can do to cram more of these onto a diskette," I mused.

"Hey, why don't you ask The Wiz?" suggested my friend. The Wiz is our local computer expert and machine language fiend.

"Just what I was about to suggest!" I announced. Don't you hate it when they take the words

Illustration—Robert Neumann

more favorite computer screens onto one unsuspecting disk.

right out of your mouth like that?

The Wiz sat in deep thought, wrinkles creasing his 13 year-old brow, sweat beading on his upper lip, as he meditated upon my problem.

"Is he all right, d'you think?" whispered my friend, unaccustomed to the introspective ways of The Wiz.

"Shhh; he's thinking," I answered. "He won't come out until he has an answer."

"But what if there isn't an answer?" demanded my friend; don't you hate people who ask stupid questions?

Just then The Wiz came out of his trance and spoke up in his wise, pre-pubescent voice: "What you need is a good data-compression algorithm," he announced. My friend and I nodded knowingly, not having the faintest idea of what a data-compression algorithm was.

The Wiz rolled his eyes at our wretched attempt of deception. "I can see this will need a little explaining," he piped. Don't you hate it when people jump to conclusions like that?

"A data-compression algorithm is a program that packs a large chunk of data, like a graphics picture, into a smaller chunk," explained The Wiz.

"But that's impossible!" exclaimed my friend rudely. "How can you make 9K of data *less* than 9K of data?" You have to excuse his ignorance in matters like these.

"Yeah, how do you?" I added, confident now that for the first time The Wiz had slipped his gears.

"Well it's not quite that simple. You have to use a special program to encode your data into a smaller chunk—"

"That's the data-compression routine, right?" interrupted my friend.

"Right," I said wisely. "Don't interrupt."

The Wiz continued as if nothing had happened: "—and then when you want your data again, you use another special program to decode it back into its original form."

"I'm still not sure I understand how you can make the data smaller by encoding it," queried my friend. I decided not to answer such an obvious question, but fortunately for my friend, The Wiz has great compassion for the ignorant masses.

"OK, here's an example. Let's say you have a set of data that's 100 numbers long and you want to compress it. It's made up of five one's, followed by five two's, five three's and so on, up to five twenties at the end, like this..." He whipped out a well-chewed pencil and illustrated for us on the back of his physics textbook:

1,1,1,1,1, 2,2,2,2,2, 3,3,3,3,3, 4,4,4,4,4, 5,5,5,5,5,... 19,19,19,19,19, 20,20,20,20,20.

"That's just our data, and it's still 100 numbers long," I pointed out.

"Be patient," advised The Wiz. "We could describe that set of numbers in a different way. See if you can figure out how this next set of numbers really contains the same information as our original set..."

5,1, 5,2, 5,3, 5,4,... 5,18, 5,19, 5,20.

"That's easy," blurted my friend, again taking the words right out of my mouth. There should be a law against that. "The first five means there's five numbers coming, and the second number is what there's five *of*."

"Right: five one's, five two's, five three's, etc., on up to five twenties at the end," explained The Wiz. "That's all you need to recreate our original set, but instead of 100 numbers, we now have only forty. We cut out sixty percent of the space needed to store our set by storing it in a coded form."

"But how do we get back our original set?" I asked, not quite sure how that vanished sixty percent was going to re-materialize when I needed it.

"You have a program that reads in a pair of numbers, like a five and a one, and spits out *five ones,* which is what there were originally. Then it reads the next pair, a five and a two, and spits out *five two's,* then five three's , and so on," explained The Wiz. "The decoding program itself does take up a little space, but not nearly as much as you save overall. It's the old trade-off between memory space and computer time. This way the computer has to spend some time encoding and decoding the data, but it's usually worth it for the space you save."

Sometimes The Wiz forgets his age and talks about things he shouldn't comprehend, but we usually humor him. He can be useful at times, too.

The light of understanding was beginning to dawn in my friend's eyes. "So anywhere you have a string of the same number, like eight three's, you can use just two numbers to represent all eight of them?" he hazarded.

"Right. Using your example of eight three's, those two numbers would be an eight and a three. This kind of compression is especially good for computer graphics pictures, which tend to have long strings of the same number, like zero, which can be replaced by just two numbers in the coded version of the picture." The Wiz was really getting worked up now. Past experience with The Wiz had taught me that right about now he was probably getting ready to throw a program at us. I resisted a strong urge to duck.

"Somewhere in here there's a set of programs that'll do the trick nicely," announced The Wiz, opening a massive notebook and wading through papers, listings, newspaper clippings, diskettes and baseball cards. "Aha," he finally cried, and produced a crumpled, faded listing covered with pencilled remarks and scribblings. "They're really pretty simple programs—" he began.

"But they're in *machine* language," I whined.

"*Assembly* language," he corrected, "but you can run them right from within your BASIC program. Here's how the first one, the encoder—"

"You mean the *data-compression algorithm*," interjected my friend, trying to show off.

"—how the *data-compression algorithm* works," resumed The Wiz. "It reads in the first number of your data, and stores it in a variable called OLDBYTE. Then it compares that to the next number, and the next, and so on, until it runs into a number that's *different* from the original one stored in OLDBYTE."

"And what happens when it finds a different number?" I asked, my head beginning to spin.

"I was just getting to that," replied The Wiz patiently. "When the program finds a number different from the one in OLDBYTE, it writes out how many of OLDBYTE there were, and the value of OLDBYTE itself, to the coded version of the data; that's our pair of numbers there. In the example I used before with one hundred numbers, the first pair would be a five and a one, since the first five numbers were ones."

"I see," exclaimed my friend, eager to impress The Wiz, "and then it starts counting how many two's there are, and when it runs out of two's, it writes out a five and a two—"

"—because there were *five*

two's," finished The Wiz. "Then it starts counting three's, which is the next number, and so on until it's done the entire set of numbers."

I was overcome by a strong sense of *deja vu* just about then, but it was finally all beginning to make sense. "And then when it's all done, the decoder program will read in each pair and write back out what was there in the first place: for your application, a graphics screen," concluded The Wiz.

My friend was looking thoughtful—a bad sign. "What happens if there aren't any long strings of the same number in your picture: what if *every* number is different from the last one?" he wondered out loud. Leave it to him to cloud the issue just when things are beginning to make sense.

"Then this kind of compression algorithm won't make your picture any smaller; in fact, it will make it *bigger*," replied The Wiz.

"*Bigger?*" my friend and I gasped in unison. "How can a data-*compression* program make something bigger?" I added, now totally confused.

"In your data, every string of the same number uses up just two numbers in the compressed version. If you have only two of the same number, then you don't save any space at all because the compressed version needs two numbers to represent them," The Wiz explained.

"And if there's only one number, and the next number is different, you still need two numbers to represent that one number, which means your so-called 'compressed' version is actually going to be bigger than your original picture..." Luckily, my friend ran out of breath at that point.

"Exactly. This kind of compression algorithm only saves space if you have a lot of strings of the same number, like I mentioned earlier. Graphics pictures usually do, so you should be safe," continued The Wiz. "These two programs scan the picture from top to bottom, working their way from the left edge of the screen over to the right edge. When you run the *de*compression program to get your picture back, you can see it sweep across the screen in the same pattern as it was read off. Try these out and see how much space you save."

Well, The Wiz was right, as usual, and now my pictures only take up about half as much disk space as they used to. They load twice as fast, too.

My friend really got interested in this compression stuff, and he tells me that he's experimenting with more complicated compression algorithms that scan the screen horizontally instead of vertically, and count half-numbers (he calls them "nybbles") instead of numbers. He says he even tried counting *bits,* whatever they are, but the program The Wiz gave us still compresses most screens the best. My friend's becoming a regular machine language fiend, at that; overexposure to The Wiz has been known to do that to people...

Data Compression Assembly Code From The Wiz

Before assembling this program, the constant SCREEN should be set to the beginning of the area you want to compress. Optionally, you can POKE the screen location directly into the INITGET routine before running the program.

BUFFER should be the address of the beginning of an unused area that is at least eight thousand bytes in length (this area is only used temporarily). This is where the compressed data is left after the compression program has been run.

If you are compressing a picture that was created using Commodore's *Super Expander,* the picture occupies memory from hex $A000 to $BF40. This is the RAM underneath BASIC ROM, so before running the compression program, you should make sure that the RAM is visible in that area by appending the following short program to the beginning of the compression program:

```
SEI
LDA $01
AND #$FE
STA $01
JMP $9000
;(run the compression
program)
```

Before exiting back to BASIC, the following should be executed to insure that BASIC ROM is once again safely in place:

```
ORA #$01
STA $01
CLI
RTS
```

After the compression program has been run, the last address of the last byte of the buffer that was used will be in zero-page memory addresses $A3—$A4, decimal 163—164 ("BUFFER" is the address of the first byte). These two addresses are the beginning and end of the compressed data you will want to save. A save can be accomplished with any good machine language monitor or by using the KERNAL save routine (see the *Commodore 64 Programmer's Reference Guide,* pages 293-294 for more details) or by simply setting the BASIC start-of-program pointer (locations 43-44) to the beginning of the data and the start-of-variables pointer (locations 45-46) to the end of the area to be saved. Then issue a normal SAVE command just as if you were saving a program. Be sure to restore these pointers to their original values when you are finished.

To load the compressed files, use BASIC's LOAD "filename", 8,1, or a load command from a machine-language monitor. The compressed data will load back into the same buffer area from which it was saved. To decompress the data back to a normal graphics picture, run the decompression program (SYS to hex $9004, decimal 36868). The picture will be written out to wherever SCREEN points. You do not have to run the short program above before a decompression, because writing data to a ROM address will write the data to the RAM of that same address.

P.S.: Don't forget to save color memory (memory from hex $D800 to $DBE8)! It is hardly long enough to bother with compressing.

(Shrinking Data: Assembly Code)

```
9000                   .OPT P2
               ;-------------------------;
               ;  COMPRESS BITMAP SCREEN ;  (VERTICALLY BY BYTE)
               ;    V2.0     10/31/83    ;
               ;-------------------------;
               ;
9000          SCREEN   =    $2000
9000          INPTR    =    $A3
9000          OUTBFR   =    $6000
9000          OUTPTR   =    $A5
               ;
9000                   *=   $9000
               ;=============================
               ;
9000 20 08 90          JSR  COMPRS
9003 60                RTS
9004 20 C0 90          JSR  EXPND
9007 60                RTS
               ;
               ;---------------------------
9008          COMPRS   =    *
               ;
9008 20 E9 90          JSR  INIT        ;INITIALIZE VARIABLES.
900B A2 00             LDX  #0          ;X IS ALWAYS 0 IN THIS PROGRAM.
900D B1 A3             LDA  (INPTR),Y   ;GET FIRST BYTE TO SET UP THE
900F 8D 0B 91          STA  NXTBYT      ;VARIABLES FOR THE COMPRESSION.
9012 20 6D 90          JSR  ADVIN       ;ADVANCE SCREEN TO NEXT BYTE.
               ;
9015          NWPAIR   =    *           ;START ON A NEW NUMBER.
9015 8E 09 91          STX  NMBYTS      ;STARTING WITH ONLY ONE.
9018 AD 0B 91          LDA  NXTBYT      ;THE LAST NUMBER READ BECOMES
901B 8D 0A 91          STA  CURBYT      ;THE CURRENT NUMBER HERE.
             ;
901E          ANOTHR   =    *           ;GET THE NEXT NUMBER FROM THE
901E EE 09 91          INC  NMBYTS      ;SCREEN; INCREMENT HOW MANY
               ;                        ;WE'VE READ.
9021 B1 A3             LDA  (INPTR),Y   ;GET THE NEXT NUMBER FROM THE
9023 8D 0B 91          STA  NXTBYT      ;SCREEN AND SAVE IT.
9026 20 6D 90          JSR  ADVIN       ;ADVANCE THE SCREEN POINTER.
9029 08                PHP              ;SAVE CARRY FLAG (DONE YET?)
               ;
902A AD 0B 91          LDA  NXTBYT      ;IF THE NEW NUMBER DOESN'T
902D CD 0A 91          CMP  CURBYT      ;EQUAL OUR CURRENT ONE,
9030 D0 0C             BNE  SAVEPR      ;SAVE THE CURRENT ONE, & HOW MANY.
9032 AD 09 91          LDA  NMBYTS      ;(OTHERWISE, IT'S ANOTHER ONE).
9035 F0 07             BEQ  SAVEPR      ;MUST SAVE THE PAIR IF 256 OF 'EM.
9037 28                PLP              ;RECOVER THE CARRY- ARE WE DONE?
9038 90 E4             BCC  ANOTHR      ;IF NOT, GO GET ANOTHER BYTE.
903A 08                PHP              ;RE-SAVE THE CARRY.
903B EE 09 91          INC  NMBYTS      ;INCLUDE LAST BYTE (THE SAME)
               ;
903E          SAVEPR   =    *           ;SAVE NUMBER, HOW MANY THERE WERE.
903E AD 09 91          LDA  NMBYTS      ;GET HOW MANY THERE WERE,
9041 81 A5             STA  (OUTPTR,X)  ;SAVE IT.
9043 20 66 90          JSR  ADVOUT      ;ADVANCE PTR IN OUR OUTPUT BUFFER.
9046 AD 0A 91          LDA  CURBYT      ;GET THE NUMBER ITSELF,
9049 81 A5             STA  (OUTPTR,X)  ;SAVE IT.
904B 20 66 90          JSR  ADVOUT      ;ADVANCE POINTER.
904E 28                PLP              ;RECOVER CARRY- ARE WE DONE YET?
904F 90 C4             BCC  NWPAIR      ;IF NOT, START ON THE NEW NUMBER.
               ;                        ;WE ARE DONE- FINISH UP.
9051 08                PHP              ;RE-SAVE THE CARRY AGAIN.
9052 AD 0B 91          LDA  NXTBYT      ;IF THE LAST BYTE WAS THE SAME AS
9055 CD 0A 91          CMP  CURBYT      ;OUR CURRENT NUMBER,
9058 F0 0A             BEQ  CODONE      ;THEN IT'S ALREADY TAKEN CARE OF.
905A 8D 0A 91          STA  CURBYT      ;IF NOT, IT'S THE LAST 'PAIR'--
905D A9 01             LDA  #1          ;ONLY ONE OF IT.
905F 8D 09 91          STA  NMBYTS      ;
9062 D0 DA             BNE  SAVEPR      ;GO BACK TO SAVE THE LAST PAIR.
```

```
                 ;WE'RE DONE--
9064 28          CODONE   PLP               ;GET OUR CARRY OFF THE STACK
9065 60                   RTS               ;AND GO ON HOME.
                 ;-----------------------------------------
9066             ADVOUT   =      *          ;ADVANCE POINTER TO OUTPUT BUFFER.
9066 E6 A5                INC    OUTPTR     ;INCREMENT LO BYTE.
9068 D0 02                BNE    ADB00      ;IF NOT ZERO, THAT'S ALL.
906A E6 A6                INC    OUTPTR+1   ;IF WAS ZERO, INCREMENT HI BYTE.
906C 60          ADB00    RTS               ;THAT'S IT.
                 ;
                 ;-----------------------------------------
906D             ADVIN    =      *          ;ADVANCE SCREEN POINTER TO NEXT
                                            ;BYTE; Y INDEXES INTO THE CURRENT
906D C8                   INY               ;CHARACTER BLOCK.
906E C0 08                CPY    #8         ;IF NOT DONE WITH THIS BLOCK,
9070 90 4C                BCC    ADVINEX    ;THAT'S ALL IT TAKES.
                 ;
9072 A0 00                LDY    #0         ;DONE BLOCK; START ON NEXT BLOCK.
9074 18                   CLC               ;INCREMENT ROW ADDRESS BY 320.
9075 AD 07 91             LDA    ROWADR     ;(THE NEXT LINE).
9078 69 40                ADC    #$40
907A 8D 07 91             STA    ROWADR     ;(320 = HEX $140).
907D AD 08 91             LDA    ROWADR+1
9080 69 01                ADC    #1
9082 8D 08 91             STA    ROWADR+1   ;IF ROW OFFSET = $1FXX THEN WE'RE PAST
9085 C9 1F                CMP    #$1F       ;THE END OF THIS ROW:
9087 D0 20                BNE    NWBASE     ;OTHERWISE, CALCULATE THE ADDRESS
                                            ;OF THE NEW CHARACTER CELL.
9089 8C 07 91             STY    ROWADR     ;DONE THIS COLUMN, START BACK AT TOP:
908C 8C 08 91             STY    ROWADR+1   ;(ROW ZERO- Y IS 0).
908F 18                   CLC
909Q AD 05 91             LDA    COLADR     ;NEXT COLUMN- INCREMENT COLUMN OFFSET
9093 69 08                ADC    #8         ;BY EIGHT FOR NEXT CHAR. BLOCK.
9095 8D 05 91             STA    COLADR
9098 90 03                BCC    ADVI01
909A EE 06 91             INC    COLADR+1   ;INCREMENT HI BYTE.
909D             ADVI01   =      *
909D AD 06 91             LDA    COLADR+1   ;CHECK IF WE ARE PAST THE LAST COLUMN
90A0 F0 07                BEQ    NWBASE     ;(OFFSET OF $140):
90A2 AD 05 91             LDA    COLADR
90A5 C9 40                CMP    #$40       ;IF SO, EXIT WITH CARRY SET TO SIGNAL
90A7 B0 16                BCS    DONEXT     ;THAT WE'RE DONE.
                 ;
90A9             NWBASE   =      *          ;CALCULATE ADDRESS OF NEW CHAR BLOCK.
90A9 18                   CLC
90AA A9 00                LDA    #<SCREEN   ;ADDRESS = START OF SCREEN
90AC 6D 05 91             ADC    COLADR     ;PLUS ROW OFFSET
90AF 6D 07 91             ADC    ROWADR     ;PLUS COLUMN OFFSET.
90B2 85 A3                STA    INPTR
90B4 A9 20                LDA    #>SCREEN   ;AND ALL THAT FOR THE HI BYTE, TOO.
90B6 6D 06 91             ADC    COLADR+1
90B9 6D 08 91             ADC    ROWADR+1
90BC 85 A4                STA    INPTR+1
                 ;
90BE             ADVINEX  =      *
90BE 18                   CLC               ;CLEAR THE CARRY FOR A NORMAL,
90BF 60          DONEXT   RTS               ;'NOT-FINISHED' EXIT.

                 ;-----------------------------------------
                 ;   THIS PART OF THE PROGRAM DE-COMPRESSES
                 ;     THE COMPRESSED DATA STORED IN THE BUFFER
                 ;     BACK TO THE AREA OF MEMORY STARTING AT SCREEN.

90C0 20 E9 90 EXPND       JSR    INIT       ;INITIALIZE THE VARIABLES.
90C3 8C 0D 91 NXPND       STY    YSAVE      ;SAVE THE POINTER INTO THE CHAR BLOCK.
90C6 A0 00                LDY    #0         ;ZERO Y TO USE FOR INDIRECT ADDRESSINC.
90C8 B1 A5                LDA    (OUTPTR),Y ;GET THE NUMBER OF BYTES FROM THE
90CA AA                   TAX               ;BUFFER AND STORE IN X.
```

```
90CB 20 66 90          JSR  ADVOUT     ;ADVANCE TO THE NEXT BYTE, WHICH IS WHAT
90CE B1 A5             LDA  (OUTPTR),Y ;THERE ARE GOING TO BE X  OF.
90D0 8D 0C 91          STA  ASAVE      ;SAVE THAT FOR LATER.
90D3 20 66 90          JSR  ADVOUT     ;ADVANCE TO NEXT BYTE (FOR NEXT TIME THRU).
90D6 AC 0D 91          LDY  YSAVE      ;RETRIEVE OUR POINTER INTO CHAR. BLOCK.
              ;
90D9 AD 0C 91 EXGN     LDA  ASAVE      ;THIS IS THE BYTE WE WRITE TO THE SCREEN.
90DC 91 A3             STA  (INPTR),Y  ;SO, WRITE IT.
90DE 20 6D 90          JSR  ADVIN      ;ADVANCE TO THE NEXT SCREEN POSITION.
90E1 B0 05             BCS  ENDEXP     ;IF WE'RE DONE, THEN EXIT.
90E3 CA                DEX             ;OTHERWISE, DECREMENT OUR COUNTER OF HOW
90E4 D0 F3             BNE  EXGN       ;MANY BYTES HAVE BEEN WRITTEN; IF WE'RE NOT
              ;                        ;THRU WITH THIS ONE YET, WRITE IT OUT AGAIN.
90E6 90 DB             BCC  NXPND      ;OTHERWISE, IF WE'RE NOT DONE, GET THE NEXT
90E8 60       ENDEXP   RTS             ;PAIR FROM THE BUFFER.
                                       ;(OR IF WE'RE DONE, RETURN).
              ;
---------------------------------
90E9          INIT     =    *          ;INITIALIZE THE VARIABLES.
              ;
90E9 A9 00             LDA  #<SCREEN   ;START THE POINTER OFF AT THE BEGINNING
90EB 85 A3             STA  INPTR      ;OF THE SCREEN.
90ED A9 20             LDA  #>SCREEN
90EF 85 A4             STA  INPTR+1
90F1 A9 00             ?LDA  #<OUTBFR  ;AND THE BUFFER POINTER AT THE START
90F3 85 A5             STA  OUTPTR     ;OF THE BUFFER.
90F5 A9 60             LDA  #>OUTBFR
90F7 85 A6             STA  OUTPTR+1
              ;
9109 A9 00             LDA  #0         ;ZERO ALL THE VARIABLES.
900B A0 07             LDY  #7         ;(START AT ROW 0 AND COLUMN 0).
900D 99 05 91 BIN00    STA  COLADR,Y
9100 88                DEY
9101 10 FA             BPL  BIN00
9103 A8                TAY             ;LEAVE WITH Y = 0 (THE FIRST BYTE
9104 60                RTS             ;IN THE FIRST CHAR BLOCK).
              ;
              ;======================================
              ;          V A R I A B L E S
              ;
9105          COLADR   *=   *+2
9107          ROWADR   *=   *+2
9109          NMBYTS   *=   *+1
910A          CURBYT   *=   *+1
910B          NXTBYT   *=   *+1
910C          ASAVE    *=   *+1
910D          YSAVE    *=   *+1
              ;======================================
```

C

Commodore Indispensable Software... For When You Really Mean Business

Computer systems don't have to be expensive to be effective—even in business. Commodore has been proving that for years.

Now your Commodore 64 combined with a full-range of application software means business when you do.

Whether you're trying to run a small business or manage your home finances, the Commodore accounting series software makes it affordable and easy.

Included in the series are the following software packages:

The **GENERAL LEDGER** program brings you into the computer era with an economical system for maintaining and monitoring profit and loss records, balance sheets and financial reports.

Monitoring payables and receivables is always a problem for any small business. Commodore's **ACCOUNTS PAYABLE/CHECKWRITING** and **ACCOUNTS RECEIVABLE/BILLING** computerize your invoicing, customer sales and credit information. Handling up to 50 vendors or 75 customers, and maximum monthly totals of $99,999,999.99, these two programs are truly indispensable.

The series also includes a comprehensive **INVENTORY CONTROL** program. With over 1000 items per disk, it's perfect for retail stores, mail order businesses, libraries, bookstores, warehouses—virtually anywhere large numbers of items need to be tracked, priced, billed, monitored and recorded.

When it comes to paying your employees, Commodore also solves that problem. Our **PAYROLL** package allows for up to 50 people and automatically computes tax calculations.

For further details on these packages and information on the complete line of application software, contact your local Commodore dealer.

The complete Commodore 64 Accounting series software—**Sophisticated business packages at an Unsophisticated price!!**

commodore
COMPUTERS
First In Quality Software

TEXT ANALYSIS *(Continued from page 33)*

on similar machines should present no problem. As given it will work with BASIC 2.0 and 3.0, using less than 8K. Notice that for ease of typing, the cursor control characters are translated into words. You should of course type the appropriate symbol. For instance, in line 150, type shift-home and two cursor-downs.

Bomb-proof input is handled in two ways. For simple Y/N responses to questions a trick INPUT statement is used (lines 150, 160, 540 and 550). For the actual typing of text, the GET statement works better (the subroutine at lines 730-760). Using the GET technique means that all punctuation can be used. In fact, only the question mark, exclamation mark and period will register (line 230), and these mark the end of a sentence. All other punctuation is ignored. A disadvantage of using this technique is that the cursor control keys are non-functional, and that includes the delete key. You will have to live with any mistakes in the typing of the text, but with a sufficiently large sample this should not alter the analysis significantly.

The analysis is carried out while the typing is taking place. This means that there is no lengthy wait at the end of the program. It is possible for a fast typist to race against the speed of the PET, and it could happen that a few characters are lost in this way. This ought not to affect the outcome, but if it happens, slow down to a more leisurely pace.

And now the acid test! The following run shows the results obtained from two verses of "Jabberwocky" by Lewis Carroll. In actual practice the text typed should contain more than 100 words and more reliable results are obtained by taking selections from throughout the text. Also note that dialogue is usually far simpler than other text. Selections from the text should take this into account.

Here is a short breakdown of the program:

210	Initialization
220	Store last character, input one more
230	Trap end of sentence
240	Trap end of word
250	Trap end of text
260	Trap non-letters
280	Count vowels
300	Except double vowels
320	Count long words
330	Do not count a final "e"
340	How many vowels in each word?
350-380	Fog Index
390-530	Fry Index
600-720	Instructions
730-840	Opening titles
750	Turn quote mode off

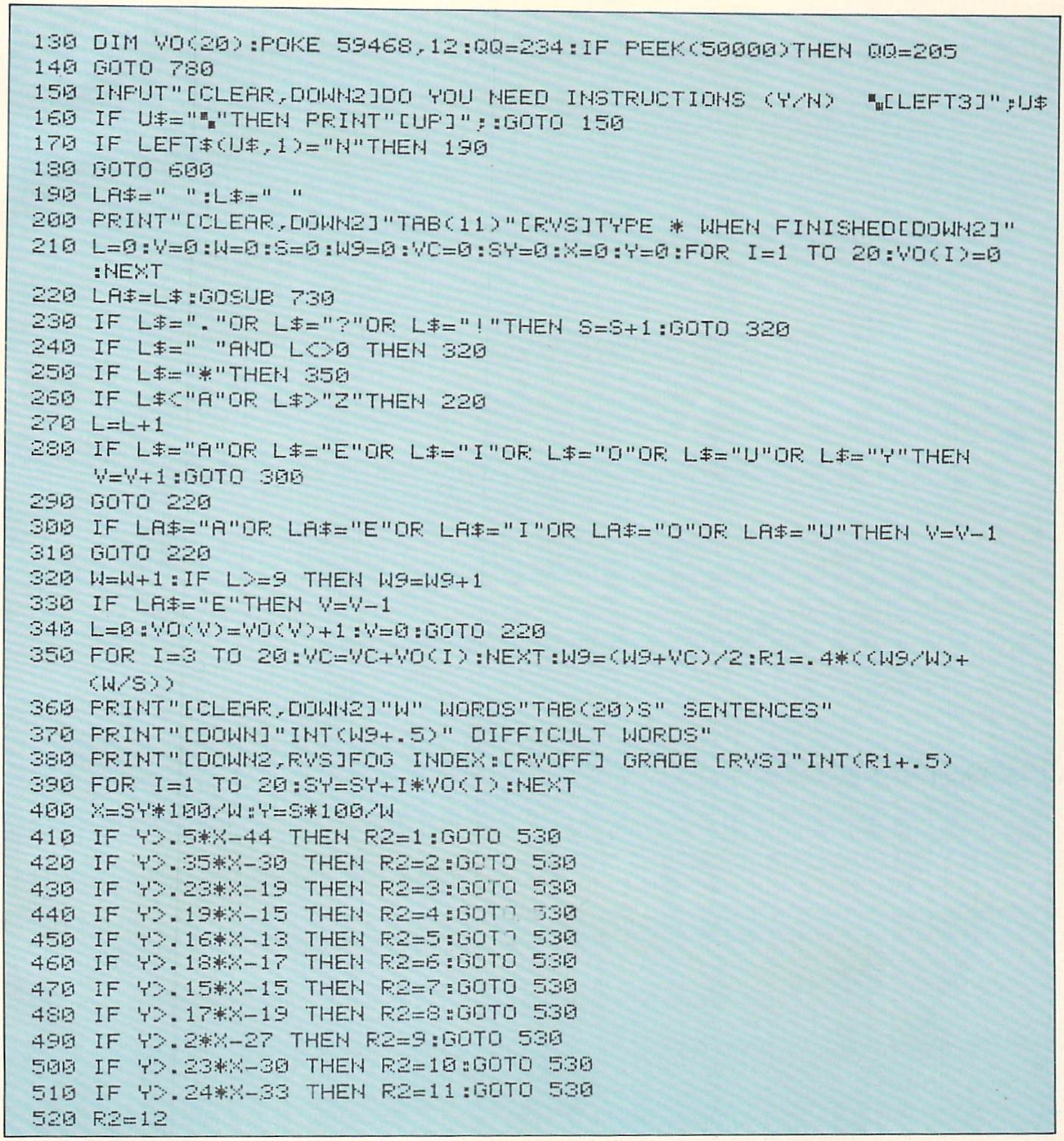
And the more important variables used:

L	Length of word
L$	Current character
LA$	Last character
U$	Y/N response
V	Number of vowels in word
W	Number of words in text
S	Number of sentences in text
W9	Number of words more than 8 letters
VC,VO	Vowel counters
SY	Number of syllables in text
X,Y	Fry Index variables
R1	Fog Index grade
R2	Fry Index grade
A1	Fog Index average
A2	Fry Index average
QQ	Quote mode flag

C

Text Analysis

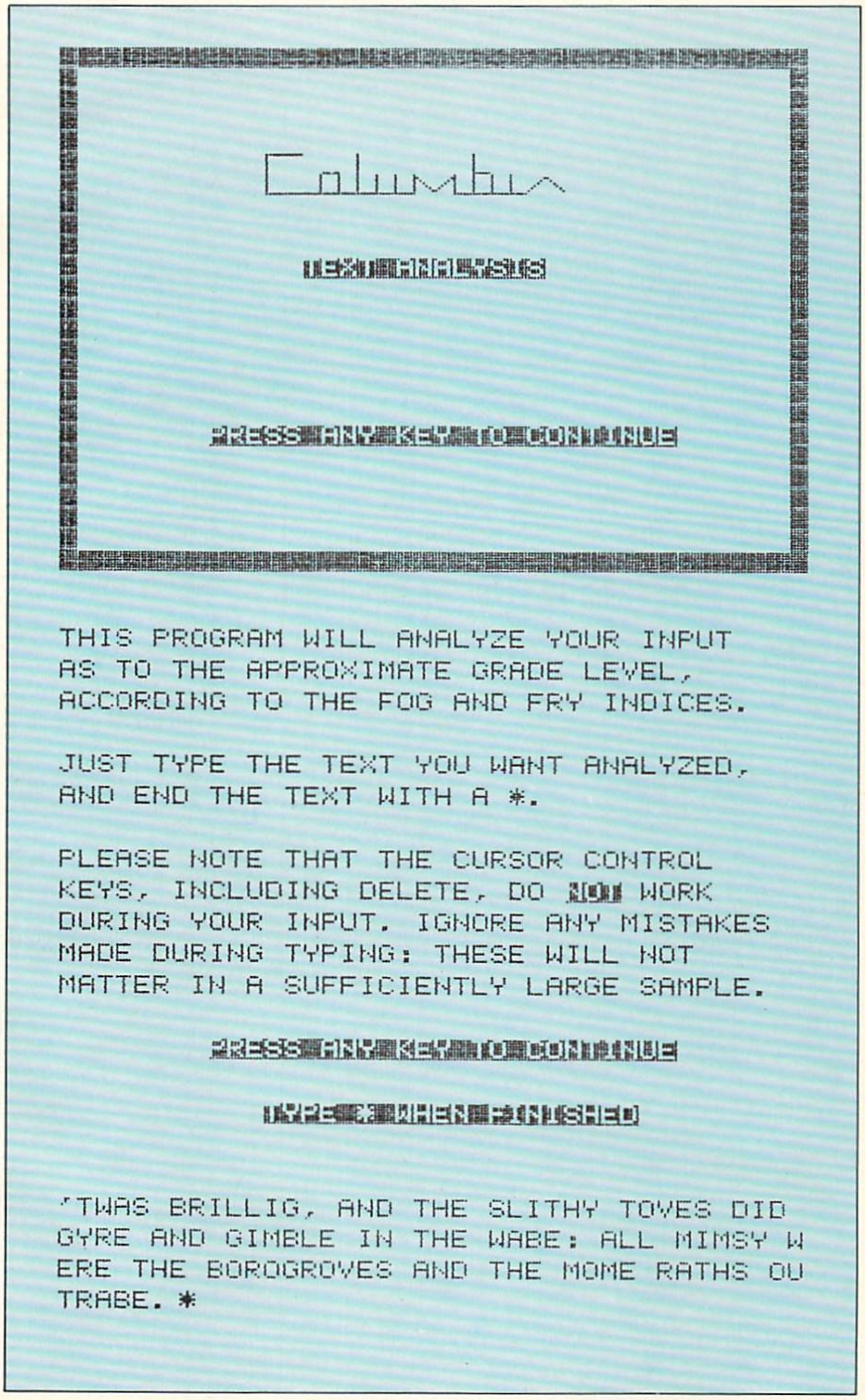
```
100 REM TEXTANALYSIS 2.0
110 REM BY  ANDY GAMBLE JAN 82
120 REM COLUMBIA COLLEGE, 1619 W 10TH      AVE, VANCOUVER BC V6J 2A2
```



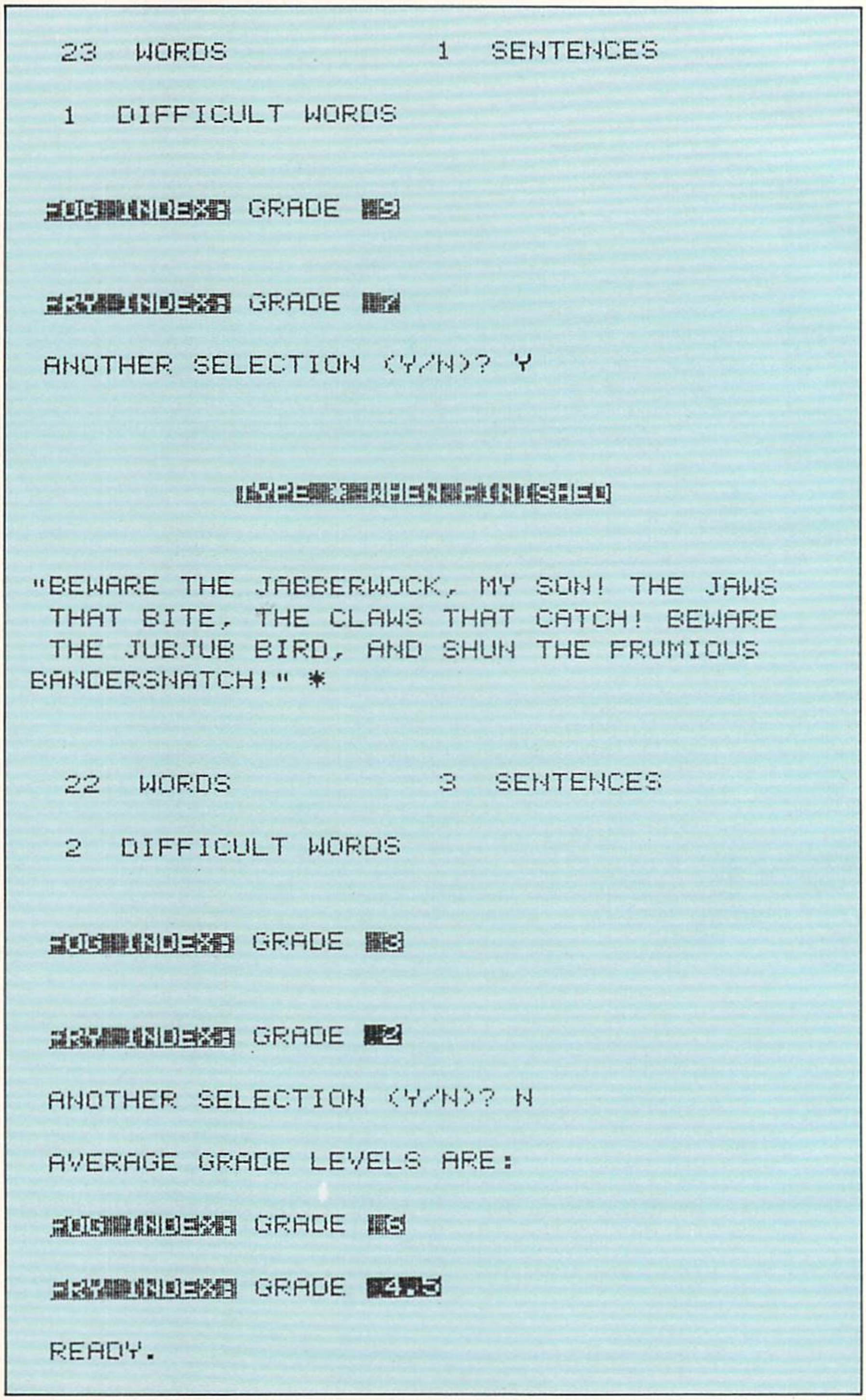
```
130 DIM VO(20):POKE 59468,12:QQ=234:IF PEEK(50000)THEN QQ=205
140 GOTO 780
150 INPUT"[CLEAR,DOWN2]DO YOU NEED INSTRUCTIONS (Y/N)  ▚[LEFT3]";U$
160 IF U$="▚"THEN PRINT"[UP]";:GOTO 150
170 IF LEFT$(U$,1)="N"THEN 190
180 GOTO 600
190 LA$=" ":L$=" "
200 PRINT"[CLEAR,DOWN2]"TAB(11)"[RVS]TYPE * WHEN FINISHED[DOWN2]"
210 L=0:V=0:W=0:S=0:W9=0:VC=0:SY=0:X=0:Y=0:FOR I=1 TO 20:VO(I)=0
    :NEXT
220 LA$=L$:GOSUB 730
230 IF L$="."OR L$="?"OR L$="!"THEN S=S+1:GOTO 320
240 IF L$=" "AND L<>0 THEN 320
250 IF L$="*"THEN 350
260 IF L$<"A"OR L$>"Z"THEN 220
270 L=L+1
280 IF L$="A"OR L$="E"OR L$="I"OR L$="O"OR L$="U"OR L$="Y"THEN
    V=V+1:GOTO 300
290 GOTO 220
300 IF LA$="A"OR LA$="E"OR LA$="I"OR LA$="O"OR LA$="U"THEN V=V-1
310 GOTO 220
320 W=W+1:IF L>=9 THEN W9=W9+1
330 IF LA$="E"THEN V=V-1
340 L=0:VO(V)=VO(V)+1:V=0:GOTO 220
350 FOR I=3 TO 20:VC=VC+VO(I):NEXT:W9=(W9+VC)/2:R1=.4*((W9/W)+
    (W/S))
360 PRINT"[CLEAR,DOWN2]"W" WORDS"TAB(20)S" SENTENCES"
370 PRINT"[DOWN]"INT(W9+.5)" DIFFICULT WORDS"
380 PRINT"[DOWN2,RVS]FOG INDEX:[RVOFF] GRADE [RVS]"INT(R1+.5)
390 FOR I=1 TO 20:SY=SY+I*VO(I):NEXT
400 X=SY*100/W:Y=S*100/W
410 IF Y>.5*X-44 THEN R2=1:GOTO 530
420 IF Y>.35*X-30 THEN R2=2:GOTO 530
430 IF Y>.23*X-19 THEN R2=3:GOTO 530
440 IF Y>.19*X-15 THEN R2=4:GOTO 530
450 IF Y>.16*X-13 THEN R2=5:GOTO 530
460 IF Y>.18*X-17 THEN R2=6:GOTO 530
470 IF Y>.15*X-15 THEN R2=7:GOTO 530
480 IF Y>.17*X-19 THEN R2=8:GOTO 530
490 IF Y>.2*X-27 THEN R2=9:GOTO 530
500 IF Y>.23*X-30 THEN R2=10:GOTO 530
510 IF Y>.24*X-33 THEN R2=11:GOTO 530
520 R2=12
```

```
530 PRINT"[DOWN2,RVS]FRY INDEX:[RVOFF] GRADE [RVS]"R2
540 INPUT"[DOWN]ANOTHER SELECTION (Y/N)  ▚[LEFT3]";U$
550 IF U$="▚"THEN PRINT"[UP2]";:GOTO 540
560 N=N+1:A1=A1+R1:A2=A2+R2
570 IF LEFT$(U$,1)="Y"THEN 200
580 PRINT"[DOWN]AVERAGE GRADE LEVELS ARE:"
582 PRINT"[DOWN,RVS]FOG INDEX:[RVOFF] GRADE [RVS]"INT(A1/N+.5)
584 PRINT"[DOWN,RVS]FRY INDEX:[RVOFF] GRADE [RVS]"A2/N
590 END
600 PRINT"[CLEAR,DOWN2]"
610 PRINT"THIS PROGRAM WILL ANALYZE YOUR INPUT"
620 PRINT"AS TO THE APPROXIMATE GRADE LEVEL,"
630 PRINT"ACCORDING TO THE FOG AND FRY INDICES."
640 PRINT"[DOWN]JUST TYPE THE TEXT YOU WANT ANALYZED,"
650 PRINT"AND END THE TEXT WITH A *."
660 PRINT"[DOWN]PLEASE NOTE THAT THE CURSOR CONTROL"
670 PRINT"KEYS, INCLUDING DELETE, DO [RVS]NOT[RVOFF] WORK"
680 PRINT"DURING YOUR INPUT. IGNORE ANY MISTAKES"
690 PRINT"MADE DURING TYPING: THESE WILL NOT"
700 PRINT"MATTER IN A SUFFICIENTLY LARGE SAMPLE.
710 GOSUB 850
720 GOTO 190
730 PRINT"_[LEFT]";
740 GET L$:IF L$=""THEN 740
750 PRINT L$;:POKE QQ,0
760 RETURN
770 END
780 PRINT"[CLEAR]":FOR I=32768 TO 32807:POKE I,224:POKE I+960,224
    :NEXT
790 FOR I=32808 TO 33688 STEP 40:POKE I,224:POKE I+39,224:NEXT
800 PRINT"[HOME,DOWN4]"
810 PRINT TAB(9)"  ┌‾ _|     L
820 PRINT TAB(9)"  └__|_|_|_|_|_|\/|_|_|_|_/\
830 PRINT"[HOME,DOWN10]"TAB(13)"[RVS]TEXT ANALYSIS"
840 PRINT"[DOWN5]":GOSUB 850:GOTO 150
850 PRINT"[DOWN]"TAB(8)"[RVS]PRESS ANY KEY TO CONTINUE"
860 GET ZZ$:IF ZZ$=""THEN 860
870 RETURN
```

Text Analysis: Sample Output



```
TEXT ANALYSIS

PRESS ANY KEY TO CONTINUE

THIS PROGRAM WILL ANALYZE YOUR INPUT
AS TO THE APPROXIMATE GRADE LEVEL,
ACCORDING TO THE FOG AND FRY INDICES.

JUST TYPE THE TEXT YOU WANT ANALYZED,
AND END THE TEXT WITH A *.

PLEASE NOTE THAT THE CURSOR CONTROL
KEYS, INCLUDING DELETE, DO NOT WORK
DURING YOUR INPUT. IGNORE ANY MISTAKES
MADE DURING TYPING: THESE WILL NOT
MATTER IN A SUFFICIENTLY LARGE SAMPLE.

PRESS ANY KEY TO CONTINUE

TYPE * WHEN FINISHED

'TWAS BRILLIG, AND THE SLITHY TOVES DID
GYRE AND GIMBLE IN THE WABE: ALL MIMSY W
ERE THE BOROGROVES AND THE MOME RATHS OU
TRABE. *
```



```
 23  WORDS                  1  SENTENCES

 1  DIFFICULT WORDS

 FOG INDEX: GRADE  2

SMOG INDEX: GRADE  4

ANOTHER SELECTION (Y/N)? Y


         (ENTER * WHEN FINISHED)


"BEWARE THE JABBERWOCK, MY SON! THE JAWS
 THAT BITE, THE CLAWS THAT CATCH! BEWARE
 THE JUBJUB BIRD, AND SHUN THE FRUMIOUS
BANDERSNATCH!" *


 22  WORDS                  3  SENTENCES

 2  DIFFICULT WORDS

 FOG INDEX: GRADE  3

SMOG INDEX: GRADE  4

ANOTHER SELECTION (Y/N)? N

AVERAGE GRADE LEVELS ARE:

 FOG INDEX: GRADE  3

SMOG INDEX: GRADE  4.5

READY.
```

Pennsylvania Funds School Computer Support

Last September Governor Dick Thornburgh of Pennsylvania announced the award of a joint $125,000 contract to the Educational Products Information Exchange—(EPIE) and Consumers Union (CU) to assist teachers and school administrators throughout the state in the selection and use of microcomputers in Commonwealth classrooms during the 1983-84 school year.

"Computers are becoming a vital classroom tool, but they must be used wisely and not willy-nilly," Thornburgh said during contract signing ceremonies at the Harrisburg (PA) School District's Middle School. "The contract we are signing today will help our school districts utilize technology dollars more effectively in choosing proper and useful equipment and software.

"Schools are now spending millions of dollars on software and computer products and, according to teachers and administrators, one of the greatest needs today is reliable information which will help in the selection of the right equipment and software for the jobs they want to do," he said.

The contract will make available to every Pennsylvania school district and intermediate unit, the free services of the Information Exchange, a national clearinghouse for the evaluation of instructional materials, including textbooks and various brands of microcomputers and software being sold for instructional use. Beginning in January 1984, the districts also will have access to an electronic data base which will summarize evaluations done by other organizations or published by national journals.

"Pennsylvania already has become a recognized leader in the promotion, support and funding of efforts to incorporate the use of advanced technology in the classroom. With this contract we become only the third state in the nation to provide a comprehensive evaluation service to schools participating in this effort," Thornburgh said.

Last fall, a state Education Department survey showed that about 7,500 microcomputers were being used in Pennsylvania school districts, intermediate units and vocational-technical schools. That number is expected to increase by more than 50 percent during the current school year.

Thornburgh praised EPIE, a non-profit service headed by P. Kenneth Komoski, for its "long history of providing objective and reliable assessments of materials being commercially sold for use in the nation's schools."

"Consumers Union," he said, "has an equally impeccable reputation for its evaluation of consumer products. The partnership of these two organizations to test the effectiveness of computer materials being produced for school will be of great advantage to us as we continue to spend substantial dollars for advanced technology in the classrooms."

Under Thornburgh's direction, efforts toward the development and implementation of computer education for Pennsylvania students, have included:

- Distribution of nearly $1.3 million for the development of projects involving the use of new technology in classrooms.
- Awarding of $800,000 in mini-grants for projects using new technologies to enhance the instruction of students in special education programs.
- Development of a computer literacy course by the state Education Department, which has been published by the Continental Press, Inc., of Elizabethtown, for national distribution. (Royalties amounting to more than $11,000 were paid to the department this summer by the publisher and will be used to support other technology-related activities.)
- Administration support for continued allocation of federal funds by local school districts for incorporating computer technology into vocational education, with over $9 million used so far for the purchase of computer hardware and software for vocational programs.
- Administration support for allocation by local school districts of at least $6.4 million in federal block grant funds for the purchase of computer equipment and programs for education and administrative purposes.
- Administration support for the State Board of Education's proposal to require every school to offer computer science courses to its students. C

What to Do With Your Computer Now that the Holidays Are Over

by Gail Austin

Type Right and Chopper Math Fill the Need

The holiday season is long passed, and the new year has begun. The children are happy with their new Commodore 64 or VIC 20 computer and always want to play their computer games. About now the parents want to see their children use the computer to do more than that. So off the entire family goes to buy more software.

Once inside the computer store with so many software packages to choose from, the parents feel like kids in a candy store—too many choices and all the wrappers look great, but there is not enough money so the choice must be the right one. With this in mind, I would like to recommend two software packages that not only teach children basic skills but also provide pleasant experiences with the computer.

The first software program is *Type Right,* manufactured by Commodore. This program will help everyone in the family learn to type, or it will give additional practice in typing skills.

Type Right is available in two versions—one for the Commodore 64 and one for the VIC 20. Below is a list of numbers and kinds of typing activities for each machine:

Activity:	Commodore 64:	VIC 20:
Introduction to finger placement	25	29
Key typing practice	12	20
Word typing practice	10	10
Sentence typing practice	2	2
Words per minute practice	5	5
Typing games	4	0

The lessons help with finger placement, speed and accuracy while the games challenge family members to think of and type words which contain certain combinations of letters or which will complete a phrase or a sentence. Scores are determined by the number of words typed in one minute minus the number of errors times five. (Note: One word is defined as a group of five keystrokes.)

It is interesting to know that many experts disagree as to where and when typing should be taught in relation to computer instruction. Some feel that it is a necessary skill which saves a great deal of time for the more important thought processes needed in logical thinking and problem solving. Others believe that children will learn typing skills when they see a need for them.

The *Type Right* program will enable parents to expose their children to typing, a skill which has become increasingly more important in this academic, technological world. Making typing a fun activity is a better way to introduce this much needed skill than to have it introduced as a monotonous activity which soon becomes a drudgery.

A second program titled *Chopper Math* would also be a good program to help the various age levels of children in a family. This program, also manufactured by Commodore and available on both the Commodore 64 and VIC 20, helps children with the basic mathematical operations of addition, subtraction, multiplication and division in a game like format.

The great part about this game is that it allows for different levels of ability. The children can choose whether they wish to use one or two digit numbers. Then the program lets them decide how they want to enter their answers—either left to right or right to left. At first glance many people wonder why this option is given, but after watching children work through math problems, it soon becomes apparent that children who are less experienced think of the answers one digit at a time going from right to left while more experienced students can often answer more easily going in the other direction.

Now the children enjoy this program from an entirely different viewpoint—they like the game format. Each game consists of trying to land helicopters without crashing. The sounds of the helicopter are very realistic as are the crashes. After three crashes, the game is over. In order to avoid crashing and land properly, the children must first answer a problem correctly as the helicopter begins to land. When the problem is answered correctly, the landing pad opens up. On the way to the landing pad the helicopter can be moved to erase as many stars and diamonds in the sky. By doing that, additional bonus points are earned which can be added to their scores.

One important *secret* to landing is making certain that the numbers line up exactly as they should. Otherwise—CRASH! Here are two examples:

This makes a perfect landing:

```
         5
        +6
        ——
        11
```

These cause the helicopter to crash:

```
     5                 5
    +6       OR       +6
    ——               ——
     11              11
```

Another important thing to know in this game is to encourage children to use the F7 key which holds the helicopter in one location while the children have time to figure out the answer. Usually children are reluctant to use any keys that appear to slow them down, but in this case they will soon see the advantage of using this pause key.

Both *Type Right* and *Chopper Math* come with manuals which explain how to load the programs and how to use them in more detail. Choosing either of these programs will meet two needs—those of the children who want to play games and those of their parents who want them to learn something worthwhile. Using them as a family in a positive learning environment makes them even better.

May you have a grand time learning and playing on the computer together! **C**

Sprites and COMAL on the Commodore 64

by Len Lindsay

COMAL, a versatile, "structured" programming language, was originally developed in Denmark for CBM computers and is in the public domain. It has recently been adapted to take advantage of the Commodore 64's graphics capabilities. This is a continuation of Len's discussion of COMAL graphics, which appeared in Issue 27.

COMAL is an advanced yet very easy to use programming language for the Commodore 64 computer. It is available on disk from the COMAL Users Group for only $15, and is in the public domain. In my last article, I presented a brief overview of the turtle graphics system included with COMAL. This allows you to take advantage of the many fantastic capabilities of the Commodore 64 computer. COMAL also has built in keywords that allow you to control the sprites. In this article I will give you a quick look at how easy it is to use sprites with COMAL.

As you may already know, there are eight different sprites available on the Commodore 64. COMAL uses sprite number seven for its turtle images. This leaves you sprites number zero through six to use. If you are not using the turtle you may also use sprite number seven. You have the same sprite capabilities with COMAL as with BASIC, except that in BASIC all dealings with sprites are via PEEK and POKE, while COMAL gives you a complete set of keywords to use. I have compiled a chart listing these sprite keywords along with an example and brief explanation of each. You may wish to refer to this chart as you read this article. It is on page 74.

Sprite Images

A sprite image consists of 64 bytes. These 64 bytes are used in the same manner as in BASIC. They can be a string in a DATA statement, read from a disk data file or put together in a program. Once you have a 64-byte image COMAL makes it easy to use.

Let's use two simple sprite images for an example. First, let's assign the 64-byte images to two string variables named BOX1$ and BOX2$. The first image will be a solid box, the second will be a striped box. COMAL allows you to manipulate the sprites in direct mode or within a program. Thus you can issue the following commands directly, or include line numbers in front of them, forming a program.

Sprite Definition

To define a sprite image in COMAL, use the word DEFINE. You then specify the definition number and the string of 64 bytes that contains its image. COMAL allows you to have 32 sprite definitions. These definitions (or images) are stored in a special area of memory, completely under COMAL control. You do not have to worry about where they are. You simply remember what number you assign to each image. Below, we assign our two box images to definition numbers zero and one:

```
AUTO
ØØ1Ø DRAW
ØØ2Ø DIM BOX1$ OF 64, BOX2$ OF 64
ØØ3Ø FOR TEMP=1 TO 63 DO
ØØ4Ø    BOX1$=BOX1$+CHR$(255)
ØØ5Ø    BOX2$=BOX2$+CHR$(15)
ØØ6Ø NEXT TEMP
ØØ7Ø BOX1$=BOX1$+CHR$(Ø)
```

```
0090 DEFINE 0,BOX1$

0100 DEFINE 1,BOX2$
```

Notice that the 64th byte must be a CHR$(0). This is a check byte used by COMAL. Notice how easily the image was assigned a definition number in the last two lines. Remember, these are just the definition of a sprite image. To identify a specific sprite with a sprite image the word IDENTIFY is used. Thus to identify sprites number zero and one with image definition zero we would add (the AUTO line number prompt for line 110 is there waiting for us yet):

```
0110 IDENTIFY 0,0

0120 IDENTIFY 1,0
```

Then to identify sprite numbers two and three with image definition number one we would add:

```
0130 IDENTIFY 2,1

0140 IDENTIFY 3,1
```

Now let's make sprite number zero white and put it near the middle of the screen:

```
0150 SPRITECOLOR 0,1

0160 SPRITEPOS 0,160,100
```

Now let's have sprite number one, yellow, slide across the screen:

```
0170 SPRITECOLOR 1,7

0180 FOR X=1 TO 300 DO
     SPRITEPOS 1,X,100
```

Next let's expand the red sprite in the center of the screen (expand it horizontally and vertically):

```
0190 SPRITESIZE 0,TRUE,TRUE
```

Sprite number two can now slide up the screen. Let's make it gray and expand its size vertically (a tall thin sprite):

```
0200 SPRITESIZE 2,TRUE,FALSE

0210 SPRITECOLOR 2,12

0220 FOR Y=1 TO 180 DO SPRITEPOS
     2,160,Y
```

Finally, sprite number three can slide diagonally across the screen, changing colors as it goes. Let's expand it horizontally (a short fat sprite) as well:

```
0230 SPRITESIZE 3,FALSE,TRUE

0240 FOR LOCATION=1 TO 180

0250   SPRITECOLOR 3,RND(0,15)

0260   SPRITEPOS 3,LOCATION,
       LOCATION

0270 NEXT LOCATION
```

Using COMAL Sprite Keywords

DATACOLLISION *<sprite number>, <reset collision flag?>*

DATACOLLISION 3, TRUE check if sprite three collided with data and reset the collision flag
returns TRUE if a collision with data occurred with this sprite

DEFINE *<sprite definition number>, <64 byte definition$>*

DEFINE 0, SHIP$ definition zero is assigned the previously defined string 64 bytes long named SHIP$
assign a sprite image definition—must be 64 bytes long

HIDESPRITE *<sprite number>*

HIDESPRITE 3 sprite three is now invisible
turn off the specified sprite—make it invisible

IDENTIFY *<sprite number>, <definition number>*

IDENTIFY 3,0 sprite three is given image definition number zero
give one of sprites zero to six an identity—note that sprite seven is the turtle and shouldn't be used at the same time as the turtle

PRIORITY *<sprite number>, <data priority?>*

PRIORITY 3, TRUE data has priority over sprite three image
determines if data has priority over the sprite image

SPRITEBACK *<number>, <number>*

SPRITEBACK 3,3 unknown effect
unknown effect

SHOWSPRITE *<sprite number>*

SHOWSPRITE 3 make sprite three visible
turn on specified sprite—make it visible

SPRITECOLLISION *<sprite number>, <reset collision flag?>*

SPRITECOLLISION 3, FALSE check if sprite three collided with another sprite and do not reset the collision flag
returns TRUE if a collision with another sprite occurred

SPRITECOLOR *<sprite number>, <color number>*

SPRITECOLOR 3,2 sprite three color is set to be red (2)
set the color of the sprite

SPRITEPOS *<sprite number>, <x coordinate>, <y coordinate>*

SPRITEPOS 3,160,100 sprite three in center screen x=160 y =100
position the sprite at the specified x,y location

SPRITESIZE *<sprite number>, <width expand?>, <height expand?>*

SPRITESIZE 3, TRUE, FALSE sprite three expands width only
set the size of the sprite specified (expand or not)

Now, let's stop. To stop the AUTOmatic line numbering, simply hit the RETURN key without typing anything else. Now to see the sprites in action simply say RUN.

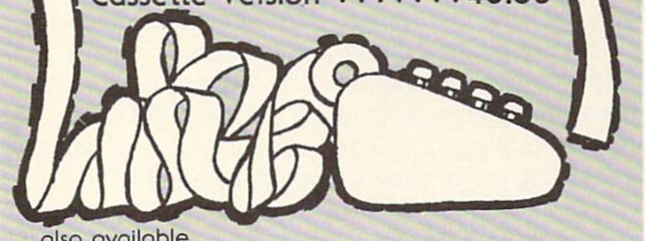
That was nothing special. But it did illustrate just how easy sprites can be when controlled by COMAL. To see the action again, simply type RUN again. To have the action keep repeating, type in these two lines:

```
165 REPEAT

28Ø UNTIL KEY$<>CHR$(Ø)
```

Now the action continues until you hit any key.

COMAL also allows you to specify whether data on the screen has PRIORITY over any of the sprites via the PRIORITY keyword. Collisions between sprites and data are reported via the DATACOLLISION function. Collisions between sprites are reported via the SPRITECOLLISION function. A sprite can be turned off with the HIDESPRITE statement, and can be turned on with the SHOWSPRITE statement. Refer to the chart of sprite keywords for correct syntax and examples. C

Len Lindsay is author of the COMAL Handbook *and heads up the COMAL Users Group. For more information on COMAL, you can write to him at 5501 Groveland Terrace, Madison, Wisconsin 53716.*



Random Thoughts:
Part 4: Bells, No Whistles

by Mark Zimmermann

You can make some handy rough estimates if you understand Gaussian distributions—otherwise known as bell-shaped curves.

In Part 3 of "Random Thoughts" we began to get involved with "continuous distributions" of random variables. We looked at how to display these continuous distributions graphically, and defined the "moments" of the distributions: the average (mean) value, the variance and its square root, the standard deviation and higher moments. This time, we're going to go into depth concerning the most important continuous distribution, the Gaussian. A Gaussian distribution is commonly called a "normal" distribution, and its graph is frequently referred to as a "bell-shaped curve".

Bells, Bells, Everywhere

Why should anyone care about a particular probability distribution over all others? And why, specifically, pick the bell-shaped curve for extra attention?

The bell-shaped curve is important because it seems to turn up wherever one looks in the world. Take a random sample of people, and plot their heights, or weights or IQs... you'll come up with something pretty close to the Gaussian distribution. Throw ten pennies on the floor and count how many come up heads... again, the probability falls within a few percent of the Gaussian. The noise you hear between radio stations on the dial tends to be uncorrelated "white noise" with values chosen moment-by-moment from a Gaussian distribution.

A rather fundamental theorem of mathematical probability, the "Central Limit Theorem", explains why the Gaussian distribution is so prevalent. The theorem states that (except in pathological cases) a random variable X when added up and averaged over many trials tends to give an average outcome that follows the Gaussian bell-shaped curve. Specifically, suppose that the mean value of the random variable X is M and its standard deviation is S. Then if you draw N random values of X, you will tend to find that the average of those values follows a Gaussian distribution, with mean M and with standard deviation S/SQR(N).

Going back to our penny-tossing example—if we throw ten coins, each of which is equally likely to be heads or tails (probability .5), what will the outcome be? A single penny produces an average of .5 heads per toss. The second moment, the mean square number of heads achieved, is also .5. So the standard deviation (defined as the square root of the difference between the second moment and the first moment squared) is SQR(.5−.5↑2)=SQR(.25)=.5. This makes sense. We sort of expect to see ½±½ heads per toss for each penny. But applying the Central Limit Theorem (or the "square-root" rule of thumb from last issue), we see that the result of tossing ten pennies should be an average of .5 heads per penny, plus or minus .5/SQR(10) per penny: overall, 5±1.58.... heads.

Note that the Central Limit Theorem only applies in the limit as the number of random trials being added up gets very large. Ten pennies isn't too large a number, but in fact the approximation suggested by the theorem is already pretty good—only off by a few percent.

The Standard Normal Bell

Figure 1 is a graph of what's called the "standard normal curve", a Gaussian distribution centered on the origin (mean value zero) and with standard deviation equal to one. The mathematical formula that produces this curve is:

```
P(X) = EXP(−X↑2/2)/SQR (2*PI)
```

where pi=3.14159. On the average, this probability distribution P produces values of X that average out to zero, but that fluctuate ±1 about that average value.

If we can generate random numbers X that follow the standard normal distribution of Figure 1, then we can easily get *any* desired Gaussian distribution of random numbers. How? Simply by using the "stretch and shift" approach discussed in my earlier articles. The philosophy is simple: (1) generate an X that follows the standard normal distribution P above, (2) multiply the value of X you got by the desired standard deviation of the Gaussian curve you want (3) add the mean value of the desired Gaussian curve.

Step two above is a "stretch" operation on the bell-shaped curve of Figure 1, which widens it (if your multiplier is bigger than one) or narrows it (if multiplier is less than one). Step three is a "shift", that slides the whole curve left or right until the distribution is centered on the mean value you want.

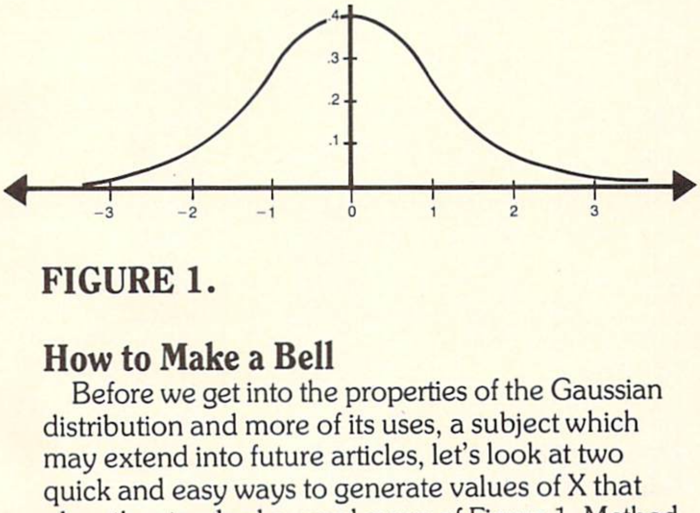


FIGURE 1.

How to Make a Bell

Before we get into the properties of the Gaussian distribution and more of its uses, a subject which may extend into future articles, let's look at two quick and easy ways to generate values of X that obey the standard normal curve of Figure 1. Method 1 is the simplest: just call the function RND(1) twelve times, add up the results and take away six. A trivial one-line routine to do it is:

```
X=0:FOR I=1 TO 12:X=X+RND(1): NEXT I:
X=X-6
```

Or, if you want a little greater speed, you can write out the FOR-NEXT loop explicitly, with twelve calls to the RND(1) function added up.

Why does it work? If you remember from last time, when we looked at the distribution of random numbers given by RND(1), each number has a mean value of 0.5 and a standard deviation of 1/SQR (12). The Central Limit Theorem suggests that if we were to add up twelve of these values we should get something with a mean of six and with a standard deviation of SQR(12) times the standard deviation of each number being added in: a result of one! So, take away six to shift the mean outcome back to zero and you've got your standard normal X. (A comment on nomenclature: variables that follow the standard normal distribution are sometimes called "normal deviates", a funny sounding term.)

Method 1 is *not* exact, but it comes pretty close and is quick and easy to program. It will never give you a result for X that is outside the range −6 to +6, but such far-out values only come up one time in 50,000 or so in a perfect standard normal distribution, so that's not bad most of the time.

If you demand a theoretically *exact* distribution of normally-distributed random variables X, then consider Method 2. It gives you two Gaussian random numbers every time you call it (you can throw one away if you don't need it). But it has the disadvantage that it requires a LOG and a SQR operation, which tend to be rather slow in BASIC. A sample routine to execute Method 2 and return random numbers X and Y is:

```
100 U=2*RND(1)-1:V=2*RND(1)-1:R=
    U*U+V*V: IF R>=1 THEN 100:REM TRY
    AGAIN UNTIL R>1
120 S=SQR(-2*LOG(R)/R) :X=U*S:Y=V*S
```

How does Method 2 work? It's based on a funny two-dimensional "stretch" operation that comes out of some advanced geometry. Line 100 of the mini-program above generates two random variables, U and V, each of which is equally likely to fall anywhere within the interval −1 to +1 before the test of R. (The 2*RND(1)−1 is a "stretch-and-shift" to take the usual range of RND(1) results, zero to one, and make it cover the −1 to +1 uniformly.) If you imagine plotting U along a horizontal axis and V along a vertical one, you see that U and V fall inside the square centered on the origin with sides of length two.

Then, still in line 100, we compute R=U↑2 + V↑2, the distance of the point (U,V) from the origin. If R is greater than or equal to one, we reject the chosen values of U and V (both of them!) and loop back to try again. On the average, about 80% of the first choices for U and V are ok. The good choices correspond to points (U,V) which lie within the unit circle centered on the origin.

Finally, when we have a value of R less than one, we compute the strange looking but perfectly legal "stretch" function S=SQR(−2*LOG(R)/R). The BASIC function LOG takes the natural logarithm of R, and since R is less than one, this gives a result between zero and minus infinity. (If you have the *very* bad luck to have a RND(1) function which gives you both U and V equal to zero, the LOG function will crash. This happens so incredibly rarely that it's not worth testing for in line 100.) The stretch S is then multiplied times the original random numbers U and V to give the normal deviates X and Y. This kind of multiplying by a constant is a radial motion on the graph paper.

Some Properties of the Bell

The Gaussian distribution has many strange and wonderful properties, most of which you probably shouldn't care about. But there are some "rules of thumb" that aren't bad to have in the back of your mind when you deal with a Gaussian. They let you make quick rough estimates of how often some numbers come up, for example.

As you recall from last time, the "probability distribution curve" we graphed in Figure 1 tells you the relative likelihood of getting a random choice X with any given value. The chance of seeing any particular X, say X=1.23456789, is tiny (since there are so many choices for X, any individual choice comes up rarely). But if you ask instead, "What is the chance of getting X in the *range* between A and B?", the answer is simply the area under the probability distribution curve between points A and B on the horizontal axis.

Table 1 is an abbreviated table that should help you estimate how likely it is to get a normal deviate X (obeying the standard normal distribution) in various ranges. If you need more precision, or values not included in Table 1, you can try a Monte Carlo experiment (see last article's discussion) or seek out larger tables in most common mathematics books. Be sure to check the definitions being used in the book you consult. Sometimes they disagree by factors of two pi, or sign differences, so compare the results with Table 1 and with common sense. Sometimes the tables you look at will be labeled "Error Function" or "Erf(x)" instead of "Normal Probability Function".

Table 1.

Z	Chance of getting X Less than **Z**	Chance of getting X between −**Z** and +**Z**	Value of distribution P (Fig. 1) at **Z**
0.00	0.5000	0	0.3989
0.25	0.5987	0.1974	0.3867
0.5	0.6915	0.3830	0.3521
0.75	0.7734	0.5468	0.3011
1.	0.8413	0.6826	0.2420
1.5	0.9332	0.8664	0.1295
2.	0.9772	0.9544	0.0540
3.	0.9987	0.9974	0.0044
4.	0.9999+	0.9999+	0.0001

The information in Table 1 can be extended to cover negative values of Z using the symmetry of the bell-shaped curve. Suppose you're interested in the chance of picking X less than −2, for example. By symmetry that is the same as the chance of picking X greater than +2. The second column of Table 1 says that the chance of getting X less than +2 is 0.9772, so the chance of getting X greater than +2 must be 1−0.9772=0.0228, and that is the answer to your original question. (You may notice that the column"chance of getting X between −Z and +Z" is simply twice the value in the previous column less one. If you draw a few examples of this graphically, in terms of areas under the curve, you'll see why it has to be so.)

What should you remember from Table 1? Only a few things. I think it's useful to recall that about 68% (0.6826) of the values of X fall within ±1 standard deviation of the mean, and that 95% fall within ±2 standard deviations. This tells you that even though the bell-shaped curve doesn't look too highly peaked or concentrated in the center, it's actually not that spread out.

Knowing these two numbers (and maybe also that 99^{+}% falls between −3 and +3 standard deviations) lets you make a lot of handy rough estimates. Go back to our example of penny-tossing. We found that ten pennies when thrown should come up roughly 5±1.6 heads each time. Table 1 then tells us that roughly two-thirds of the time we should expect to see four, five or six heads. The exact answer for pennies is 65.6%, not far off. Within two standard deviations, 5±3.2 heads, we expect results to fall about 95% of the time, and the exact answer is that two through eight heads come up 97.8% of the time... again, not bad!

Next time, we'll take a look at some more examples of Gaussian distributions, work out more details, answer some questions and (if there's room) move on to other distributions that cover rare events, radioactive decay, etc. **C**

Number Crunching on the 6502

by Craig R. Hessel

This follows up Craig's "Public-Key Cryptography for Commodore Microcomputers" (Issue 26) by describing LIAL, the language in which those cryptographic subroutines were written. A short sample program that computes large Fibonacci numbers is presented here, along with general directions for running LIAL programs on 6502/6510 microcomputers. Keep the previous article handy for reference, if you have it.

Why LIAL?

LIAL is a compact language for computing with large integers. Instructions are one or two bytes long, although in a typical program, one-byte instructions will predominate. The language provides for a large integer accumulator and 16 large integer variables. Programs have dynamic control over the length of these integers (1-255 bytes). For handling small numbers, a general purpose register and 16 auxiliary registers are available. The computational instructions cover multiplication, division, addition, subtraction, logical operations, shifts and pseudorandom number generation.

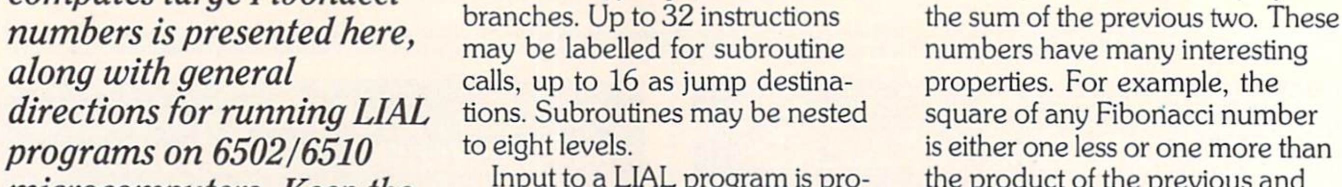
Program flow is controlled with subroutines, jumps, and relative branches. Up to 32 instructions may be labelled for subroutine calls, up to 16 as jump destinations. Subroutines may be nested to eight levels.

Input to a LIAL program is provided by initializing registers and variables before execution. Output is read after execution halts. Since LIAL is strictly computational (no I/O instructions), it was possible to write a portable, relocatable interpreter for the language. The interpreter is contained in "Cryptocode 5/83" (Issue 26).

See the accompanying LIAL Instruction Set for the opcode, name and operation description of each instruction. In the table, 6502-like mnemonics generally indicate 6502-like operations.

Fibonacci and Friends

The Fibonacci number sequence (0, 1, 1, 2, 3, 5, 8, 13, and so on) has a long and colorful history, dating back nearly eight centuries to 1202 when Leonardo Fibonacci first used it to estimate the population growth of rabbits. Each Fibonacci number, after the first two, is just the sum of the previous two. These numbers have many interesting properties. For example, the square of any Fibonacci number is either one less or one more than the product of the previous and following Fibonacci numbers (5*5=25, 3*8=24). Perhaps more surprisingly, the ratio of consecutive Fibonacci numbers approaches the "golden ratio" (1+SQR(5))/2 = 1.618+ as the numbers grow larger (3/2 = 1.500, 5/3 = 1.667–, 8/5 = 1.600, 13/8 = 1.625, and so on). The following LIAL program churns out a one to 255-byte Fibonacci number, depending on the initial value of the Z-register:



```
1D     START LDN Z     ;SET INTEGER SIZE TO Z BYTES
30           JSR FIBON ;SUBROUTINE FINDS FIRST FIBONACCI
10           END       ;  NUMBER THAT FILLS ACCUMULATOR
12     FIBON ZER A     ;INITIALIZE V0 WITH
80           STA V0    ;  ZERO AND ACCUMULATOR
14           INC A     ;  WITH ONE
A0     LOOP  EXA V0    ;SWAP ACCUMULATOR WITH V0
F0           ADD V0    ;REPLACE SMALLER NUMBER WITH SUM
0C FC        BPL LOOP  ;LOOP UNTIL LEADING BIT IS SET
11           RTS       ;ACCUMULATOR HOLDS RESULT
```

Running a LIAL Program

This section gives a step-by-step procedure for running the above program, or any LIAL program, on a 6502/6510 microcomputer. A machine language monitor is useful for much of the following.

1. Load the interpreter plus parameters. That is, load $16EC-$1B19 from the hex listing of "Cryptocode 5/83" (Issue 26). The interpreter is relocatable and may be moved elsewhere, if desired. The memory references that follow correspond to the original location of the interpreter.

2. Load the LIAL program. A convenient place to locate LIAL programs is immediately following the interpreter. For example, put the Fibonacci program into locations $1B1A-$1B24.

3. Set pointers to the LIAL program, 6502 storage and LIAL storage. These pointers are located at $17D0-1, $17CC-D and $17CE-F respectively and specify the starting locations of the three areas relative to the start of the parameter-interpreter package ($16EC). For example, set pointer $17D0-1 (low-high) to $042E to let the interpreter know where the Fibonacci program begins ($16EC+$042E=$1B1A). The half-page 6502 storage area may as well be put immediately after the Fibonacci program, so set pointer $17CC-D (low-high) to $0439 ($16EC+$0439=$1B25). Finally, LIAL storage may as well immediately follow 6502 storage so set $17CE-F (low-high) to $04B9 ($16EC+$04B9=$1BA5). The amount of LIAL storage required depends on the program. The accumulator and then variables 0-15 are packed upward in memory from the start of storage. The Fibonacci program uses only the accumulator and variable zero at a maximum of 255 bytes each, so 510 bytes are required. Programs should generally use the lowest index variables available to save space.

4. Fill the label table as needed for jumps and subroutine calls. High label bytes are located at $16FC-$171B; low bytes are at $171C-$173B. Each label is the address of the designated instruction relative to the start of the LIAL program. For the Fibonacci program put $00 into $16FC and put $03 into $171C since label zero (the FIBON label) is three bytes from the START instruction.

5. Initialize registers and variables as needed. The interpreter initializes the LIAL Z-register from location $17EB and the auxiliary registers from the table at $16EC-16FB. These locations may be given any values, but note that a zero in $17EB is a special flag for the interpreter rather than an initial value for the Z-register (see next section). The interpreter does not zero out LIAL storage initially, so any data placed there will initialize the accumulator/variables. The Fibonacci program needs only Z-register input, so place the desired value (1-255) in $17EB.

6. Call the interpreter. Either SYS 6122 or JSR $17EA will do. With Z-register input of 255, the Fibonacci program takes about 42 seconds to run and in the process executes the three-line program loop nearly 3000 times.

7. Read results. LIAL storage holds the large integer results and 6502 storage holds final register values. In particular, the third byte of 6502 storage holds the final Z-register value and the third block of 16 bytes there holds the final auxiliary register values. For Z-register input of one, the two-hex-digit Fibonacci result, $90, is found at $1BA5, for input of 255, the 510-hex-digit result, $94ACD946 ... D0706EB9, is found at $1BA5-$1CA3.

Enabling Stop/Continue

This section describes how to enable the stop/continue feature of the interpreter. This feature is very useful during the debugging of LIAL programs, but it does require some machine-dependent code.

First, a short 6502 subroutine is needed for detecting an external "break" signal such as a depressed key. The subroutine may freely use 6502 registers (X, Y, A) but may not tamper with any memory area used by the interpreter. In addition, the subroutine should not rely on output from ROM interrupt-handling routines, since

LIAL Instruction Set

TWO BYTE		
OP	NAME	ACTION
00 xy	MUL	[A(high),Vx(low)] ←A+Vx*Vy

01 xy	DIV	A←remainder of,
		Vx←quotient of
		[A(high),Vx(low)]
		/Vy

02 xy	NOP	None
03 xy	NOP	None
04 xy	NOP	None
05 00	LDAN	A←N
05 xy	LDA#	A←xy (xy≠00)
06 xy	LDZ#	Z←xy
07 xy	BRN	PC←PC+xy
08 xy	ZNE	PC←PC+xy if Z≠0
09 xy	ZEQ	PC←PC+xy if Z=0
0A xy	BCC	PC←PC+xy if C=0
0B xy	BCS	PC←PC+xy if C=1
0C xy	BPL	PC←PC+xy if A≥0
0D xy	BMI	PC←PC+xy if A<0
0E xy	BNE	PC←PC+xy if A≠0
0F xy	BEQ	PC←PC+xy if A=0

NO INDEX		
OP	NAME	ACTION
10	END	Ends program
11	RTS	Pulls PC from stack
12	ZERA	A←0
13	DECA	A←A−1
14	INCA	A←A+1
15	NEGA	A←−A
16	ASLA	Shifts A left one bit, LSB←0, C←old MSB
17	LSRA	Shifts A right one bit, MSB←0, C←old LSB
18	RNDA	A←random, S changes, 0≤random≠old A
19	RRAZ	Shifts A right one byte, HB←Z, Z←old LB
1A	RLAZ	Shifts A left one byte, LB←Z, Z←old HB
1B	DECZ	Z←Z−1
1C	INCZ	Z←Z+1
1D	LDNZ	N←Z (N←1 if Z=0)
1E	LDSZ	Shifts S left one byte, LB←Z
1F	BRNZ	PC←PC+Z

ONE INDEX		
OP	NAME	ACTION
0x	----	Two byte
1x	----	No index

2x	JSR	Pushes PC
		on stack,
		x←x+16,
		PC←Lx

3x	JSR	Pushes PC on stack, PC←Lx
4x	JMP	PC←Lx
5x	STZ	Rx←Z
6x	LDZ	Z←Rx
7x	NOP	None
8x	STA	Vx←A
9x	LDA	A←Vx
Ax	EXA	A↔Vx
Bx	EOR	A←A⊻Vx
Cx	ORA	A←A∨Vx
Dx	AND	A←A∧Vx
Ex	SUB	A←A−Vx
Fx	ADD	A←A+Vx

A = Large integer accumulator
V = Large integer variable (16)
Z = General purpose register
R = Auxiliary register (16)

N = Length register
PC = Program counter
MSB = Most significant bit
LSB = Least significant bit
L = Label address (32)

S = Seed register
C = Carry bit
HB = Highest byte
LB = Lowest byte

Notes:

1) MUL operands and results are treated as non-negative. The result is invalid if x=y. The DIV result is invalid if x=y, if A<0, if Vy≤0, or if the quotient exceeds the current integer size.
2) LIAL branches (including BRNZ) affect program flow just as 6502 branches do by treating the branch byte as an integer in the range [−128, 127].
3) Only ASLA and LSRA affect the carry bit.
4) RNDA treats the accumulator as a non-negative number. Before generating pseudo-random numbers in a LIAL program, use LDSZ to give the four-byte seed register an initial non-zero value.
5) RRAZ and RLAZ act as the 6502 instructions RORA and ROLA act, but in a byte-wise rather than bit-wise fashion.
6) When LDNZ sets the current integer size, pointers to integer variables are adjusted, but the contents of the integer storage area are not changed.

interrupts are inhibited during calls to the interpreter. If a "break" is detected, the 6502 equality flag should be cleared; otherwise, that condition flag should be set.

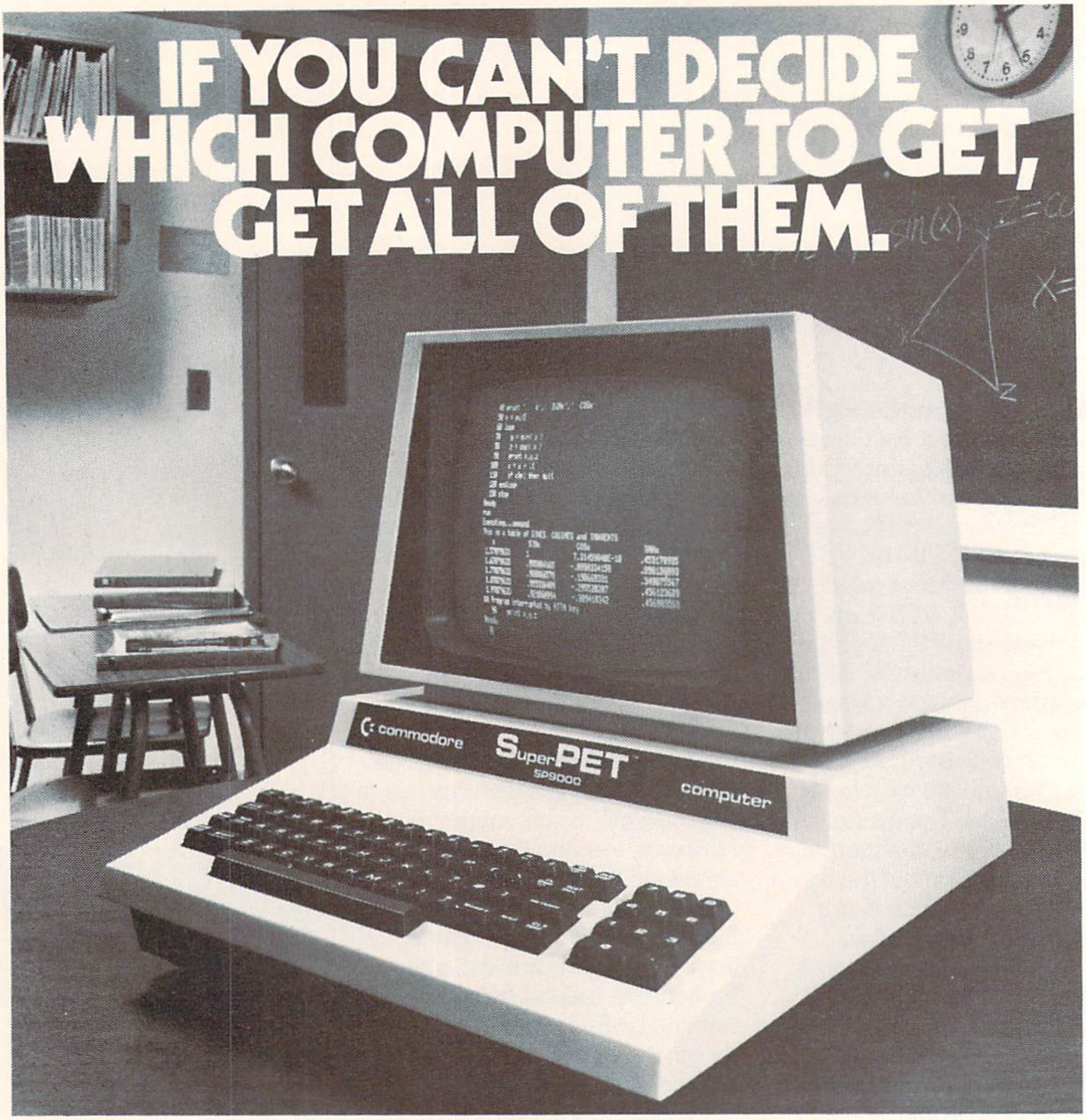
On the original ROM 8K PET, the following subroutine will do nicely: INC $E812, RTS. Location $E812 is tied to the PET keyboard. Except during an interrupt, $E812 scans several bottom row keys including the STOP key. If none of these keys is depressed, $E812 contains 255, otherwise, it holds some other value. The instruction INC $E812 will not change the value of that location but the condition flags will be affected as if the increment operation had taken place. On VIC or 64 computers some similar such "break" detection subroutine should work here.

Next, the subroutine must be linked to the interpreter so that it can be called before each LIAL instruction is executed. Do this by setting the pointer $17A0-1 to the address of the new subroutine relative to $16EC. For example, on the PET put the subroutine ($EE, $12, $E8, $60) into, say, unused locations $17E6-9. Then set $17A0-1 (low-high) to $00FA ($16EC+$00FA=$17E6).

Finally, enable the stop/continue feature by setting the flag at $17AC to $FF. At this point any LIAL program may be halted by sending a "break" signal until the signal is detected. On the PET, this may require holding the STOP key down for 16 seconds (the time needed to divide 255-byte integers) in the worst case. Later, to continue execution, set $17EB to special flag $00 and then call the interpreter. Setting $17AC back to $00 will disable the stop/continue feature. Then the "break" detection subroutine will not be called and, if $17EB holds $00, the interpreter will do nothing.

In Closing

As a closing note, LIAL was designed with a particular application in mind—public-key cryptography. However, the language is versatile enough for many number-crunching problems. Some of these problems, like generating large Fibonacci numbers, could be handled instead with slow-running BASIC programs. But for others, such as finding large prime numbers or computing pi (3.14159...) to many decimal places, BASIC is too slow and machine language is unwieldy. For these, a high-precision language like LIAL is a practical alternative. **C**



THE SUPERPET.™ 5 LANGUAGES

It's not easy finding a microcomputer that can fulfill all of your requirements. Unless you've discovered the Commodore SuperPET.™

The SuperPET is the *only* microcomputer that comes with five of the most powerful structured languages: MicroBASIC, MicroAPL, MicroFORTRAN, MicroPASCAL, and MicroCOBOL.

We think you'll agree it's far and away the most complete computer you can buy for your school.

commodore
COMPUTER

Converting VIC 20 Programs for the 64

by Jim Gracely

Here it is, finally, the article everybody seems to want. After the conversion article (64-PET) in the March Commodore, *questions and requests have come pouring in. In this article, we'll take a look at how* some *programs can be converted from the VIC 20 to the Commodore 64.*

In order to approach this subject with some sense of direction, I have divided the whole operating system of the VIC 20 into five sections:

1) BASIC: User RAM, commands, syntax and keyboard
2) The Screen: Screen and color locations, background and border colors
3) Programmable Characters: Setting up, number available
4) Music: Notes, voices and register locations
5) Other Memory Locations: Hi-res graphics, Kernal routines, BASIC and ROM entry points

BASIC

The BASICs used on the VIC 20 and the 64 are very similar. In fact, just about the only differences are the amount available and the starting and ending locations of user RAM.

BASIC begins at 4096 on the unexpanded VIC 20 and ends at 7679. This represents 3583 bytes of RAM available for programming. On the 64, BASIC begins at 2048 and continues to 40960, representing 38913 bytes of RAM for programming. Obviously there is no problem with fitting a large VIC 20 program into the 64.

One of the nice features of both the VIC 20 and the 64 is called auto-relocation of programs loaded into memory. This means that any program written on the VIC 20 and saved will load correctly (without any adjustments) into the 64.

The version of BASIC used by both machines is identical. The commands themselves are all identical and any syntax that works on the VIC will work on the 64 (there are a couple of differences going the other way, *i.e.*, an INPUT prompt can be up to 38 characters on the 64 but only 22 on the VIC).

The keyboard and the keys are identical on both machines. The 64 uses the Commodore logo key to access eight additional cursor colors, but that won't affect us. Although the keyboards are identical, the key matrix values (PEEK(197)) of the keys are different. The Fall, 1983, *Power/Play* contains the table of values for the 64 and the *VIC 20 Programmer's Reference Guide* has the table for the VIC.

Now, here is our first level of conversion: Any program written using the standard set of BASIC commands (no POKE, PEEK, SYS, or USR) will run right from the VIC 20 to the 64.

This might not sound like much, but many of the BASIC programs written before televisions and computers were connected used this standard set of commands. There are many books of such programs (usually they have the word BASIC in the title without the name of a computer or manufacturer).

The Screen

The screens for the VIC and the 64 are very similar. The starting locations are different and the number of rows and columns are different, but the basic operation of each is the same. All of the screen POKE values for characters are the same and the method of putting characters onto the screen is the same. Converting background and border colors from the VIC to the 64 is mostly a matter of switching memory locations.

The screen that you watch on the television or monitor is actually the display of a series of memory locations. The screen on the VIC 20 starts at memory location 7680 (upper left) and continues 506 bytes to location 8185 (lower right). There is also a section of memory (506 bytes) which stores the color for each screen memory location. On the VIC, this memory is from 38400-38905. On the 64, these same memory locations are from 1024-2023 (screen) and 55296-56295 (color). As you may have noticed, the 64 screen uses 1000 memory locations.

programmer's tips

To convert screen POKEs from the VIC 20 to the 64, it looks like you just have to subtract 6656 (7680−1024) from the screen memory location and add 16896 (55296−38400) to the color locations. This is correct, except that the width of the two screens is different. The VIC 20 screen is 22 columns wide, and the 64 screen is 40 columns wide. This is important because a POKE to 7680+22 will appear on the first column of the second row on the VIC 20. Simply subtracting 6656 from this value will place the poke at column 22 on the first row of the 64. So we need a way to change rows on the 64 after 22 columns.

Some programs are written with provisions for converting the size of the screen. One way programmers do this is by using the following lines of code:

```
10 BS = 7680 : BC = 38400 : REM Set base screen and color
locations
20 ROW = 4 : COL = 17 : REM Choose row and column
30 POKE BS + 22*ROW + COL, 83 : REM Poke a heart at row and col
40 POKE BC + 22*ROW + COL, 2  : REM Poke red color to heart
```

If a program uses this kind of POKing to the screen (the row and column variables can change), here is the way to fix it. The base locations for the 64 should be substituted in line ten. Lines 30 and 40 multiply the row by 22 because each row on the VIC 20 is 22 columns (the width of the screen). To change these to lines for the 64, just change the 22 to 40 (the width of the 64 screen) in lines 30 and 40. The same program for the 64 would be:

```
10 BS = 1024 : BC = 55296
20 ROW = 4 : COL = 17
30 POKE BS + 40*ROW + COL, 83
40 POKE BC + 40*ROW + COL, 2
```

Not all programs are this neat, however. Sometimes graphics are just POKEd all over the screen. If this is the case, here is a short program which allows you to input a screen POKE value for the VIC 20 and will convert it to the correct screen and color POKEs for the 64. This program will automatically fix the problem of different screen widths.

```
10 PRINT"[CLEAR]SCREEN[SPACE]LOCATION[SPACE]FOR[SPACE]THE
   [SPACE]VIC";:INPUT P
20 ROW=INT((P-7680)/22):COL=(P-7680)-22*ROW
30 D=40*ROW+COL:SP=1024+D:CP=55296+D
```

```
40 PRINT"SCREEN[SPACE]LOCATION[SPACE]FOR[SPACE]64[SPACE]
   IS"SP
50 PRINT"COLOR[SPACE]LOCATION[SPACE]FOR[SPACE]64[SPACE]
   IS"CP
```

Background and border colors are POKEd into location 36879 on the VIC. The number to be POKEd is the background number times 16 plus the border color. On the 64, a different location is used for each of the border (53280) and the background (53281) colors. Given a value to be POKEd into 36879, this little program will calculate the correct POKEs for the 64:

```
10 PRINT"[CLEAR]WHAT[SPACE]IS[SPACE]THE[SPACE]VALUE[SPACE]
   POKED[SPACE]INTO[SPACE]36879";:INPUT P
20 BD=INT(P/16):BC=P-16*BD
30 PRINT"POKE[SPACE]53280,"BC
40 PRINT"POKE[SPACE]53281,"BD
```

Now you can convert any program which has POKEs and PEEKs only related to the screen and doesn't use any SYS or USR commands.

Programmable Characters

Programmable characters are one of the most fun and interesting ways to enhance game programs on the VIC 20. Although the memory locations for the VIC 20 and the 64 are quite different, the method for setting up and creating the characters is very similar.

The VIC 20 requires just three steps to relocate a set of 64 characters from ROM to RAM:

```
10 POKE 52,28 : POKE 56,28 : CLR

20 FOR I = 7168 TO 7679 : POKE I, PEEK(I+25600) : NEXT

30 POKE 36869,255
```

These same steps on the 64 are a little more complicated. They are:

```
10 POKE 52,48 : POKE 56,48 : CLR

20 POKE 56334, PEEK(56334) AND 254 : REM Turn off keyscan clock

30 POKE 1, PEEK(1) AND 251 : REM Switch out I/O
```

```
40 FOR I = 0 TO 511 : POKE I + 12288, PEEK(I + 53248) : NEXT

50 POKE 1, PEEK(1) OR 1 : REM Switch in I/O

60 POKE 56334, PEEK(56334) OR 1 : REM Turn on keyscan
```

This puts the starting point of the characters at 12288 in RAM. The VIC 20 *Reference Guide* puts the starting point at 7168. The difference between these two locations is 5120. Adding 5120 to any POKE location of data into the programmable character memory in a VIC 20 program will convert it for the 64.

Now we have an additional set of programs that can be converted. The only programs which are still problems are those with music, SYS commands and POKEs into non-screen locations.

Music

Music is one area that is difficult to convert from VIC 20 to 64 because the 64 is much more complicated. There is a way to convert POKE values of notes on the VIC 20 to the 64; however, the procedure is a little complicated. If you will bear with me, I'll try to go through it slowly and explain each step.

First let's review music on the VIC 20. The VIC chip has a register for volume (36878) and four registers for the notes of the four voices (36874-36877). To play a note, you turn on the volume (POKE 36878,15) and POKE a note value (from 128 to 255) into a register (*i.e.*, POKE 36874,135). A note is now flowing out of the speaker.

Music on the 64 has a couple of differences. A special chip in the 64 called SID creates all the music on the 64. Like the VIC 20 it has one register for volume (54272); however, it has two registers for each of the four voices (for a total of eight). In addition, each voice has a control, an attack/decay and a sustain/release register. These registers must be initialized before any notes can be played. Here is a simple routine that sets the volume and initializes all of these registers:

```
10 V=54296:S1=54272:REM SET CONSTANTS
20 POKE S1+5,17:POKE S1+6,243:REM SET ATTACK/DECAY AND SU
   STAIN/RELEASE
25 POKE S1+4,33:REM SET WAVEFORM - SAWTOOTH
30 POKE V,15:REM SET VOLUME
```

Once these values are all set, we can start to think about the notes. The first step in converting the notes to the 64 is to change the note values into their true frequencies. The formula for this is on page 216 of the *VIC 20 Programmer's Reference Guide:*

$$\text{Frequency} = \text{Voice clock} / (127 - X)$$

X is a value from zero to 127, which represents the note. The highest bit is used to turn the note on, so the values that are actually POKEd are X + 128. We will call "P" the POKE value of the note. Here it is equal to X + 128, so X equals P minus 128. Now, substituting X into the previous equation we find:

$$\text{Frequency} = \text{Voice clock} / (127 - (P - 128))$$

or

$$\text{Frequency} = \text{Voice clock} / (255 - P)$$

The value of the voice clock comes from page 217 of the *Reference Guide.* We'll use the value for voice one because it is within the range of the 64. Now, we can substitute the value of 3995 for the voice clock:

Frequency = 3995 / (255−P)

We can now compute the frequency for any POKE value from 128 to 254 (note that if the POKE value is 255, a ?DIVIDE BY ZERO error will occur).

On the 64 there is a similar equation that we can use:

POKE value = SID clock * Frequency / System clock

The value of the SID clock is 16777216, and the value of the system clock is about 1Mhz (1000000). The SID clock divided by the system clock is just about 16.7. The value for frequency in the 64 equation is the same frequency as in the VIC 20 equation. Because of this, we can substitute the VIC 20 equation in the 64 equation:

POKE value = 16.7 * 3995 / (255−P)

To solve the problem when P=0, we can use 256 rather than 255 in the denominator of the equation. This won't affect the overall tones at all.

We can now find the POKE value on the 64 for almost any POKE value on the VIC 20. The only exception is when P=0. A little bit of extra programming can overcome this problem. A POKE or (P) of zero is used on the VIC 20 to turn a note off. The easiest way to fix this is to watch for P equal to zero. When it occurs, use a zero for the POKE value on the 64. It won't turn the voice off, but it will produce a note so low that it won't bother anyone.

All we need now is a short program to set up the SID values on the 64 and to READ and convert the data statements. Here is a program to do just that:

```
10 V=54296:S1=54272:REM SET CONSTANTS
20 POKE S1+5,17:POKE S1+6,243:REM SET ATTACK/DECAY AND SU
   STAIN/RELEASE
25 POKE S1+4,33:REM SET WAVEFORM - SAWTOOTH
30 POKE V,15:REM SET VOLUME
40 C=3995*16.7
50 READ P:IF P=-1 THEN 120
60 READ D
70 PK=C/(256-P):REM COMPUTE POKE VALUE
75 IF P=0 THEN PK=0
80 H=INT(PK/256):L=PK-256*H
90 POKE S1+1,H:POKE S1,L:REM COMPUTE HIGH-BYTE AND LOW-BY
   TE
100 FOR Z=1 TO D:NEXT
110 GOTO 50
120 POKE V,0:END
130 DATA 217,400,213,400,223,400,227,200,234,200,230,400,
    227,200
140 DATA 234,200,230,400,223,400,227,400,217,400,213,600,
    -1
```

This is the theory of converting music from the VIC to the 64. The program and theory presented here will allow you to "play" a list of music data statements written for the VIC on the 64. The 64 has so many additional options available for adjusting the sound of the music that it is unlikely that you will be satisfied with this procedure for long.

Other Memory Locations

Now, what's left? High resolution graphics have not been converted yet. While it is possible to convert VIC hi-res programs to the 64, I don't think that it is very practical. The VIC is limited to an eight-by-eight character block when working in hi-res. This is a 64-by-64 bit area or a total of 4096 pixels (bits). The 64 allows a hi-res screen which is almost 16 times bigger! (The full screen—200 by 320 bits or 64000 pixels.) The method of plotting the bits is the same, but why settle for a little square when you can have the whole screen?

What about kernal routines? Did you ever read the little paragraph on page 182 of the *VIC 20 Programmer's Reference Guide*? You know, the one that says "By taking advantage of the ... [kernal] ... routines ... you will ... make it easier to translate your programs from one Commodore computer to another". Well it's true! The kernal jump addresses are identical. So all of you who struggled through using kernal routines in your BASIC and machine language programs can sit back and pat yourselves on the back.

BASIC and ROM entry points? There are some minor differences in the ROM entry points. However the kernal is located at 57344-65535 ($E000-$FFFF) on both machines and many of the entry points are identical. The BASIC ROM entry points are also mostly the same except that all the addresses are 8K ($2000) higher in the VIC 20 than in the 64. To adjust VIC 20 addresses for the 64, simply subtract 8192 or $2000.

Machine language? What is it about machine language that throws everyone for a loop? Just think of it as a machine level BASIC. If the machine language routine is located in the cassette buffer (828-1020 on both machines) and uses only kernal routines, it will work on either machine. Unfortunately, most machine language programs use many screen and other memory locations. In this case, the same conversions which we have been discussing also apply to machine language.

Well, there's quite a pile of information! Don't expect to be able to convert every program from the VIC 20 to the 64. However, with a little patience, many programs can be converted. Times like this are when the importance of good programming techniques become apparent. An off-the-wall, non-documented program for the VIC 20 will probably never be converted to the 64. On the other hand, a well structured, documented program will be easy to modify to run on the 64.

Next issue, I'll go step-by-step through the procedure of converting one or two VIC 20 programs to the 64. In the meantime, try some of your own. Have fun and good luck!! C

PETSpeed Tips

by Joe Rotello

Once again, our PET/CBM expert is back with more tips on using his favorite compilers.

From time to time, it is important to define words, phrases and processes that are used in programs like PETSpeed and the Integer Basic Compilers. If you have any questions, let us know.

What is the difference between PETSpeed and Integer Basic Compilers?

PETSpeed is a compiler that takes a BASIC program as it sits, without requiring the user to make any special changes in their BASIC program. It then refines the BASIC program, making it much more efficient, and then compiles it down to disk as a transformed program. It is not fully in machine language. Rather, the transformed program is now in a "speed code" of its own, with its own speed code interpreter saved along with the program. One might say that the program is now somewhere between PET BASIC and machine code. An example method for speed improvement used in PETSpeed is the ability to address program variables in much the same way as the 6502 CPU chip would do it in machine language.

An Integer Basic Compiler (IB for short) is a completely different animal, yet similar in some respects. IB does not take an existing CBM BASIC program and change it. Rather, the user actually programs in a BASIC language that is very similar to CBM BASIC. Some commands available in CBM BASIC do not exist in IB, but many commands, speed features and automatic functions exist in IB's version of BASIC.

IB then transforms the IB BASIC source code into ACTUAL 6502 machine object code and stores it to the disk. IB places various limitations on variables and subscripts (126), total number of variables (110), program lines (250 maximum) as well as the fact that all subroutines in the source code are placed *after* the main IB BASIC program text.

However, IB is potentially one of the fastest and most powerful ways to program your 4000/8000 series PET/CBM computer. And PETSpeed and Integer Basic can be used together to form an astoundingly powerful program with the best of all worlds; a compiled program that calls a true machine language program for its subroutines and other time consuming work.

What is Source Code? Object Code?

When used in conjunction with PETSpeed/IB, source code refers to the BASIC text. The BASIC program is the source code.

Object code refers to the output from PETSpeed or IB. The compiled output program is the object code.

We hope you are finding our "PETSpeed Tips" column informative. Please send comments, questions, tips and programs to the address below. Your input is needed and we promise more programs and information to come. C

Joe Rotello
4734 E. 26th St.
Tucson, AZ 85711

Finding The Roots of a Cubic Equation With The VIC 20

by Ronald Suddendorf

Many problems in science and mathematics are defined by a polynomial. Although any degree of polynomial can be encountered, the majority of practical problems are defined by either a second order (quadratic) or a third order (cubic) polynomial. The solution of quadratics is relatively straightforward through the use of the well known quadratic equation. However, determining the roots of a cubic equation is slightly more complicated.

Several techniques are available to determine the roots of a cubic equation, including successive approximation, synthetic division and the use of exact equations. These techniques, although not difficult, are time consuming and subject to error when done by hand or on a pocket calculator. The purpose of this article is to provide a program for the VIC 20 capable of determining both the real and imaginary roots of a cubic equation. The technique used here to determine the roots is that of exact equations. To spare you the agony of looking at the mathematics used in this program (it is readily discernable from the program listing anyway), I will refer you to the reference at the end of this article, rather than include the math here.

The program is designed to provide the roots of a cubic equation having the general form:

$$Y^3 + PY^2 + QY + R = 0.$$

All you have to do is run the program and enter the value for coefficients P, Q and R when the computer prompts you to enter the values. Note that if the equation is of a different form such as:

$$Y^3 - PY^2 + R = 0$$

the value for P must be entered as a negative number and the value for Q must be entered as zero.

After the coefficients have been entered, the computer will calculate a value for U in line 36. The value for U will be either greater than zero, equal to zero or less than zero. If the value of U is greater than zero, then there are one real root and two conjugate imaginary roots. The calculation of these roots will start at line 40. If the value of U is equal to or less than zero, then there will be three real roots with the calculations starting at line 150. After the calculations are complete, the computer will display the coefficients you entered and identify the corresponding real and imaginary roots.

That's all there is to it! So the next time you are faced with finding the roots of a cubic equation, give this program a try. **C**

Reference:
Handbook of Chemistry and Physics, *48th edition, 1967-1968, page A245. The Chemical Rubber Co.: Cleveland.*

Roots of Cubic Equations

```
9 REM    ROOTS OF CUBIC EQUATIONS 10/16/83
10 REM          BY RONALD F. SUDDENDORF
11 PRINT CHR$(147)
12 PRINT"ENTER COEFFICIENTS"
13 PRINT"OF CUBIC EQUATION"
14 PRINT"Y↑3+P*Y↑2+Q*Y+R=0"
15 PRINT:INPUT"P=";P
20 INPUT"Q=";Q
25 INPUT"R=";R
30 A=(3*Q-(P)↑2)/3
35 B=(2*(P)↑3-9*P*Q+27*R)/27
36 U=((B)↑2)/4+((A)↑3)/27
38 IF U<0 THEN 150
39 IF U=0 THEN 150
40 B1=(-B/2)-U↑.5
45 IF B1<0 THEN 55
50 GOTO 70
55 B1=-B1
```

```
60 B1=-(B1)↑(1/3)
65 GOTO 75
70 B1=(B1)↑(1/3)
75 A1=(-B/2)+U↑.5
80 IF A1<0 THEN 90
82 A1=(A1)↑(1/3)
85 GOTO 108
90 A1=-A1
95 A1=-(A1)↑(1/3)
108 X=A1+B1
109 PRINT CHR$(147)
110 PRINT"COEFFICIENTS"
111 PRINT"P=";P
112 PRINT"Q=";Q
113 PRINT"R=";R
114 PRINT:PRINT"REAL ROOT"
115 PRINT"Y1=";X-(P/3)
120 PRINT:PRINT"IMAGINARY ROOTS"
125 PRINT"Y2=";-((A1+B1)/2)-(P/3)"+"((A1-B1)/2)*3↑.5"I"
130 PRINT:PRINT"Y3=";-((A1+B1)/2)-(P/3)"-"((A1-B1)/2)*3↑.5"I"
135 PRINT:PRINT"MORE DATA? Y=YES"
136 GET A$
137 IF A$="Y" THEN 11
138 IF A$="" THEN 136
139 END
150 W=(-B/2)/((-(A)↑3)/27)↑.5
160 D=-ATN(W/(-W*W+1)↑.5)+π/2
170 X1=(2*(-A/3)↑.5)*COS(D/3)
175 X2=(2*(-A/3)↑.5)*COS(D/3+(120/180)*π)
180 X3=(2*(-A/3)↑.5)*COS(D/3+(240/180)*π)
181 PRINT CHR$(147)
182 PRINT"COEFFICIENTS"
183 PRINT:PRINT"P=";P
184 PRINT"Q=";Q
185 PRINT"R=";R
190 PRINT:PRINT"REAL ROOTS"
195 PRINT:PRINT"Y1=";X1-(P/3)
200 PRINT"Y2=";X2-(P/3)
205 PRINT"Y3=";X3-(P/3)
210 PRINT:PRINT"MORE DATA? Y=YES"
211 GET A$
212 IF A$="Y" THEN 11
213 IF A$="" THEN 211
214 END
```

House Inventory Printer

by Robert W. Baker

Since the appearance of my "House Inventory" program in Volume 4, Number 4, Issue 25 of *Commodore* magazine, many people have written asking for an additional feature for printing the data. The desired feature is really quite simple to add and I've included the required changes and additions in the accompanying program listing excerpt. The original program appears on page 96.

As noted in the comments, replace the original program lines 120, 150, 160 and 170 with the new lines provided. Then add the new lines 125, 385, and 2000-2220. Do not enter the comment lines 10 through 26; they were added only for documentation purposes.

Lines 120-125 add the new print mode to the main menu as item five and change the former help selection from item five to item six. Lines 150-170 make the required changes to verify the mode selection, open the disk command channel and branch to the start of the proper function. The new line 385 is the entry to the actual print function after the proper file has been opened.

Lines 2000-2220 supply the new ability to print the inventory data with seven items per page. The disk data is read by the subroutine call in line 2030 and printed by lines 2040-2100.

Line 2110 provides paging of the output while lines 2120-2170 allow pausing, resuming and terminating printing. The small subroutine at lines 2200-2220 prints a heading line complete with the file name and page number at the start of each page.

As the data is printed, a reminder will be displayed that you can pause the printing at any time by hitting the space bar on the keyboard. After the space bar is pressed, printing will be stopped at the end of the current inventory item. At that time another message will be displayed on the screen reminding you that printing has been paused. You can then press "C" to continue or "Q" to quit the printing function. Whenever you press "Q" to quit, a short message will be printed at the end of the listing to indicate the printout was terminated prior to the end of the file.

Just a few short comments. The original program was written for the PET and CBM models but would run on the Commodore 64 as well, thus the request for the disk drive number in lines 190-210 and 320-340. Since the Commodore 64 1541 disk drive has only one drive, you could eliminate these lines. The best method would be to delete lines 190-210 and 320-340 and then change the T$+":" in lines 260 and 370 to simply "0:" to force all operations to drive zero.

Also, because you have only one drive, you must use a different file name for the output file when editing an existing inventory file. You can later delete the old file and rename the newly created file after you are done with your editing. With two drives on a PET/CBM you could edit an inventory file on drive zero and create a new version on drive one with the same name. With a single drive there's no way to do it. **C**

House Inventory Printer

```
10 REM **********************************
11 REM
12 REM ADDITIONS/MODIFICATIONS FOR
13 REM HOUSE INVENTORY PROGRAM
14 REM TO PROVIDE PRINTED OUTPUT OF
15 REM INVENTORY DATA STORED ON DISK.
16 REM
17 REM LINES 120,150,160 & 170
```

```
18 REM REPLACE ORIGINAL LINES.
19 REM
20 REM LINES 125.385 & 2000-2220
21 REM ARE NEW LINES TO BE ADDED.
22 REM
23 REM DO NOT ADD THESE COMMENTS!!
24 REM
25 REM *********************************
26 :
120 PRINT"[SPACE]4[SPACE]=[SPACE]EDIT[SPACE]DATA[SPACE]
    FILE":PRINT"[SPACE]5[SPACE]=[SPACE]PRINT[SPACE]DATA
    [SPACE]FILE"
125 PRINT"[SPACE]6[SPACE]=[SPACE]HELP[SPACE](INFORMATION)"
150 R=VAL(R$):IF R<1 OR R>6 THEN 140
160 IF Z<6 THEN OPEN 15,8,15
170 Z=R:ON R GOTO 310,180,310,310,310,1400
385 IF Z=5 THEN 2000
1900 REM *** NEW PROGRAM LINES FOR PRINTING DATA
2000 OPEN 4,4:P=1:GOSUB 2200
2010 PRINT"[CLEAR]PRINTING[SPACE]DATA..."
2020 PRINT"[DOWN]PRESS[SPACE]'SPACE'[SPACE]TO[SPACE]PAUSE
     [SPACE]PRINTING":PRINT"AT[SPACE]THE[SPACE]END[SPACE]
     OF[SPACE]ANY[SPACE]ITEM"
2030 GOSUB 1160:IF C>1 THEN 2180
2040 PRINT#4,"[SPACE6]ITEM:[SPACE]";I$
2050 PRINT#4,"[SPACE6]MAKE:[SPACE]";W$
2060 PRINT#4,"[SPACE5]MODEL:[SPACE]";M$
2070 PRINT#4,"SERIAL#/ID:[SPACE]";S$
2080 PRINT#4,"DATE[SPACE]ACQ'D:[SPACE]";D$
2090 PRINT#4,"VALUE:[SPACE6]";V$
2100 PRINT#4:PRINT#4
2110 L=L+1:IF L=7 THEN GOSUB 2200
2120 GET R$:IF R$<>"[SPACE]"THEN 2030
2130 PRINT"[CLEAR]PRINTING[SPACE]PAUSED..."
2140 PRINT"[DOWN]PRESS[SPACE]'C'[SPACE]TO[SPACE]CONTINUE
     [SPACE]OR[SPACE]'Q'[SPACE]TO[SPACE]QUIT"
2150 GET R$:IF R$="C"THEN 2030
2160 IF R$<>"Q"THEN 2150
2170 PRINT#4:PRINT#4,"*****[SPACE]LISTING[SPACE]
     TERMINATED"
2180 PRINT#4,CHR$(12):CLOSE 4:GOTO 550
2200 PRINT#4,CHR$(12);"INVENTORY[SPACE]LISTING[SPACE]OF
     [SPACE]FILE:[SPACE5]";F$;
2210 PRINT#4,"[SPACE5]*****[SPACE5]PAGE:[SPACE]";P
2220 PRINT#4:PRINT#4:L=0:P=P+1:RETURN
```

House Inventory: Original Program

```
10 REM ** HOUSE INVENTORY *** DISK **
20 REM
30 REM          ROBERT W. BAKER
40 REM 15 WINDSOR DR., ATCO, NJ 08004
50 REM
60 REM ******************************
70 :
80 PRINT"[CLEAR,SPACE5]HOUSEHOLD INVENTORY PROGRAM": GOSUB 1290
90 PRINT"DESIRED PROGRAM MODE:": PRINT: PRINT" 0 = DONE"
100 PRINT" 1 = READ DATA"
110 PRINT" 2 = WRITE NEW DATA FILE": PRINT" 3 = COPY DATA FILE"
120 PRINT" 4 = EDIT DATA FILE": PRINT" 5 = HELP (INFORMATION)"
130 GOSUB 1290: PRINT: PRINT"MODE ?";
140 GOSUB 1360: IF R$="0" THEN PRINT"[CLEAR]": END
150 R=VAL(R$): IF R<1 OR R>5 THEN 140
160 IF Z<5 THEN OPEN 15,8,15
170 Z=R: ON R GOTO 310,180,310,310,1400
180 GOSUB 1250
190 INPUT"[DOWN]OUTPUT TO DISK DRIVE# (0 OR 1)   0[LEFT3]";T$:
    T$=LEFT$(T$,1)
210 T=VAL(T$): IF T$<>"0" AND T$<>"1" THEN 80
220 PRINT: PRINT"OUTPUT ";: GOSUB 1340
230 IF F$<>"-" THEN 260
240 F$="INVENTORY DATA"
250 PRINT: PRINT"DEFAULT FILE = ";T$;":";F$
260 OPEN 2,8,5,T$+":"+F$+",S,W": GOSUB 1600
270 IF Z=3 THEN 560
280 IF Z=4 THEN 610
290 GOSUB 900: IF C THEN GOSUB 1130: GOTO 290
300 GOTO 550
310 GOSUB 1250
320 INPUT"[DOWN]INPUT FROM DISK DRIVE# (0 OR 1)   0[LEFT3]";T$
330 T=VAL(T$): T$=LEFT$(T$,1)
340 IF T$<>"0" AND T$<>"1" THEN 80
350 PRINT: PRINT"INPUT ";: GOSUB 1340
360 IF F$="-" THEN F$="INVENTORY DATA": PRINT"[DOWN]DEFAULT FILE =
    ";T$;":";F$
370 OPEN 1,8,6,T$+":"+F$+",S,R": GOSUB 1600
380 X$=""
390 IF Z>2 THEN 190
400 GOSUB 1160: IF C>1 THEN 490
410 GOSUB 1090: IF C THEN 510
420 GOSUB 1160: IF C>1 THEN 510
```

```
430 GOSUB 1100: IF C THEN 510
440 GOSUB 1160: IF C>1 THEN 510
450 GOSUB 1100: IF C THEN 510
460 GOSUB 1300
470 GOSUB 1380: IF R$<>"D" THEN 400
480 GOTO 550
490 PRINT"[CLEAR,RVS]END OF MODE #1[RVOFF,SPACE2]DONE READING
    DATA FILE" : PRINT
510 GOSUB 1300
520 IF C=1 THEN PRINT"END OF DATA FILE!"
530 IF C>1 THEN PRINT"DISK ERROR ( STATUS =";ST;")"
540 GOSUB 1350
550 CLOSE 1: CLOSE 2: CLOSE 15: GOTO 80
560 I9$="": GOSUB 1250: PRINT"[RVS]PLEASE WAIT[RVOFF,SPACE2]
   ******  COPYING DATA FILE![DOWN]"
570 GOSUB 1160:IF C>1 THEN 820
580 IF Z=4 AND LEFT$(I$,LEN(I9$))=I9$ THEN GOSUB 1250: GOTO 620
590 GOSUB 1130: IF C=1 THEN 820
600 IF Z=3 OR I9$<>"" THEN 570
610 GOSUB 1160: IF C>1 THEN 820
620 GOSUB 1250: GOSUB 1100: GOSUB 1290: PRINT"DESIRED ACTION:":PRINT
630 PRINT"  1 = COPY THIS ITEM, NO CHANGE"
640 PRINT"  2 = DELETE THIS ITEM"
650 PRINT"  3 = INSERT ITEMS BEFORE THIS ONE"
660 PRINT"  4 = SEARCH & COPY TILL ITEM FOUND": PRINT
670 PRINT"ACTION ? ";
680 GOSUB 1360: R=VAL(R$): IF R<1 OR R>4 THEN 680
690 PRINT R$
700 PRINT"OK": I9$="": ON R GOTO 590,710,730,760
710 IF C=1 THEN 820
720 GOTO 610
730 I9$=I$: W9$=W$: M9$=M$: S9$=S$: D9$=D$: V9$=V$: C9=C
740 GOSUB 900: IF C THEN GOSUB 1130: GOTO 740
750 I$=I9$: W$=W9$: M$=M9$: S$=S9$: D$=D9$: V$=V9$: C=C9: GOTO 620
760 GOSUB 1250: PRINT"ALL ENTRIES WILL BE COPIED UNTILL"
770 PRINT: PRINT"DESIRED ITEM IS FOUND;"
780 PRINT: PRINT: PRINT"ENTER ITEM TO SEARCH FOR:"
790 INPUT"    -[LEFT3]";I9$
800 IF I9$="-" THEN I9$="": PRINT"[DOWN3]SEARCH ABORTED": GOTO 620
810 PRINT: PRINT: PRINT: PRINT"SEARCHING": GOTO 580
820 IF Z=3 THEN 520
830 GOSUB 1250: IF C>1 THEN 530
840 PRINT"END OF INPUT FILE!"
```

```
850 PRINT: PRINT"DO YOU WANT TO ADD ANY ENTRIES TO THE"
860 PRINT: PRINT"END OF THE DATA FILE";
870 GOSUB 1310: IF R$="N" THEN 550
880 GOSUB 900: IF C THEN GOSUB 1130: GOTO 880
890 GOTO 550
900 C=0: PRINT"[CLEAR]ENTER ITEM INFORMATION:[DOWN]"
  : PRINT"D = DONE ENTERING DATA"
910 PRINT"E = ERROR, RESTART ENTIRE ITEM"
920 PRINT: PRINT"DO NOT USE ',' OR ':' WITHIN THE DATA"
930 PRINT: PRINT"PRESS [RVS]RETURN[RVOFF,SPACE]AFTER EACH ENTRY"
940 GOSUB 1290: INPUT"[RVS]ITEM[RVOFF,SPACE3]?[LEFT3]";I$: IF I$="E"
    THEN 900
950 IF I$="D" THEN RETURN
960 INPUT"[RVS]MAKE[RVOFF,SPACE3]?[LEFT3]";W$: IF W$="E" THEN 900
970 IF W$="D" THEN RETURN
980 INPUT"[RVS]MODEL[RVOFF,SPACE3]?[LEFT3]";M$: IF M$="E" THEN 900
990 IF M$="D" THEN RETURN
1000 INPUT"[RVS]SERIAL#/ID[RVOFF,SPACE3]?[LEFT3]";S$: IF S$="E"
     THEN 900
1010 IF S$="D" THEN RETURN
1020 INPUT"[RVS]DATE ACQ'D[RVOFF,SPACE](MONTH/DAY/YEAR)
     ?[LEFT3]";D$ : IF D$="E" THEN 900
1030 D$=LEFT$(D$,8): IF D$="D" THEN RETURN
1040 INPUT"[RVS]$VALUE[RVOFF,SPACE3]?[LEFT3]";V$:IF V$="E" THEN 900
1050 IF V$="D" THEN RETURN
1060 GOSUB 1090: GOSUB 1290
1070 PRINT"IS THIS ENTRY CORRECT";: GOSUB 1310: IF R$="N" THEN 900
1080 C=1: RETURN
1090 PRINT"[CLEAR]";
1100 PRINT"[RVS]ITEM:[RVOFF,SPACE]";I$:PRINT"[RVS]MAKE:[RVOFF,SPACE]
     ";W$ : PRINT"[RVS]MODEL:[RVOFF,SPACE]";M$
1110 PRINT"[RVS]SERIAL#/ID:[RVOFF,SPACE]";S$
1120 PRINT"[RVS]DATE ACQ'D:[RVOFF,SPACE]"D$;TAB(22);"[RVS]VALUE
   :[RVOFF,SPACE]$";V$: PRINT: RETURN
1130 X$=I$: GOSUB 1150: X$=W$: GOSUB 1150: X$=M$: GOSUB 1150
1140 X$=S$: GOSUB 1150: X$=D$: GOSUB 1150: X$=V$
1150 PRINT#2,X$;CHR$(13);: GOTO 1600
1160 GOSUB 1230: I$=X$: IF C THEN RETURN
1170 GOSUB 1230: W$=X$: IF C THEN RETURN
1180 GOSUB 1230: M$=X$: IF C THEN RETURN
1190 GOSUB 1230: S$=X$: IF C THEN RETURN
1200 GOSUB 1230: D$=X$: IF C THEN RETURN
1210 GOSUB 1230: V$=X$: IF C=2 THEN C=1
```

```
1220 RETURN
1230 C=0: INPUT#1,X$: IF ST THEN C=3: IF ST=64 THEN C=2
1240 GOTO 1600
1250 IF Z=1 THENPRINT"[CLEAR,RVS]MODE #1[RVOFF,SPACE2]READ DATA FILE"
1260 IF Z=2 THENPRINT"[CLEAR,RVS]MODE #2[RVOFF,SPACE2]WRITE NEW DATA
     FILE"
1270 IF Z=3 THENPRINT"[CLEAR,RVS]MODE #3[RVOFF,SPACE2]COPY DATA FILE"
1280 IF Z=4 THENPRINT"[CLEAR,RVS]MODE #4[RVOFF,SPACE2]EDIT DATA FILE"
1290 PRINT
1300 PRINT"---------------------------------------": PRINT: RETURN
1310 PRINT" (Y/N) ? ";
1320 GOSUB 1360: IF R$<>"Y" AND R$<>"N" THEN 1320
1330 PRINT R$: RETURN
1340 INPUT"FILENAME    -[LEFT3]";F$: RETURN
1350 PRINT: PRINT"HIT ANY KEY WHEN READY TO CONTINUE";: GOTO 1390
1360 GET R$: IF R$="" THEN 1360
1370 RETURN
1380 PRINT: PRINT"HIT ANY KEY TO CONTINUE, D=DONE";
1390 GOSUB 1360: PRINT: PRINT"OK": RETURN
1400 PRINT"[CLEAR]THIS PROGRAM WAS DESIGNED TO WRITE,"
1410 PRINT"READ, COPY, OR EDIT DISK DATA FILES"
1420 PRINT"CONTAINING INFORMATION ON YOUR"
1430 PRINT"HOUSEHOLD POSSESSIONS. THIS INFORMATION"
1440 PRINT"INCLUDES AN ITEM DESCRIPTION ALONG WITH"
1450 PRINT"THE MAKE, MODEL, SERIAL NUMBER (OR"
1460 PRINT"OTHER IDENTIFYING MARKS), DATE ACQUIRED"
1470 PRINT"AND THE VALUE. THIS DATA SHOULD BE OF"
1480 PRINT"GREAT VALUE FOR INSURANCE RECORDS"
1490 PRINT"IN CASE OF FIRE OR THEFT; AND MAY EVEN"
1500 PRINT"BE OF SOME USE FOR TAX RECORDS."
1510 PRINT: PRINT"DISK FILE HANDLING HAS BEEN INCLUDED TO"
1520 PRINT"ALLOW USING SEPERATE FILES FOR EACH"
1530 PRINT"ROOM, SPECIAL COLLECTIONS, ETC."
1540 PRINT"THIS PROVIDES EASY DATA MAINTENANCE"
1550 PRINT"WHILE ALL DATA CAN EASILY BE STORED ON"
1560 PRINT"A SINGLE DISKETTE."
1570 PRINT: PRINT"WHY NOT KEEP A COPY IN YOUR BANK"
1580 PRINT"SAFETY DEPOSIT BOX FOR SAFE KEEPING?"
1590 GOSUB 1350: GOTO 80
1600 INPUT#15,EN,EM$,ET,ES: IF EN=0 THEN RETURN
1610 PRINT"[CLEAR,RVS]DISK ERROR[RVOFF]": PRINT
1620 PRINT EN,EM$;ET;ES
1630 GOSUB 1290: GOTO 540
```

8032 Windows

by Joe Rotello

One of the most useful features of the 8000/9000 series CBM computers yet perhaps the least used is the ability to create windows on the video screen.

When you create a window (or data window as it is sometimes termed) on the CBM you section off an area of the video screen and in that area allow video action and movement as if the entire screen were in use. The area not windowed remains static and does not move or respond to the standard CBM set of video commands (clear screen, home cursor, etc.). The window area becomes a screen unto itself and it is that area alone that responds to the various screen commands.

This window ability enables both long-time programmers and new users to build video displays that are active, useful and can make use of the computer more efficient and understandable.

Window Pains

There are quite a few ways to enable the window function in the CBM series. We will focus on three window modes in this article.

1. POKEd windows that default to the full screen width. That is, regardless of the window depth, the width of the window remains at 80 characters.
2. Windows created by character strings. In this method the user moves the cursor to the appropriate location, issues a CHR$(15) print command to define the upper left hand window location, lowers the cursor down a number of lines, sends the cursor to the bottom right window location and issues a CHR$(143) command to set the window end point.
3. Windows formed by printing the proper control key symbols at the appropriate points. This method is not suggested if you are a new user but can be used by more advanced programmers.

If you can use a window where the full 80-column screen width can be tolerated, we feel that the POKEd window arrangement is the best method. This window can be set up very fast with a minimum of code and the POKEs also allow the user to "hop" from one window to another a little more quickly through a simple one line subroutine.

When you have to create a window that sections off only a certain selected portion of the video screen, the CHR$ method of window selection would be the best alternative. However, the CHR$ method takes a little more thought, setup time and somewhat more memory and might be confusing especially where multiple windows are required.

Ground Rules

Let's set the ground rules that the system requires:

1. Only one window may be in use at any one time.
2. Multiple windows can be defined by coding in the various locations and "hopping" back and forth between those locations to give the effect that more than one window is in operation at the same time.
3. It is in the best interests of efficiency to put all window coding, either to create or change windows, in subroutines that may be called at any time from the main program.
4. Although the system POKE locations for the POKE window may be changed in a future version or model of CBM, the POKEs still remain as a choice window creation method given the overall simplicity of the coding.

POKE Window Method

The POKE method of creating data windows is rather simple. Program 1 is an example that creates actually two windows, one covering between lines 2 and 20 on the screen and a second smaller window covering between lines 22 and 25 on the screen.

We detailed Program 1 profusely so the operation of the program is clear. In real use, delete all the REMs and colon remarks and combine more than one instruction on each line. The program will reduce to a very few lines.

The POKE method directly modifies the screen line address bytes in the BASIC operating system. By POKing the required information to these two locations, the system is really fooled into thinking that a window has been set up.

When POKing, remember that the window will start *one line greater* than the POKE value and end *one line greater* than the POKE value. For example, POKE 224,1 (the starting location) followed by POKE 225,19 (the ending location) will define a window that starts on line 2 and ends on line 20 of the screen.

The window is created and the cursor is now part of the created window immediately after POKing. Therefore, all cursor action, for example clearing the screen, dropping the cursor lower or TABbing right, will now take place in the newly created window area and nowhere else.

As we stated before, Program 1 creates two windows, each existing only when called for. This brings up an important and logical use for windows. If you define two windows of the approximate sizes in Program 1, you can use the smaller window (or bottom window in this case) for instructions or perhaps a short, concise menu. Use the larger window (or top window in this case) to express the active material in the program—perhaps a disk directory, an accounts payable record, a mailing label set or what have you.

The bottom line here is that by using a window you can have a program that makes more sense to a user, even if you will be the only one using it. It will certainly be more efficient and error-proof should another less computer-knowledgeable person have to use it.

As you might have noticed, Program 1 also shows how easy it is to hop back and forth between windows. Again, only one window may exist or be in use at any given time, but we give the effect of multiple windows by being able to easily go between larger and smaller windows as required.

Remember that windows using the POKE creation method will always be 80 characters wide or, in any event, the width of the 80-column video screen.

As a final note, there may be CBM users who do not yet have a disk drive. If so, replace line 370 in Program 1 with a simple FOR... NEXT loop—for example: 400 FOR I = 1 to 100:PRINT I,I*I: NEXT. This will suffice as an alternative to printing the disk directory in the window to show activity.

Window With CHR$

As we stated earlier, windows created via CHR$ calls are a little harder to implement at times. However, when the window has to be located in a corner of the video screen or where the window dimensions otherwise preclude using the POKE method, carry on with CHR$.

Program 2 details this method. Note that the window takes up only a fraction of the entire video screen area. The starting point of the window, the upper left hand corner, is defined by executing the CHR$(15) statement. The cursor is then moved downward to form the window body or depth. Next a TAB statement is executed to form the window width dimension. Finally the ending point for the window, the lower right hand corner, is defined by executing the CHR$(143) statement. Be aware that the semicolon following many of the cursor movements is *very* important.

Note how the window is defined in line 130 and that the preceding REM statement lists the window parameters. When defining a window using the CHR$ method, it's probably a good idea to define the dimensions of your window in a REM statement. Then you can "build" the window and refer to the REM statement for your list of specifications as you go along. Not needed all the time, but a good practice for those who are "windowing" for the first few times.

When defining multiple windows using the CHR$ method, it is best to disable the previous window by executing a "double home" cursor movement *before* executing the next window. Putting a window inside of a window can look pretty bad if you do not delete the previous screen window. Sort of like putting two mirrors face to face and getting a confused image. Try it, you'll see what we mean!

Window With Control Characters

In a manner closely akin to defining a window via CHR$, windows can be defined using control characters that are generated from the CBM keyboard. This method is really no more complex than any other, but is not used very much in general programming. In this method the control characters are part of a defined PRINT statement and are generated within the PRINT statement quotes.

The window's top left corner is generated by pressing the ESCape key, releasing it, pressing the RVS key, releasing it, then pressing the unshifted (lower case) letter "o". Likewise the window bottom right corner is generated using the same sequences as above, but using the shifted (upper case) "O". Between the two control characters, you place the required cursor movements to define the actual window dimensions, just as in the CHR$ method.

When properly executed, both the lower case and upper case symbols will appear in the print statement as reversed or "lit up" letters. Be sure to close quotes when the proper symbol has been entered.

Escape From the Window

Outside of having lightning strike the computer, nothing irritates a programmer or user more than being stuck in a window. Fortunately, the exit from a window is quite simple. From within your program, execute the "double home" statement to disable the window. Electronically speaking, the computer sees the "double home" call and suddenly contracts amnesia. The window is exited and the cursor is returned to the standard screen upper left hand corner, as if nothing had ever happened.

If you do not happen to like cursor movements, you can accomplish the same feat by simply saying PRINT CHR$(19)CHR$(19) and the window will be disabled.

The CBM computers have a host of other text features above and beyond the window capability. For example, text may be scrolled up and down from within a window, text may be erased either on a complete line basis or from a certain point onward, etc. We will cover these functions in a later article, so be on the lookout! **C**

Window 1

```
10 REM      PROGRAM #1
20 :
30 REM      WINDOW DEMO FOR 8032/-96/9000
40 REM      WITHOUT CHR$ CALLS
50 :
60 REM      CLR SCREEN, BEGIN PRG
70 PRINT"[CLEAR]":GOTO 170
80 :
90 REM    SUBROUTINES TO SET DATA WINDOWS
100 :
110 REM     UPPER DATA WINDOW - START ROW 2 & END ROW 20
120 POKE 224,1:POKE 225,19:RETURN
130 :
140 REM     LOWER DATA WINDOW - START ROW 22 & END ROW 25
150 POKE 224,21:POKE 225,24:RETURN
160 :
170 REM     REVERT SCREEN TO LOWER WINDOW
```

```
180 GOSUB 150
190 :
200 REM    CLEAR LOWER WINDOW FOR NEW DISPLAY
210 PRINT"[CLEAR]";
220 :
230 REM    DISPLAY LOWER WINDOW MESSAGE
240 PRINT TAB(25)"DISK[SPACE]DIRECTORY[SPACE]IN[SPACE]
    UPPER[SPACE]WINDOW[SPACE2]"
250 PRINT TAB(27)"PRESS[SPACE,RVS]SPACE[RVOFF,SPACE]KEY
    [SPACE]TO[SPACE]CONTINUE"
260 :
270 REM   WAIT FOR USER TO PRESS SPACE BAR
280 GET A$:IF A$<>"[SPACE]"THEN 280
290 :
300 REM    REVERT SCREEN TO UPPER WINDOW
310 GOSUB 120
320 :
330 REM    CLEAR UPPER WINDOW FOR NEW DISPLAY
340 PRINT"[CLEAR]";
350 :
360 REM    DISPLAY DISK DIRECTORY IN UPPER WINDOW
370 DIRECTORY D0
380 :
390 REM    WAIT FOR USER TO PRESS SPACE BAR AGAIN
400 GET A$:IF A$<>"[SPACE]"THEN 400
410 :
420 REM    EXIT FROM WINDOW WITH DOUBLE HOME
430 PRINT"[HOME2]"
440 :
450 REM   CLR SCREEN, PRINT TERMINATION MESSAGE, END PRG
460 PRINT"[CLEAR]":PRINT"WINDOW[SPACE]DEMO[SPACE]COMPLETE,
    [SPACE]WINDOW[SPACE]DISABLED":END
```

Window 2

```
10 REM    PROGRAM #2
20 :
30 REM    WINDOW DEMO WITH CHR$
40 :
50 REM    WRITE SCREEN FILL, 39 CHRS WIDE
60 A$="#######################################"
```

```
70 PRINT"[CLEAR]";:FOR I=1 TO 15:PRINT A$:NEXT
80 :
90 REM    MORE SCREEN FILL, 78 CHRS WIDE
100 A$=A$+A$:FOR I=1 TO 9:PRINT A$:NEXT
110 :
120 REM   FORM WINDOW - START COLUMN 42, 14 ROWS DEEP,
     END COLUMN 79
130 PRINT"[HOME]";:PRINT TAB(42);CHR$(15);"[DOWN14]";
    TAB(79);CHR$(143)
140 :
150 REM  PERFORM SONG AND DANCE IN WINDOW
160 PRINT"[CLEAR]DATA[SPACE]WINDOW:":FOR Y=1 TO 2000:NEXT
170 FOR N=1 TO 50:PRINT"[RVS]"N"[RVOFF]",INT(N^2),INT(N^3)
    :NEXT N
180 :
190 REM  SHOW OVER, WAIT FOR USER TO HIT SPACE KEY
200 PRINT"[DOWN2]HIT[SPACE]SPACE[SPACE]KEY[SPACE]TO[SPACE]
    EXIT"
210 GET A$:IF A$<>"[SPACE]"THEN 210
220 :
230 REM  DISABLE WINDOW
240 PRINT"[HOME2]"
250 :
260 REM  CLR SCREEN, PRINT MESSAGE, END PRG
270 PRINT"[CLEAR]":PRINT"WINDOW[SPACE]DEMO[SPACE]COMPLETE,
    [SPACE]WINDOW[SPACE]DISABLED":END
```

SuperPET Potpourri

by Dick Barnes, Editor, SuperPET Gazette

At last SuperPETters can come out of the closet! We're glad to welcome Dick Barnes as a regular contributor to Commodore's magazines.

The best place to start a new column is at the beginning, with a new SuperPET, just out of the box. Yet that might bore the experts, so I'll break this column into two parts, the first for beginners, and the second for more advanced users.

Beginners

This issue, we'll cover talking to disk drives and printers from all SuperPET languages except for APL (which is another kettle of fish). I'll distinguish the SuperPET languages with a prefix of "m", for "micro", as in "mBASIC", so we won't confuse SuperPET's languages with BASIC 4.0 and other versions of PASCAL, FORTRAN, COBOL, and APL.

We're lucky that all languages except mAPL use the Waterloo microEDITOR, (or mED), since we can issue one set of commands to disks or printers from the mED, whatever the language. That makes learning the commands easy.

SuperPET thinks all devices (printers, the serial port, disk drives, keyboard, and the screen) are files, so you talk to them all in the same basic way. But you must know how to structure a filename to do so:

1. Disk Drives: The examples below show clearly how to handle two files, "sample", and "blank sample". Commands default to device eight and to drive zero. Enter commands at the command cursor in mED.

Command in microEDITOR	What it Does:
put sample	Files "sample" on drive 0, device 8
put disk/1.sample	Files "sample" on drive 1, device 8
put disk9/1.sample	Files "sample" on drive 1, device 9
get disk 9/0.sample	Gets "sample" off drive 0, device 9, and brings it into memory.
g 'disk/1.blank sample'	Gets "blank sample" on drive 1, device 8. Apostrophes or quotation
("p" is an abbreviation for "put"; "g" for "get")	marks must be used as shown if there's a space in the file-designator, or title.

2. Printers: Two ports or busses has SuperPET: user/IEEE-488 and serial. The serial port is a file. The IEEE-488 is not (it handles many devices, including printers and disk drives). So how you talk to the printer depends on where it is connected, on the type of printer and on its device number. In the examples below I assume the printer is device number four.

Port/Bus	Printer	Printer Filename
User/IEEE-488	Any Commodore using PET ASCII	printer
Serial	Any ASCII printer	serial
User/IEEE-488	Any ASCII printer, including Commodore 8300P	ieee4 (for device 4)

I printed the manuscript for this column with a Commodore 8300P, and sent this text to the printer with: "p ieee4", which "puts" to the printer a "file" of whatever is in mED's memory. Do you wonder how I paged it? The microEDITOR allows you to print any line or lines by stating the line number(s), as in: "1,54 p ieee4", which sends to the printer lines one through 54 inclusive. (You didn't know SuperPET has a built-in text-processor, which will print *exactly* what you see on the screen? Well, it does! As you'd expect, you can handle disk files much the same way.)

You have two more ways to talk to disks/printers:

3. Disk Directories (Version 1.1 Software)

Command	Effect
di disk ieee4	Puts directory of disk/0 to ASCII printer on user/IEEE-488
di disk/1 printer	Puts directory of disk/1 to Commodore printer on user/IEEE-488.
di disk9/0 serial	Puts directory of disk9/0 to ASCII printer on serial port.
di disk index	Puts directory of disk/0 on that disk as "index"

Command	Effect
di disk/1 disk/1.index	Puts directory of disk/1 on that disk as "index"
di disk9/1 index	Puts directory of disk9/1 on disk8/0 as "index"

Of course, if you want to comment a directory or to print only part of it, you send it to disk and then "get" it back into the microEDITOR, where you can comment and print it as you wish.

4. Talking to Disk Drives from Menu: When you first power-up SuperPET, the system ROMs generate a menu on screen. The system loader expects you to select the first letter of the language you want ("f" for mFORTRAN, "b" for mBASIC), but mCOBOL was issued after the ROMs were made, so enter COBOL in capitals, and then hit <RETURN>. The loader assumes the language disk is in drive one, but you can load from any other drive (or disk device) if you preface your command in this form: disk/0.b <RETURN>. (This command will load microBASIC from drive zero). If you have a second set of drives as device nine, you can load any language/facility this way: disk9/1.COBOL <RETURN>.

You can load machine language programs of your own from menu. I have one that "profiles" my printer; another that retrieves any language and its programs if I accidentally leave it (it happens!); another that resets back to the 6809 language last used after I've been in 6502; one that gives me three different tab settings; and another that dumps any screen to disk at the touch of a key, several screen dumps to printer, and so on. Some machine! (To obtain more information, write: Editor, *SuperPET Gazette*, PO Box 411, Hatteras, NC, 27943.)

For More Advanced Users: That Beautiful Macro

In assembler, it's easy to make mistakes when you send parameters (parms) to the stack, and then reset the stack pointer when you're through. John Toebes, of Raleigh, North Carolina, wrote a SuperPET macro which calls system routines, stuffs the proper data in the D register or on the stack, and then cleans up the stack. Though some SuperPET system routines will take six parms, the example in Listing 1 handles only three, to make it clear. You can expand it to handle six. (Be not confused by the backslashes "\" in the macro. They mean that the number following is designated as a pseudo-variable. If it picks up a non-null parameter, it marks a real variable. \0 is the first parm you pass to the macro, \1 is the second, etc.)

Let's pass two parameters to system routine OPENF_, which opens a file for you. You must tell OPENF_ the file name and the file mode (read, write, etc.). Normally, you'd stuff the second parm on the stack and put the first parm into the D register. With Toebes' macro you simply make a call, like this:

```
; parm 0      parm 1      parm 2      (semicolon
shows a comment)
      CALL OPENF_,#FILENAME, #FILEMODE
```

The macro stuffs the address of FILEMODE on the stack, that of FILENAME in the D register and calls subroutine OPENF_ It then resets the stack pointer. The example is easy but—write the code by hand to handle six parms, with the sixth stuffed on the stack first, then the fifth, then the fourth....

Puzzled by the periods preceding and following the backslash in the macro? I'll let you figure it out, but give you a hint: the comma's a data separator; IFNC (IF Not Comparable) compares the parms in front of and behind the comma. (the \ is ignored). How do you send a very common parm, ",S", which means the current value of the stack pointer? The comma in ",S" raises heck—if the periods aren't in the macro. Second hint: when no parameter is passed, the macro assembler compares: ".,." (or *null period* with *null period*). And IFNC executes the code following it *only* if a parm is *not* null. **C**

Listing 1: CALL.MACRO

```
call     macr                  ;Call a system proc., parm 1, parm 2...parm n
pcount__ set  0                ;Set parameter count to zero
         ifnc  .,\1.           ;Test for 1st parm. Is it null?
            ifnc  .,\2.        ;Test for 2nd parm; if not null,
pcount__       set   2         ;increment parm count for 2 bytes.
               ifnc  .,\3.     ;Test for 3rd parm; if not null,
pcount__          set   4      ;increment parm count for 4 bytes.
                  ldd   \3     ;Load 3rd parameter,
                  pshs  d      ;and push it onto stack.
               endc
               ldd   \2        ;Load 2nd parm,
               pshs  d         ;and stack it.
            endc
            ldd   \1           ;Load 1st parm in D register.
         endc
         jsr   \0              ;and jump to the called subroutine.
         ifne     pcount__
            leas  pcount__,s   ;Then clear the stack if parm count > 0
         endc
         endm
```

1. Comments *must* be removed before this macro is used, or they form part of the IFNC comparison and cause a crash. You may enter all code in lower case.
2. Put this macro on disk, filenamed "call.macro". Annotate your assembler code at left margin (screen column 1) with: ;include <call.macro>. Leave no space between the semicolon and the word "include".
3. The phrase "call macr", line one of this listing, must begin at screen column one before you file to disk. Put the macro on the same disk as your assembler code. **C**

Reprise: Starting a User Group

by Rita McCann
Stockton, California

Starting a user group, as many of you know, can be a rewarding experience . . . ?

When my friend and I both purchased Commodore 64's within a month of each other early in 1983, we spent a great deal of time consulting on the telephone. We hauled our machines and peripherals back and forth as we worked out various problems and bugs. Then we started discussing the desirability of having a user group in Stockton. An article on forming a group appeared in *Commodore* magazine at about that time ("Starting a User Group", May, 1983). We decided that it didn't seem too difficult. As soon as summer was over, we would proceed.

The first thing we did was to meet, article in hand, to plan the first meeting. Because he spoke quicker than I did, I was "elected" to chair the first meeting. The idea of holding the first few meetings in members' homes seemed easy enough, so I volunteered to have the first. (Act early, and get it over with.) Bob volunteered to make the contacts with the local newspaper.

As the temporary chairperson, I plugged in my word processing program and developed a tentative agenda based on the outline in the article—name, officers, projects, purpose, dues, bylaws—BYLAWS? We had neglected bylaws! Since Bob and I are both educators and school had just started for the fall term, a real crisis appeared in the offing. Bob rose to the occasion immediately—"*You* can knock them out."

The announcement appeared in the newspaper on Monday. Before the paper had settled on my porch the phone began to ring. We began to suspect that the ten or fifteen people we had expected to show up might be a few more. As the phone continued to ring over the next two days, Bob and I began to feel the first stirrings of panic. I started to remove the excess furniture from my family room. Bob borrowed a dozen folding chairs from his school. The couches and chairs were pushed back against the wall to make room for them. I borrowed a hundred-cup coffee pot from my school. We were ready!

On Wednesday, I rushed home from school to make the last few preparations. Bob arrived 45 minutes ahead of the starting time so we could set up his data management system for sign-ins and check over our preparations for the final time. *We were ready* (maybe). The first prospective User arrived a half hour before the scheduled time of 7:30. We were underway!

As the time wore on, more and more people crowded into my family room. The air conditioning apparently surrendered to the heat wave. The temperature climbed, the noise level rose and the enthusiasm level seemed to surpass both. Thirty-five people attended our first meeting—from children of eight or nine to retirees and brand new computer owners, to one who had been in programming for twenty years. Engineers, lawyers, teachers, programmers, students. It was fantastic and exciting! It was everything we had hoped.

It was also crowded, with people standing on the stairs. It was friendly, with everyone accepting the discomfort of the overcrowded room and the high temperature. We made a few comments on Commodore's suggestion about holding the meetings in homes, and sought an alternate site. It now looks like we will be meeting at the local university if permission can be obtained. The University of Pacific is using nothing but Commodore 64's in their BASIC workshop, so they have the equipment and certainly the room that we lack.

The phone has continued to ring. New names are being added to our database. People are amazed at our "courage" for undertaking the project. Was it worth it? YES! Would we recommend that others do the same? YES! Are we looking forward to the future meetings of the Stockton Commodore 64s? You bet! **C**

Rita McCann and Bob Eustis are listed on this month's User Bulletin Board under "User Groups Forming".

User Group Listing

ALABAMA

Huntsville PET Users Club
9002 Berclair Road
Huntsville, AL 35802
Contact: Hal Carey
Meetings: every 2nd
Thursday

Riverchase Commodore Users Group
617 Grove St.
Birmingham, AL 35209
(205) 988-1078
Ken Browning

Wiregrass Micro-Computer Society
Commodore SIG
109 Key Bend Rd.
Enterprise, AL 36330
(205) 347-7564
Bill Brown

Tiger Byte: E. Alabama CBM 64
Users Group
c/o The Computer Store, Inc.
Midway Plaza
Opelika, AL 36801
Jack Parsons
1st & 3rd Wed. of Month

ALASKA

COMPOOH-T
c/o Box 118
Old Harbor, AK 99643
(907) 286-2213

Alaska 84 Computer Club
c/o Line 49 Management
P.O. Box 6043
Anchorage, AK 99502

First City Users Group
P.O. Box 6692
Ketchikan, AK 99901
(907) 225-5695
James Llanos

ARIZONA

VIC Users Group
2612 E. Covina
Mesa, AZ 85203
Contact: Paul Muffuletto

Catalina Commodore Computer Club
2012 Avenida Guillermo
Tucson, AZ 85710
(602) 296-6766
George Pope
1st Tues. 7:30 p.m.

Central Arizona PET People
842 W. Calle del Norte
Chandler, AZ 85224
(602) 899-3622
Roy Schahrer

ACUG
c/o Home Computer Service
2028 W. Camelback Rd.
Phoenix, AZ 85015
(602) 249-1186
Dan Deacon
First Wed. of month

West Mesa VIC
2351 S. Standage
Mesa, AZ 85202
Kenneth S. Epstein

Arizona VIC 20-64 Users Club
232 W. 9th Place North
Mesa, AZ 85201
Donald Kipp

Arizona VIC & 64 Users
904 W. Marlboro Circle
Chandler, AZ 85224
(602) 963-6149
Tom Monson

ARKANSAS

Commodore/PET Users Club
Conway Middle School
Davis Street
Conway, AR 72032
Contact: Geneva Bowlin

Booneville 64 Club
c/o A. R. Hederich
Elementary School
401 W. 5th St.
Booneville, AR 72927
Mary Taff

The Siloam Commodore
Computer Club
P.O. Box 88
Siloam Springs, AR 72761
(501) 524-5624
Ken Emanualson

Russellville Commodore User Group
401 S. Arlington Dr.
Russellville, AR 72801
(501) 967-1868
Bob Brazeal

Arkansas River Valley
Commodore Users
401 S. Arlington Dr.
Russellville, AR 72801
(501) 967-1868
Bob Brazeal

CALIFORNIA

SCPUG Southern California
PET Users Group
c/o Data Equipment Supply
Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Meetings: First Tuesday of
each month

California VIC Users Group
c/o Data Equipment Supply
Corp.
8315 Firestone Blvd.
Downey, CA 90241
(213) 923-9361
Meetings: Second Tues. of
each month

Valley Computer Club
1913 Booth Road
Ceres, CA 95307

PUG of Silicon Valley
22355 Rancho Ventura Road
Cupertino, CA 95014

Lincoln Computer Club
750 E. Yosemite
Manteca, CA 95336
John Fung, Advisor

PET on the Air
525 Crestlake Drive
San Francisco, CA 94132
Max J. Babin, Secretary

PALS (Pets Around)
Livermore Society
886 South K
Livermore, CA 94550
(415) 449-1084
Every third Wednesday
7:30 p.m.
Contact: J. Johnson

SPHINX
7615 Leviston Ave.
El Cerrito, CA 94530
(415) 527-9286
Bill MacCracken

San Diego PUG
c/o D. Costarakis
3562 Union Street
(714) 235-7626
7 a.m.-4 p.m.

Walnut Creek PET
Users Club
1815 Ygnacio Valley
Road
Walnut Creek, CA 94596

Jurupa Wizards
8700 Galena St.
Riverside, CA 92509
781-1731
Walter J. Scott

The Commodore Connection
2301 Mission St.
Santa Cruz, CA 95060
(408) 425-8054
Bud Massey

San Fernando Valley
Commodore Users Group
21208 Nashville
Chatsworth, CA 91311
(213) 709-4736
Tom Lynch
2nd Wed. 7:30

VACUUM
277 E. 10th Ave.
Chico, CA 95926
(916) 891-8085
Mike Casella
2nd Monday of month

VIC 20 Users Group
2791 McBride Ln. #121
Santa Rosa, CA
(707) 575-9836
Tyson Verse

South Bay Commodore Users Group
1402 W. 218th St.
Torrance, CA 90501
Contact: Earl Evans

Slo VIC 20/64 Computer Club
1766 9th St.
Los Osos, CA

The Diamond Bar R.O.P. Users Club
2644 Amelgado
Haciendo Hgts., CA 91745
(213) 333-2645
Don McIntosh

Commodore Interest Association
c/o Computer Data
14660 La Paz Dr.
Victorville, CA 92392
Mark Finley

Fairfield VIC 20 Club
1336 McKinley St.
Fairfield, CA 94533
(707) 427-0143
Al Brewer
1st & 3rd Tues. at 7 p.m.

Computer Barn Computer Club
319 Main St.
Suite #2
Salinas, CA 93901
757-0788
S. Mark Vanderbilt

Humboldt Commodore Group
P.O. Box 570
Arcata, CA 95521
R. Turner

Napa Valley Commodore
Computer Club
c/o Liberty Computerware
2680 Jefferson St.
Napa, CA 94558
(707) 252-6281
Mick Winter
1st & 3rd Mon. of month

S.D. East County C-64 User Group
6353 Lake Apopka Place
San Diego, CA 92119
(619) 698-7814
Linda Schwartz

Commodore Users Group
4237 Pulmeria Ct.
Santa Maria, CA 93455
(805) 937-4174
Gilbert Vela

Bay Area Home Computer Asso.
Walnut Creek Group
1406 N. Broadway at Cypress
Walnut Creek, CA 94596
Wil Cossel
Sat. 11 a.m. to 3 p.m.

Amateurs and Artesians Computing
P.O. Box 682
Cobb, CA 95426

Manteca VIC 20 Users Organization
429 N. Main St.
Manteca, CA 95336
Gene Rong

Pomona Valley Vic Users Group
1401 W. 9th, #77
Pomona, CA 91766
(714) 620-8889
Mark Joerger
1st & 3rd Wed. of month 7 p.m.

20/64 Users Group
P.O. Box 18473
San Jose, CA 95158
Don Cracraft
1st Sunday, 6 p.m., Mercury Sav

VIC TORII-The VIC 20 Users Group
PSC #1, Box 23467
APO San Francisco, CA 96230
Wesley Clark

The Valley Computer Club
2006 Magnolia Blvd.
Burbank, CA 91506
1st Wed. 7 p.m.

The Commodore Tech. Users
of Orange Co.
P.O. Box 1497
Costa Mesa, CA 92626
(714) 731-5195
Roger Fisher

VIC 20 Software Exchange Club
10530 Sky Circle
Grass Valley, CA 95945
Daniel Upton

user groups

C-64 West Orange County
Users Group
P.O. Box 1457
Huntington Beach, CA 92647
(714) 842-4484
Philip Putman
2nd & 4th Tues. of month

Antelope Valley Commodore
Users Group
POB 4436
Lancaster, CA 93539
(805) 942-2626
James Haner
1st Saturday

Diablo Valley Commodore
Users Group
762 Ruth Dr.
Pleasant Hill, CA 94523
(415) 671-0145
Ben Braver
2nd & 4th Thurs. 7:30 p.m.

Commodore Connection
11652 Valverde Ave.
Riverside, CA 92505
(714) 689-7447
Tony Alvarez

CA. Area Commodore Terminal
Users Society
C.A.C.T.U.S.
P.O. Box 1277
Alta Loma, CA 91701
Darrell Hall

20/64
P.O. Box 18473
San Jose, CA 95158
(408) 978-0546
1st Sun. of month (6-9 p.m.)

8120 Sundance Dr.
Orangevale, CA 95662
(916) 969-2028
Robyn Graves

Software 64
353 California Dr.
Burlingame, CA 94010
(415) 340-7115
Mario Abad

Sacramento Commodore Users Group
8120 Sundance Dr.
Orangevale, CA 95662
(916) 969-2028
Robyn Graves

Peninsula Commodore Users Group
549 Old County Rd.
San Carlos, CA 94070
(415) 593-7697
Timothy Very
2nd Thurs. of Month

San Francisco Commodore
Users Group
278-27th Ave. #103
San Francisco, CA 94121
(415) 387-0225
Roger Tierce

South Bay Commodore 64
Users Group
P.O. Box 3193
San Ysidro, CA 95073

Commodore 64 West Computer Club
2917 Colorado Ave.
Santa Monica, CA 90404
(213) 828-9308
Don Campbell

COLORADO

VICKIMPET Users Group
4 Waring Lane, Greenwood
Village
Littleton, CO 80121
Contact: Louis Roehrs

Colorado Commodore
Computer Club
2187 S. Golden Ct.
Denver, CO 80227
986-0577
Jack Moss
Meet: 2nd Wed.

CONNECTICUT

John F. Garbarino
Skiff Lane Masons Island
Mystic, CT 06355
(203) 536-9789

Commodore User Club
Wethersfield High School
411 Wolcott Hill Road
Wethersfield, CT 06109
Contact: Daniel G. Spaneas

VIC Users Club
c/o Edward Barszczewski
22 Tunxis Road
West Hartford, CT 06107

New London County
Commodore Club
Doolittle Road
Preston, CT 06360
Contact: Dr. Walter Doolittle

DISTRICT OF COLUMBIA

USO Computer Club
USO Outreach Center
207 Beyer Rd., SW
Washington, DC 20332
Steven Guenther

FLORIDA

Jacksonville Area
PET Society
401 Monument Road, #177
Jacksonville, FL 32211

Richard Prestien
6278 SW 14th Street
Miami, FL 33144

South Florida
PET Users Group
Dave Young
7170 S.W. 11th
West Hollywood, FL 33023
(305) 987-6982

PETs and Friends
129 NE 44 St.
Miami, FL 33137
Richard Plumer

Sun Coast VICs
P.O. Box 1042
Indian Rocks Beach, FL
33535
Mark Weddell

Bay Commodore Users
Group
c/o Gulf Coast Computer
Exchange
241 N. Tyndall Pkwy.
P.O. Box 6215
Panama City, FL 32401
(904) 785-6441
Richard Scofield

Gainesville Commodore
Users Club
3604-20A SW 31st Dr.
Gainesville, FL 32608
Louis Wallace

Brandon Users Group
108 Anglewood Dr.
Brandon, FL 33511
(813) 685-5138
Paul Daugherty

Brandon Commodore Users Group
414 E. Lumsden Rd.
Brandon, FL 33511

Gainesville Commodore
Users Group
Santa Fe Community College
Gainesville, FL 32602
James E. Birdsell

Commodore Computer Club
P.O. Box 21138
St. Petersburg, FL 33742

Commodore Users Group
545 E. Park Ave.
Apt. #2
Tallahassee, FL 32301
(904) 224-6286
Jim Neill

The Commodore Connection
P.O. Box 6684
West Palm Beach, FL 33405

El Shift OH
P.O. Box 548
Cocoa, FL 32922
Mike Schnoke
Sat. mornings/every 4 to 6 weeks

Miami 20/64
12911 S.W. 49th St.
Miami, FL 33175
(305) 226-1185

Tampa Bay Commodore
Computer Club
10208 N. 30th St.
Tampa, FL 33612
(813) 977-0877

Commodore Computer Club
P.O. Box 9726
Jacksonville, FL 32208
(904) 764-5457
David Phillips
2nd & 4th Tues. of Month

VIC/64 Heartland Users Group
1220 Bartow Rd. #23
Lakeland, FL 33801
(813) 666-2132
Tom Keough
4th Wed. of Month at PRC

64 Educators Users Group South
FDLRS-South
9220 S.W. 52nd Terrace
Miami, FL 33165
(305) 274-3501
Dr. Eydie Sloane

64 Educators Users Group North
16330 N.E. 2nd Ave.
North Miami Beach, FL 33162
(305) 944-5548
Robert Figueroa

Suncoast 64S
c/o Little Professor Book Center
2395 U.S. 19 North
Palm Harbor, FL 33563
(813) 785-1036
Curtis Miller

GEORGIA

VIC Educators Users Group
Cherokee County Schools
110 Academy St.
Canton, GA 30114
Dr. Al Evans

Bldg. 68, FLETC
Glynco, GA 31524
Richard L. Young

VIC-tims
P.O. Box 467052
Atlanta, GA 30346
(404) 922-7088
Eric Ellison

Golden Isles Commodore
Users Club
Bldg. 68, FLETC
Glynco, GA 31524
Richard L. Young

Commodore Club of Augusta
1011 River Ridge Rd.
Apt. #14-A
Augusta, GA 30909
David Dumas

Dataswapper Users Group
1794 Alabama Ave.
Albany, GA 31705
(912) 436-5596
David Via

HAWAII

Commodore Users Group of Honolulu
c/o PSH
824 Bannister St.
Honolulu, HI
(808) 848-2088
3rd Fri. every month

20/64 Hawaii
P.O. Box 966
Kailua, HI 96734
Wes Goodpaster

Commodore Users Group of Honolulu
1626 Wilder #701
Honolulu, HI 96822
(808) 848-2088
Jay Calvin (808) 944-9380

IDAHO

GHS Computer Club
c/o Grangeville High School
910 S. D St.
Grangeville, ID 83530
Don Kissinger

S.R.H.S. Computer Club
c/o Salmon River H.S.
Riggins, ID 83549
Barney Foster

Commodore Users
548 E. Center
Pocatello, ID 83201
(208) 233-0670
Leroy Jones

Eagle Rock Commodore Users Group
900 S. Emerson
Idaho Falls, ID 83401
Nancy J. Picker

64 Bug (Boise Users Group)
P.O. Box 276
Boise, ID 83701
(208) 344-6302
John Rosecrans

ILLINOIS

Shelly Wernikoff
2731 N. Milwaukee
Avenue
Chicago, IL 60647

VIC 20/64 Users Support
Group
c/o David R. Tarvin
114 S. Clark Street
Pana, IL 62557
(217) 562-4568

Central Illinois PET User
Group
635 Maple
Mt. Zion, IL 62549
(217) 864-5320
Contact: Jim Oldfield

ASM/TED User Group
200 S. Century
Rantoul, IL 61866
(217) 893-4577
Contact: Brant Anderson

PET VIC Club (PVC)
40 S. Lincoln
Mundelein, IL 60060
Contact: Paul Schmidt,
President

Rockford Area PET Users
Group
1608 Benton Street
Rockford, IL 61107

Commodore Users Club
1707 East Main St.
Olney, IL 62450
Contact: David E. Lawless

VIC Chicago Club
3822 N. Bell Ave.
Chicago, IL 60618
John L. Rosengarten

Chicago Commodore 64
Users & Exchange Group
P.O. Box 14233
Chicago, IL 60614
Jim Robinson

Fox Valley PET Users
Group
833 Willow St.
Lake in the Hills, IL 60102
(312) 658-7321
Art DeKneef

The Commodore 64 Users
Group
P.O. Box 572
Glen Ellyn, IL 60137
(312) 790-4320
Gus Pagnotta

RAP 64/VIC Regional
Assoc. of Programmers
10721 S. Lamon
Oak Lawn, IL 60453
Bob Hughes

The Kankakee Hackers
RR #1, Box 279
St. Anne, IL 60964
(815) 933-4407
Rich Westerman

WIPUG
Rt. 5, Box 75
Quincy, IL 62301
(217) 656-3671
Edward Mills

Papug-Peoria Area Pet Users Group
6 Apple Tree Lane
East Peoria, IL 61611
(309) 673-6635
Max Taylor
2nd Fri. of Month

INDIANA

PET/64 Users
10136 E. 96th St.
Indianapolis, IN 46256
(317) 842-6353
Jerry Brinson

Cardinal Sales
6225 Coffman Road
Indianapolis, IN 46268
(317) 298-9650
Contact: Carol Wheeler

CHUG (Commodore
Hardware Users Group)
12104 Meadow Lane
Oaklandon, IN 46236
Contact: Ted Powell

VIC Indy Club
P.O. Box 11543
Indianapolis, IN 46201
(317) 898-8023
Ken Ralston

Northern Indiana
Commodore Enthusiasts
927 S. 26th St.
South Bend, IN 46615
Eric R. Bean

Commodore Users Group
1020 Michigan Ave.
Logansport, IN 46947
(219) 722-5205
Mark Bender

Computer Workshop VIC 20/64 Club
282 S. 600 W.
Hebron, IN 46341
(219) 988-4535
Mary O'Bringer

The National Science Clubs
of America
Commodore Users Division
7704 Taft St.
Merrillville, IN 46410
Brian Lepley or Tom Vlasic

East Central Indiana VIC User Group
Rural Route #2
Portland, IN 47371
Stephen Erwin

National VIC 20 Program Exchange
102 Hickory Court
Portland, IN 47371
(219) 726-4202
Stephen Erwin

Commodore Computer Club
3814 Terra Trace
Evansville, IN 47711
(812) 477-0739
John Patrick, President

Commodore 64 Users Group
912 South Brown Ave.
Terre Haute, IN 47803
(812) 234-5099
Dennis Graham

Seymour Peekers
c/o D&L Camera Shop
108 N. Chestnut
Seymour, IN 47274
Dennis Peters

IOWA

Commodore User Group
114 8th St.
Ames, IA 50010

Quad City Commodore Club
1721 Grant St.
Bettendorf, IA 52722
(319) 355-2641
John Yigas

Siouxland Commodore Club
2700 Sheridan St.
Sioux City, IA 51104
(712) 258-7903
Gary Johnson
1st & 3rd Monday of month

421 W. 6th St.
Waterloo, IA 50702
(319) 232-1062
Frederick Volker

Commodore Computer Users
Group of Iowa
Box 3140
Des Moines, IA 50316
(515) 263-0963 or (515) 287-1378
Laura Miller

Commo-Hawk Commodore
Users Group
P.O. Box 2724
Cedar Rapids, IA 52406
Vern Rotert

KANSAS

Wichita Area PET
Users Group
2231 Bullinger
Wichita, KS 67204
(316) 838-0518
Contact: Mel Zandler

Kansas Commodore
Computer Club
101 S. Burch
Olathe, KS 66061
Contact: Paul B. Howard

Commodore Users Group
6050 S. 183 St. West
Viola, KS 67149
Walter Lounsbery

Walnut Valley Commodore
User Group
1003 S. 2nd St.
Arkansas City, KS 67005
Bob Morris

KENTUCKY

VIC Connection
1010 S. Elm
Henderson, KY 42420
Jim Kemp

Louisville Users of Commodore KY.
(LUCKY)
P.O. Box 22244
Louisville, KY 40222
(502) 425-2847
2nd Tues. of Month

The Bowling Green Commodore
Users Group
Route 11, Creekside Apt. #6
Bowling Green, KY 42101
(502) 781-9098
Alex Fitzpatrick

LOUISIANA

Franklin Parish Computer
Club
#3 Fair Ave.
Winnisboro, LA 71295
James D. Mays, Sr.

NOVA
917 Gordon St.
New Orleans, LA 70117
(504) 948-7643
Kenneth McGruder, Sr.

VIC 20 Users Group
5064 Bowdon St.
Marrero, LA 70072
(504) 341-5305
Wayne D. Lowery, R.N.

64-Club News
5551 Corporate Blvd.
Suite 3L
Baton Rouge, LA 70808
(504) 766-7408
Tom Parsons
3rd Tues. of month at CWA

Commodore Users Group of Oachita
P.O. Box 175
Swaric, LA 71281
(318) 343-8044
Beckie Walker

Ark-La-Tex Commodore 64 Club
198 India Dr.
Shreveport, LA 71115
(318) 797-9702
Pete Whaley

MARYLAND

Assoc. of Personal
Computer Users
5014 Rodman Road
Bethesda, MD 20016

Blue TUSK
700 East Joppa Road
Baltimore, MD 21204
Contact: Jim Hauff

House of Commodore
8835 Satyr Hill Road
Baltimore, MD 21234
Contact: Ernest J. Fischer

Long Lines Computer Club
323 N. Charles St., Rm. 201
Baltimore, MD 21201
Gene Moff

VIC & 64 Users Group
The Boyds Connection
21000 Clarksburg Rd.
Boyds, MD 20841
(301) 428-3174
Tom DeReggi

Rockville VIC/64 Users Group
5112 Parklawn Terrace
Apt. #103
Rockville, MD 20853
(301) 231-7823
Tom Pounds

The Compucats' Commodore
Computer Club
680 W. Bel Air Ave.
Aberdeen, MD 21001
(301) 272-0472
Betty Jane Schueler

user groups

Westinghouse BWI
Commodore User Group
Attn: L. Barron   Mail Stop 5156
P. O. Box 1693
Baltimore, MD 21203

HUG (Hagerstown Users Group)
23 Coventry Lane
Hagerstown, MD 21740
(301) 797-9728
Joseph Rutkowski
1st & 3rd Fri. of Month

The Montgomery Ct. Commodore
Computer Soc.
P.O. Box 6444
Silver Springs, MD 20906
(301) 946-1564
Meryle Pounds

MASSACHUSETTS

Eastern Massachusetts
VIC Users Group
c/o Frank Ordway
7 Flagg Road
Marlboro, MA 02173

VIC Users Group
c/o Ilene Hoffman-Sholar
193 Garden St.
Needham, MA 02192

Commodore Users Club
Stoughton High School
Stoughton, MA 02072
Contact: Mike Lennon

Berkshire PET Lovers
CBM Users Group
Taconic High
Pittsfield, MA 01201

The Boston Computer
Society
Three Center Plaza
Boston, MA 02108
(617) 367-8080
Mary E. McCann

Masspet Commodore Users Group
P.O. Box 307
East Taunton, MA 02718
David Rogers

Raytheon Commodore Users Group
Raytheon Company
Hartwell Rd. GRA-6
Bedford, MA 01730
John Rudy

Commodore 64 Users
Group of The Berkshires
184 Highland Ave.
Pittsfield, MA 01201
Ed Rucinski

VIC Interface Club
48 Van Cliff Ave.
Brockton, MA 02401
Bernie Robichaud

Cape Cod 64 Users Group
358 Forrest Rd.
S. Yarmouth, MA 02664
1 (800) 225-7136
Jim Close
(In MA. call) 1 (800) 352-7787

The Cursor Club
442 Mulpuf Rd.
Lunenburg, MA 01462
(617) 582-0529
John

MICHIGAN

David Liem
14361 Warwick Street
Detroit, MI 48223

VIC Users Club
University of Michigan
School of Public Health
Ann Arbor, MI 48109
Contact: John Gannon

Commodore User Club
32303 Columbus Drive
Warren, MI 48093
Contact: Robert Steinbrecher

Commodore Users Group
c/o Family Computer
3947 W. 12 Mile Rd.
Berkley, MI 48072

VIC for Business
6027 Orchard Ct.
Lansing, MI 48910
Mike Marotta

South Computer Club
South Jr. High School
45201 Owen
Belleville, MI 48111
Ronald Ruppert

Commodore Users Group
c/o Eaton Rapids Medical Clinic
101 Spicerville Hwy.
Eaton Rapids, MI 48827
Albert Meinke III, M.D.

South East Michigan Pet
Users Group
Box 214
Farmington, MI 48024
Norm Eisenberg

Commodore Computer Club
4106 Eastman Rd.
Midland, MI 48640
(517) 835-5130
John Walley
9:30 p.m. Sept/May

VIC, 64, PET Users Group
8439 Arlis Rd.
Union Lake, MI 48085
363-8539
Bert Searing

COMP
486 Michigan Ave.
Marysville, MI 48040
(313) 364-6804
M. Gauthier

Ann Arbor Commodore Users Group
Ann Arbor, MI 48103
(313) 994-4751
Art Shaw
3rd Tues. 7:30-10:00

DAB Computer Club
P. O. Box 542
Watervliet, MI 49098
(616) 463-5457
Dennis Burlingham

West Michigan Commodores
c/o R. Taber
1952 Cleveland Ave., S.W.
Wyoming, MI 49509
(616) 458-9724
Gene Traas

Debug
P.O. Box 196
Berrien Springs, MI 49103
(616) 471-1882
Herbert Edward
Last Thursday of Month

Jackson Commodore Computer Club
201 S. Grinnell St.
Jackson, MI 49203
Alfred Bruey
Last Thur. of Month 7:30 p.m.

SMCUG
1002 Pfau St.
Mankato, MI 56001
(507) 625-6942
Dean Otto

SEM 64
25015 Five Mile #3
Redford, MI 48239
(313) 537-4163
Gary Groeller

MINNESOTA

MUPET (Minnesota Users
of PET)
P.O. Box 179
Annandale, MN 55302
c/o Jon T. Minerich

Twin Cities Commodore
Computer Club
6623 Ives Lane
Maple Grove, MN 55369
(612) 424-2425
Contact: Rollie Schmidt

MISSISSIPPI

Commodore Biloxi
User Group (ComBUG)
Universal Computer Services
3002 Hwy. 90 East
Ocean Springs, MS 39564
(601) 875-1173
John Lassen

MISSOURI

KCPUG
5214 Blue Ridge Boulevard
Kansas City, MO 64133
Contact: Rick West
(816) 356-2382

Commodore User Group of St. Louis
Box 6653
St. Louis, MO 63125-0653
Dan Weidman, New Members
1541 Swallowtail Dr.
St. Louis, MO

VIC INFONET
P.O. Box 1069
Branson, MO 65616
(417) 334-6099
Jory Sherman

Worth County PET Users
Group
Grant City, MO
(816) 564-3551
David Hardy

Mid-Missouri Commodore Club
1804 Vandiver Dr.
Columbia, MO 65201
(314) 474-4511
Phil Bishop

Joplin Commodore Computers
Users Group
422 S. Florida Ave.
Joplin, MO 64801
R. D. Connely

MONTANA

Powder River
Computer Club
Powder River County
High School
Broadus, MT 59317
Contact: Jim Sampson

Commodore User Club
1109 West Broadway
Butte, MT 59701
Contact: Mike McCarthy

NEBRASKA

Greater Omaha Commodore 64
Users Group
2932 Leawood Dr.
Omaha, NE 68123
(402) 292-2753
Bob Quisenberry

NEVADA

Las Vegas PET Users
Suite 5-315
5130 E. Charleston Blvd.
Las Vegas, NV 89122
Gerald Hasty

Compu Club 64
4220 S. Maryland Parkway
Bldg. B—Suite 403
Las Vegas, NV 89109
(702) 369-7354
Cindy Springfield

NEW JERSEY

Commodore Friendly User Group
49 Hershey Rd.
Wayne, NJ 07470
(201) 696-8043
Rich Pinto/Colin Campbell

Somerset Users Club
49 Marcy Street
Somerset, NJ 08873
Contact: Robert Holzer

Educators Advisory
P.O. Box 186
Medford, NJ 08055
(609) 953-1200
John Handfield

VIC-TIMES
46 Wayne Street
Edison, NJ 08817
Thomas R. Molnar

VIC 20 User Group
67 Distler Ave.
W. Caldwell, NJ 07006
(201) 284-2281
G. M. Amin

VIC Software Development Club
77 Fomalhaut Ave.
Sewell, NJ 08080
H. P. Rosenberg

ACGNJ PET/VIC/CBM
User Group
30 Riverview Terr.
Belle Mead, NJ 08502
(201) 359-3862
J. M. Pylka

South Jersey Commodore Computer
Users Club
46-B Monroe Park
Maple Shade, NJ 08052
(609) 667-9758
Mark Orthner
2nd Fri. of month

Parsippany Computer Group
51 Ferncliff Rd.
Morris Plains, NJ 07950
(201) 267-5231
Bob Searing

Information 64
16 W. Ridgewood Ave.
Ridgewood, NJ 07450
(201) 447-4432
Dave Garaffa

NEW HAMPSHIRE

Northern New England
Computer Society
P.O. Box 69
Berlin, NH 03570

TBH VIC-NICs
P.O. Box 981
Salem, NH 03079

C-64 U.S.E.R.S. User Software
Exchange Pro
P.O. Box 4022
Rochester, NH 03867
Paul Kyle

NEW MEXICO

Commodore Users Group
6212 Karlson, NE
Albuquerque, NM 87113
(505) 821-5812
Danny Byrne

NEW YORK

Capital District 64/VIC 20
Users Group
363 Hamilton St.
Albany, NY 12210
(518) 436-1190
Bill Pizer

Long Island PET Society
Ralph Bressler
Harborfields HS
Taylor Avenue
Greenlawn, NY 11740

PET User Club
of Westchester
P.O. Box 1280
White Plains, NY 10602
Contact: Ben Meyer

LIVE (Long Island
VIC Enthusiasts)
17 Picadilly Road
Great Neck, NY 11023
Contact: Arnold Friedman

Commodore Masters
25 Croton Ave.
Staten Island, NY 10301
Contact: Stephen Farkouh

VIC Users Club
76 Radford St.
Staten Island, NY 10314
Contact: Michael Frantz

West Chester County VIC
Users Group
P.O. Box 146
Pelham, NY 10552
Joe Brown

SPUG
4782 Boston Post Rd.
Pelham, NY 10803
Paul Skipski

VIC 20 User Club
151-28 22nd Ave.
Whitestone, NY 11357
Jean F. Coppola

VIC 20 User Club
339 Park Ave.
Babylon, NY 11702
(516) 669-9126
Gary Overman

VIC User Group
1250 Ocean Ave.
Brooklyn, NY 11230
(212) 859-3030
Dr. Levitt

L&M Computer Club
VIC 20 & 64
4 Clinton St.
Tully, NY 13159
(315) 696-8904
Dick Mickelson

Commodore Users Group
1 Corwin Pl.
Lake Katrine, NY 12449
J. Richard Wright

VIC 20/Commodore 64
Users Group
31 Maple Dr.
Lindenhurst, NY 11757
(516) 957-1512
Pete Lobol

VIC Information Exchange
Club
336 W. 23 St.
Deer Park, NY 11729
Tom Schlegel
SASE & phone please

New York Commodore
Users Group
380 Riverside Dr., 7Q
New York, NY 10025
(212) 566-6250
Ben Tunkelang

Hudson Valley Commodore Club
1 Manor Dr.
Woodstock, NY 12498
F.S. Goh
1st Wednesday of month

LIVICS (Long Island VIC Society)
20 Spyglass Lane
East Setauket, NY 11733
(516) 751-7844
Lawrence Stefani

VIC Users Group
c/o Stoney Brook Learning Center
1424 Stoney Brook Rd.
Stoney Brook, NY 11790
(516) 751-1719
Robert Wurtzel

Poughkeepsie VIC User Group
2 Brooklands Farm Rd.
Poughkeepsie, NY 12601
(914) 462-4518
Joe Steinman

VIC 20 User Group
Paper Service Division
Kodak Park
Rochester, NY 14617
David Upham, Sr.

Manhattan 64
426 West 48th
New York, NY 10036
(212) 242-3900
Charles Honce

Adirondack Commodore 64
Users Group
205 Woodlawn Ave.
Saratoga Springs, NY
(518) 584-8960
Paul Klompas

Rockland County Commodore
Users Group
P.O. Box 573
Nanuet, NY 10965
Ross Garber

New York 64 Users Group
222 Thompson St.
New York, NY 10012
(212) 673-7241
Bruce Cohen

Finger Lakes Commodore
Users Group
c/o Rose City Computer Associates
229 West Union St.
Newark, NY 14513
(315) 331-1185

The Commodore Users Group
Rochester
78 Hardison Rd.
Rochester, NY 14617
(716) 544-5251
Tom Werenski
Phone Evenings between 7-10

Commodore Computer Club
Publications Dept.,
Grumman Aerospace
1111 Stewart Ave.
Bethpage, NY 11714
(516) 575-9558
Neil Threulsen

Hello, Central!
76-12 35th Ave.
Jackson Heights, NY 11372
Jared Sherman

VIC 20/64 Users Group
NYU Waverly Place
New York, NY 10003
(212) 358-5155
Lawrence Schulman

SCUG (Schenectady Commodore
Users Group)
c/o the Video Connection
Canal Square
Schenectady, NY 12305
Timothy Davis
1st Mon. of Month

NORTH CAROLINA

Amateur Radio PET Users Group
P.O. Box 30694
Raleigh, NC 27622
Contact: Hank Roth

VIC Users Club
c/o David C. Fonenberry
Route 3, Box 351
Lincolnton, NC 28092

Microcomputer Users Club
Box 17142 Bethabara Sta.
Winston-Salem, NC 27116
Joel D. Brown

VIC Users Club
Rt. 11, Box 686
Hickory, NC 28601
Tim Gromlovits

Raleigh VIC 20/64 Users Group
410-D Delta Court
Cary, NC 27511
(919) 469-3862
Larry Diener

Down East Commodores
302 Belltown Rd.
Havelock, NC 28532
(919) 447-4536
Bruce Thedin

OHIO

Dayton Area PET
User Group
933 Livingston Drive
Xenia, OH 45385
B. Worby, President
(513) 848-2065
J. Watson, Secretary
(513) 372-2052

Central Ohio PET
Users Group
107 S. Westmoor Avenue
Columbus, OH 43204
(614) 274-0304
Contact: Philip H. Lynch

Commodore Computer Club
of Toledo
734 Donna Drive
Temperance, MI 48182
Gerald Carter

Chillicothe Commodore
Users Group
P.O. Box 211
Chillicothe, OH 45601
William A. Chaney

Licking County 64 Users Group
323 Schuler St.
Newark, OH 43055
(614) 345-1327

11433 Pearl Rd.
Strongsville, OH 44136
Paul M. Warner

C.P.U. Connection
P.O. Box 42032
Brook Park, OH 44142
Danni Hudak

Commodore Users Group
18813 Harlan Dr.
Maple Heights, OH 44137
(216) 581-3099
Carl Skala

Commodore Users of
Blue Chip (Cincinnati)
816 Beecher St.
Cincinnati, OH 45206
(513) 961-6582
Ted Stalets

OKLAHOMA

Southwest Oklahoma
Computer Club
c/o Commodore Chapter
P.O. Box 6646

user groups

Lawton, OK 73504
1:30 at Lawton City Library

Tulsa Area Commodore Users Group
Tulsa Computer Society
P.O. Box 15238
Tulsa, OK 74112
Annette Hinshaw

Commodore Oklahoma Users Club
4000 NW 14th St.
Oklahoma City, OK 73107
(405) 943-1370
Stanley B. Dow

Commodore Users
Box 268
Oklahoma City, OK 73101
Monte Maker, President

Commodore Users of Norman
209 Brookwood
Noble, OK 73068
Matt Hager

OREGON

NW PET Users Group
John F. Jones
2134 N.E. 45th Avenue
Portland, OR 97213

U.S. Commodore Users Group
P.O. Box 2310
Roseburg, OR 97470
(503) 672-7591
Richard Tsukiji

Southern Oregon VIC/64
Users Group
3600 Madrona Lane
Medford, OR 97501
(503) 779-7631
James Powell

PENNSYLVANIA

PET User Group
Gene Beals
P.O. Box 371
Montgomeryville, PA 18936

Penn Conference Computer Club
c/o Penn Conference of SDA
720 Museum Road
Reading, PA 19611
Contact: Dan R. Knepp

PACS Commodore Users Group
LaSalle College
20th & Olney Ave.
Philadelphia, PA 19141
(215) 951-1258
Stephen Longo

Glen Schwartz
807 Avon
Philadelphia, PA 19116

Gene Planchak
4820 Anne Lane
Sharpsville, PA 15150
(412) 962-9682

PPG (Pittsburgh PET Group)
c/o Joel A. Casar, DMD
2015 Garrick Drive
Pittsburgh, PA 15235
(412) 371-2882

Westmoreland Commodore
Users Club
c/o DJ & Son Electronics
Colonial Plaza
Latrobe, PA 15650
Jim Mathers

Commodore Users Club
3021 Ben Venue Dr.
Greensburg, PA 15601
(412) 836-2224
Jim Mathers

VIC 20 Programmers, Inc.
c/o Watson Woods
115 Old Spring Rd.
Coatesville, PA 19320
Robert Gougher

G.R.C. User Club
300 Whitten Hollow Rd.
New Kensington, PA 15068
Bill Bolt

NADC Commodore Users Club
248 Oakdale Ave.
Horsham, PA 19044
Norman McCrary

CACC (Capitol Area Commodore
Club)
134 College Hill Rd.
Enola, PA 17025
(717) 732-2123
Lewis Buttery
Union Deposit Mall at 7 p.m.

G/C Computer Owners Group
c/o Gilbert Associates, Inc.
P.O. Box 1498
Reading, PA 19607
Extension 6472
Jo Lambert (215) 775-2600

Boeing Employees Personal
Computer Club
The Boeing Vertol Co.
P.O. Box 16858
Philadelphia, PA 19142
(215) 522-2257
Jim McLaughlin

South Central PA Commodore Club
2109 Cedar Run Dr.
Camp Hill, PA 17011
(717) 763-4219
David Persing

Main Line Commodore Users
Group (MLCUG)
c/o Main Line Computer Center
1046 General Allen Lane
West Chester, PA 19380
(215) 388-1581
Emil Volcheck

Commodore Users Group
781 Dick Ave.
Warminster, PA 18974
Matt Matulaitis

The Commodore Users Club
of S.E. Pittsburgh
c/o Groves Appliance & TV
2407 Pennsylvania Ave.
West Mifflin, PA 15122
Charles Groves

Compstars
130 Blue Teal Circle
Audubon, PA 19403
Mike Norm
Meet at Audio Video Junct.

Scranton Commodore Users Group
P.O. Box 211
Clarks Summit, PA 18411

Clifton Heights Users Group
P.O. Box 235
Clifton Heights, PA 19018

Oxford Circle 64 Users Group
Frankford Cong. Un. Church of Christ
Oxford Ave. & Pratt St.
Philadelphia, PA 19124
(215) 743-8999
Roger Nazeley (215) 535-9021
4th Wed. of Month

VIC Software Development Club
440 W. Sedgwick
Apt. A-1
Philadelphia, PA 19119
(215) 844-4328
Tracy Lee Thomas

Bits & Bytes
1015 Dale Rd.
Secane, PA 19018
(215) 544-5875
Dave Boodey

PUERTO RICO

CUG of Puerto Rico
RFD #1, Box 13
San Juan, PR 00914
Ken Burch

VIC 20 User Group
655 Hernandez St.
Miramar, PR 00907
Robert Morales, Jr.

RHODE ISLAND

Irving B. Silverman, CPA
160 Taunton Ave.
E. Providence, RI 02914
Contact: Michelle Chavanne

Newport VIC/64 Users
10 Maitland Ct.
Newport, RI 02840
(401) 849-2684
Dr. Matt McConeghy

The VIC 20 Users Club
Warwick, RI 02886
Tom Davey

Commodore Users Group
c/o Data-Co.
978 Tiogue Ave.
Coventry, RI 02816
(401) 828-7385
Victor Moffett

SOUTH CAROLINA

Beaufort Technical College
100 S. Ribaut Rd.
Beaufort, SC 29902
Dean of Instruction

Computer Users Society
of Greenville (CUS)
Horizon Records-Home Computers
347 S. Pleasantburg Dr.
Greenville, SC 29607
(803) 235-7922
Bo Jeanes

Commodore Computer Club
of Columbia
318 Quincannon Dr.
Columbia, SC 29210
Buster White Sect./Treas.

Spartanburg Commodore
Users Group
803 Lucerne Dr.
Spartanburg, SC 29302
(803) 582-5897
James Pasley

SOUTH DAKOTA

PET User Group
515 South Duff
Mitchell, SD 57301
(605) 996-8277
Contact: Jim Dallas

VIC/64 Users Club
608 West 5th
Pierre, SD 57501
(605) 224-4863
Larry Lundeen

TENNESSEE

River City Computer
Hobbyists
Memphis, TN
1st Mon. at Main Library

Nashville Commodore Users Group
P.O. Box 121282
Nashville, TN 37212
(615) 331-5408
Dave Rushing
3rd Thurs. at Cumberland Mus

Commodore User Club
Metro Computer Center
1800 Dayton Blvd.
Chattanooga, TN 37405
Mondays 7:30 pm

Metro-Knoxville 64 Users Club
7405 Oxmoor Rd., Rt. #20
Knoxville, TN 37921
(615) 938-3773
Ed Pritchard

Memphis Commodore Users Group
2476 Redvers Ave.
Memphis, TN 38127
(901) 358-5823
Harry Ewart

TEXAS

PET Users
2001 Bryan Tower
Suite 3800
Dallas, TX 75201

Larry Williams
P.O. Box 652
San Antonio, TX 78293

PET User Group
John Bowen
Texas A & M
Microcomputer Club
Texas A & M, TX

CHUG (Commodore Houston
Users Group)
8738 Wildforest
Houston, TX 77088
(713) 999-3650
Contact: John Walker

Commodore Users Group
5326 Cameron Rd.
Austin, TX 78723
(512) 459-1220
Dr. Jerry D. Frazee

VIC Users Group
3817 64th Dr.
Lubbock, TX 79413

64 Users Group
2421 Midnight Circle
Plano, TX 75075
S.G. Grodin

Savid Computer Club
312 West Alabama
Suite 2
Houston, TX 77006
Davi Jordan, Chairman

Gulf Coast Commodore
Users Group
P.O. Box 128
Corpus Christi, TX 78403
(512) 887-4577
Lawrence Hernandez

Mid-Cities Commodore Club
413 Chisolm Trail
Hurst, TX 76053
Garry Wordelman

Mid-Cities Commodore Club
413 Chisolm Trail
Hurst, TX 76053
Bruce Nelson

Interface Computer Club
814 North Sabinas
San Antonio, TX 78207
M. E. Garza, President

Gulf Coast Commodore
Users Group
P.O. Box 128
Corpus Christi, TX 78403
(512) 887-4577
Lawrence Hernandez

ICUG (Irving Commodore
Users Group)
3237 Northgate #1289
Irving, TX 75062
(214) 252-7017
Robert Hayes

UTAH

Utah PUG
Jack Fleck
2236 Washington Blvd.
Ogden, UT 84401

The Commodore Users
Club
742 Taylor Avenue
Ogden, UT 84404
Contact: Todd Woods Kap,
President
David J. Shreeve,
Vice President

The VIClic
799 Ponderosa Drive
Sandy, UT 84070
Contact: Steve Graham

VIC 20 Users
324 N. 300 W.
Smithfield, UT 84335
Dave DeCorso

Northern Utah VIC & 64
Users Group
P.O. Box 533
Garland, UT 84312
David Sanders

The Commodore Users Group
652 West 700 North
Clearfield, UT 84015
(801) 776-3950
Rodney Keller, Richard Brenchly

Mountain Computer Society
P.O. Box 1154
Sandy, UT 84091
Dave Tigner

VIRGINIA

Northern VA PET Users
Bob Karpen
2045 Eakins Court
Reston, VA 22091
(803) 860-9116

VIC Users Group
Rt. 2, Box 180
Lynchburg, VA 24501
Contact: Dick Rossignol

VIC Users Group
c/o Donnie L. Thompson
1502 Harvard Rd.
Richmond, VA 23226

Dale City Commodore
User Group
P.O. Box 2004
Dale City, VA 22193
(703) 680-2270
James Hogler

Tidewater Commodore
Users Group
4917 Westgrove Rd.
Virginia Beach, VA 23455
Fred Monson

Fredericksburg Area
Computer Enthusiasts
P.O. Box 324
Locust Grove, VA 22508
(703) 972-7195
Michael Parker

VIC 20 Victims
4301 Columbia Pike #410
Arlington, VA 22204
(703) 920-0513
Mike Spengel

Peninsula Commodore 64
Users Group
124 Burnham Place
Newport News, VA 23606
(804) 595-7315
Richard G. Wilmoth

Norfolk Users Group
1030 West 43rd St. B-4
Norfolk, VA 23508
489-8292
Larry Pearson

NASA VIC 20 User Group
713 York Warwick Dr.
Yorktown, VA 23692
Harris Hamilton

135 Beverley Rd.
Danville, VA 24541
David Gray

R.A.C.E. Commodore
Users Group
4726 Horseman Dr.
Roanoke, VA 24019
(703) 362-3960
Larry Rackow

WASHINGTON

NW PET Users Group
2565 Dexter N. 3203
Seattle, WA 98109
Contact: Richard Ball

PET Users Group
c/o Kenneth Tong
1800 Taylor Ave. N102
Seattle, WA 98102

Whidbey Island Commodore
Computer Club
947 N. Burroughs Ave.
Oak Harbor, WA 98277
Michael D. Clark

Central Washington
Commodore Users Group
1222 S. 1st St.
Yakima, WA 98902
Tim McElroy

Blue Mountain Commodore
Users Club
15 Stone St.
Walla Walla, WA 99362
(509) 525-5452
Keith Rodue

Spokane Commodore User Group
N. 4311 Whitehouse
Spokane, WA 99205
(509) 328-1464
Stan White

CBM Users Group
803 Euclid Way
Centralia, WA 98531
(206) 736-4085
Rick Beaber

Computer Club
c/o Honeywell, Inc.
5303 Shilshole Ave., NW
Seattle, WA 98107
(206) 789-2000
John Goddard

WEST VIRGINIA

Personal Computer Club
P.O. Box 1301
Charleston, WV 25325
Cam Cravens

TriState Commodore Users
73 Pine Hill Estates
Kenova, WV 25530
(304) 453-2124
Marc Hutton

WISCONSIN

Sewpus
c/o Theodore J. Polozynski
P.O. Box 21851
Milwaukee, WI 53221

Waukesha Area Commodore
User Group (WACUG)
256½ W. Broadway
Waukesha, WI 53186
Contact: Walter Sadler
(414) 547-9391

Commodore User Group
1130 Elm Grove St.
Elm Grove, WI 53122
Tony Hunter

Commodore 64 Software
Exchange Group
P.O. Box 224
Oregon, WI 53575
E. J. Rosenberg

C.L.U.B. 84
6156 Douglas Ave.
Caledonia, WI 53108
(414) 835-4645 pm
Jack White
2nd Sat every month 10:00 am

VIC-20 & 64 User Group
522 West Bergen Dr.
Milwaukee, WI 53217
(414) 476-8125
Mr. Wachtl

Menomonie Area Commodore
Users Group
510 12th St.
Menomonie, WI 54751
(715) 235-4987
Mike Williams

C.U.S.S.H.
3614 Sovereign Dr.
Racine, WI 53406
(414) 554-0156
Tim Tremmel
3rd Saturday of month

Madison Area Commodore
Users Group
1552 Park St.
Middleton, WI 53562
(608) 831-4852
John Carvin
3rd Thurs. each month

S.W.I.T.C.H.
W156 N8834 Pilgrim Rd.
Menomonee Falls, WI 53051
(414) 255-7044
Len Lutz

Milwaukee Area CBM64
Enthusiasts (M.A.C.E.)
P.O. Box 340
Elm Grove, WI 53122
(414) 259-5991
Kevin Wilde

The Eau Claire CBM64
Users Group
Rt. 5, Box 179A
Eau Claire, WI 54703
(715) 874-5972
John Slavsky, Jr.
2nd Thurs. 7 p.m.

WYOMING

Commodore Users Club
c/o Video Station
670 North 3rd #B
Laramie, WY 82070
(307) 721-5908
Pamela Nash

AUSTRALIA

WA VIC-Ups (VIC 20/CBM 64 Users)
14 Glengariff Dr.
Floreat Park 6014
Western Australia
B. J. Cook

CANADA

Toronto PET
Users Group, Inc.
1912A Avenue Rd., Ste. 1
Toronto, Ontario, Canada
M5M 4A1
(416) 782-8900
or call 416-782-9252
Contact: Chris Bennett

PET Users Club
c/o Mr. Brown
Valley Heights Secondary School
Box 159
Langton, Ont. N0E 1G0

user groups

Vancouver PET Users Group
P.O. Box 91164
West Vancouver, British
Columbia
Canada V7V 3N6

CCCC (Canadian
Commodore Computer Club)
c/o Strictly Commodore
47 Coachwood Place
Calgary, Alberta, Canada
T3H 1E1
Contact: Roger Olanson

W.P.U.G.
9-300 Enniskillen Ave.
Winnipeg, Manitoba R2V 0H9
Larry Neufeld

VIC-TIMS
2-830 Helena St.
Trail, British Columbia
V1R 3X2
(604) 368-9970
Greg Goss

Arva Hackers
Medway High School
Arva, Ontario N0M 1C0
D. Lerch

Nova Scotia Commodore
Computer Users Group
66 Landrace Cres.
Dartmouth, N.S. B2W 2P9
Andrew Cornwall

Bonnyville VIC Cursors
Box 2100
Bonnyville, Alberta T0A 0L0
(403) 826-3992
Ed Wittchen

Commodore Users Club of Sudbury
938 Brookfield Ave.
Sudbury, Ontario
P3A 4K4

PET Educators Group
P.O. Box 454
Station A
Windsor, Ontario
N9A 6L7

COMVIC
P.O. Box 1688
St. Laurent
Montreal, Quebec
H4L 4Z2

Calgary Commodore Users Group
37 Castleridge Dr., N.E.
Calgary, Alberta
T3J 1P4
John Hazard

Fledging Barrie User Group (BUG)
58 Steel St.
Barrie, Ontario
Canada L4M 2E9

FINLAND

VIC-Club in Helsinki
c/o Matti Aarnio
Linnustajankj 2B7
SF-02940 ESP00 94
Finland

GERMANY

Kettenberg 24
D 5880 Lueden Scheid
West Germany
Rudi Ferrari

ITALY

Commodore 64 Club
Universita di Studi shan
V. Avigliana 13/1
10138 TORINO
ITALY

KOREA

Commodore Users Club
K.P.O. Box 1437
Seoul, Korea
Contact: S. K. Cha

MEXICO

Asociacion De Usarios
Commodore
c/o Alejandro Lopez
Arechiga
Holbein 174-6° Piso
Mexico 18, D.F.

Club de Usarios Commodore
Sigma del Norte
Mol del Valle, Local 44
Garza Garcia, N.L. 66220

Club Microvic
Villaldama 225
Col. Chapultepec
Monterrey, N.L.
Mexico 66450
Oscar Sosa, President

NEW ZEALAND

Commodore Users Group
Meet at VHF Clubrooms
Hazel Ave.
Mount Roskill
3rd Wed. of month, 7:30 pm
Roger Altena 278-5262

Nelson VIC Users Group
c/o P.O. Box 860
Nelson, New Zealand
Peter Archer

E.R. Kennedy
c/o New Zealand Synthetic
Fuels Corp. Ltd.
Private Bag
New Plymouth

NORWAY

VIC Club of Norway
Nedre Bankegt 10,
1750 Halden
Norway

UNITED KINGDOM

North London Hobby
Computer Club
Dept. of Electronics &
Communications
Engineering
The Polytechnic of North
London
Holloway Rd.
London N7 8DB

Croydon Microcomputer Club
111 Selhurst R.
Selhurst, London SE25 6LH
01-653-3207
Vernon Gifford

User Bulletin Board

User Groups Forming:

CALIFORNIA

Stockton Commodore 64's
Contact Rita McCann
at 209-948-5890
or Bob Eustis at 948-1624.

MISSISSIPPI

VIC 20 User Group
Contact Kris Hauser
24 Canterbury
Starkville 39759

MISSOURI

McDonnell-Douglas
Commodore Group
St. Louis employees who wish
to join contact:
Joe Sughero
EO32/74A/1

NEW HAMPSHIRE

Contact New Hampshire College
Portsmouth Center
Route 1 Bypass, Portsmouth
603-436-2831

NORTH CAROLINA

Charlotte Commodore Users Group
Contact H.S. Hanks, Jr.
3717 Flowerfield Road
Charlotte 28210

PENNSYLVANIA

Commodore User Club
Contact Joe Kamenar
225 J Dunbar Lane
Horsham 19044
215-441-8204, 938-0171

WASHINGTON

Washington VIC Club
Contact Eric Thomson
8711 NE 140
Bothell 98011
206-823-6025

MEXICO

Commodore 64 Group
Contact Peter & Lucille Shipley
Apdo 64, Puerto Vallarta
Jalisco, Mexico 48300
(011-52) 322-2-1697

Commodore Amateur Radio Net Forming:

Meets Sundays on 7.260 kilohertz at 1 p.m. Mountain Time. Per John Luker, WB7QBC.

SuperPET Users Group

ISPUG (International SuperPET Users Group) supports SuperPET users throughout the world, provides software and resource information and publishes the SuperPET Gazette bi-monthly. Dues are $15 per year. Write the Editor, SuperPET Gazette, P.O. Box 411, Hatteras, NC 27943, U.S.A.

Electronic Projects Club

Exchange electronic projects and technical articles with others. Membership $15 a year, includes 32-48 page newsletter. Contact Joe Kamenar, 225J Dunbar Lane, Horsham, PA 19044.

POWER64

Reviewed by Elizabeth Deal

POWER64 adds important commands to the Commodore 64 BASIC.

POWER has been around for over two years for the PET/CBM computers. POWER64 is a similar set of great utilities, with several most useful additions over the PET version. As in the PET, POWER does not interfere with program execution. Programs written using POWER run correctly on systems not equipped with POWER. Both tape and disk systems can use POWER.

POWER is distributed by Pro-Line Software Ltd, 755 The Queensway East, Unit 8, Mississauga, Ontario, L4Y 4C5, Canada. Their phone is 416-273-6350. The Canadian price is $99. The U.S. price is $49.95.

POWER64 comes on disk. Since memory use in the 64 can vary, so does the location of POWER. The user gets a chance to pick its living quarters. Useful.

POWER is built of two units: POWER and MOREPOWER. MOREPOWER contains vital disk commands lacking in the Commodore 64, as well as several editing commands that go beyond POWER's 4K size. MOREPOWER is separate, so we can pick and choose. POWER uses little memory outside its own program area: 32 bytes, from 704-736 ($02C0-02E0). It modifies, remembers and restores, when needed, three system vectors to stay alive.

POWER was written by Brad Templeton. Jim Butterfield wrote the instructions—clear, complete, concise and educational—about the whole computer system.

General Information

POWER is not a run-of-the-mill BASIC extender. It contains well thought-out solutions to working with BASIC, and permits customizing the computer to your needs. It even permits adding commands of your choice, the design of this feature being meticulously thought-out.

There is little in the way of hard coding. Programs can be relocated. Every POWER command that works with BASIC in its usual location can be used in any other position. POWER makes no assumptions about device numbers in your system or your general system configuration. It helps, but does not get in the way.

In my opinion a computer without POWER is like a computer without electricity. POWER is just about a must if you do anything more than load programs. It is user-oriented, and makes everything you do easier. For instance, a program can be scrolled on the screen by use of the up and down cursor keys, eliminating the need for typing LIST line numbers.

The entire POWER with or without MOREPOWER is a pleasure to work with. Imagine pushing one key to do a job such as changing the screen colors or calling some elaborate data processing routine!

Program Management Commands

Changing programs is made easy by the search and search-and-replace commands. Both can be asked to report about the next line or all lines in the program text. If you ask for a line-by-line work, pressing one key displays the lines.

Both search and search-and-replace are intelligent commands. By use of several defining characters you may conduct a meaningful search for keywords, variables or text occurring only in a specified place, or for sequences of text, as in "find all occurrences of FOR —NEXT". In the last instance POWER will look for all lines containing the words FOR and NEXT, disregarding what lies in between.

Program lines can be deleted, automatically numbered and renumbered. You can renumber the entire program or any section of the program with any starting line and increment that fits (no moving). Hence, if you like to insert a subroutine or several DATA lines from tape or disk or if you simply need a bit of room for a few lines of new coding, you are not limited to renumbering a whole program. Automatic line numbering can be used at all times or a special procedure can be used to insert prenumbered lines into an existing opening. POWER knows where to begin and where to stop. This is handy.

Existing subroutines or DATA lines can be connected to a program in memory. There are two ways of merging: the EXEC command of POWER can merge programs previously listed as ASCII files, while MERGE (of MOREPOWER) can merge program files. Both have their place. EXEC is powerful, because it can bring in any kind of ASCII file, perhaps even from a communication device. Both commands work by inserting program lines into correct places. They are not limited to appending at the end.

The EXEC command is not limited to merging programs. Anything that can be done in direct mode from the keyboard can be placed on file and automatically executed when brought back to screen—a marvelous feature.

The FIX command is used to reset BASIC pointers following a BASIC load of programs to weird places, such as the tape buffer, which, many of you may know, can damage BASIC pointers. FIX is also used to reconnect POWER in the event it is disconnected by the RESTORE key.

The PTR (pointer fix) command permits fixing BASIC pointers without restarting POWER. It keeps the connections of your own extensions intact. This is a very important command that was missing in the PET version of POWER but was in the POWAID-POWER extension.

POWER64 permits you to have two programs in memory for a while. If you are working on one program and decide to test or use another, you can do it. Example: you have just loaded a long program but forgot to load a machine code routine or to change disk device number. POWER allows you to load in the other program, run it, and come right back to the big job as if nothing happened. MOREPOWER extends this concept even further. It can run program number two automatically after saving all the variables (excluding some strings). While I'm not a great fan of anything running automatically, I must admit that, to my own surprise, I have used this feature on numerous occasions and it was useful to me after all.

MOREPOWER permits combined BASIC plus machine code programs created by PAL (Brad Templeton's assembler) to run in such a partition. This is a super way of testing programs, while the main source program is in memory.

Debugging Commands

The DUMP command displays all single variables and their values (see instant phrases below for dump of arrays). The WHY command puts a marker on the last executed command before an error occurred, telling you where the bug might be. To ease the typing job, POWER provides AUTOmatic line numbering and one-keystroke entry of BASIC keywords.

POWER includes a three-speed TRACE command that is invaluable in detecting errors. There are several types of TRACE permitted, from the simplest display of line numbers being executed to the fanciest display, at the top of the screen, of the line number, its statements and values of the most recent variable, including the results of logical tests used in IF-THEN statements. Alternately, the information can be seen on a scrolling screen with the statements, values and output all mixed up. Depending on your need there is always one way of tracing that will prove more useful than another.

TRACE should prove invaluable not only as a debugging aid, but as an educational aid. Since you can actually see BASIC executing in slow motion, the mysteries of the computer's number-crunching process can be explained by a simple visual demonstration. It's a nice show to watch.

All Purpose Features

I've been a POWER addict ever since I got it about two years ago for the PET. I suffer when I have to program on a computer without POWER. One of the reasons is that POWER spoils you into lazy way of working—one keypress does whole big jobs. You define the jobs. Every day they may be different.

POWER has instant keys, called REM macros. MOREPOWER has an instant key feature called KEY.

There are three levels of complexity: instant keywords, instant phrases and instant subroutines, the latter two being a monument to creative laziness. The user controls which of the three can be active at one time or what combinations of the three can be used to advantage in a particular situation.

"Instant keywords" is the

simplest level. When enabled, typing a key puts in an entire keyword on the screen. For instance, shifted I puts the word INPUT on the screen.

"Instant phrases" is the next level and, at least with me, seem to get the most use. You can define a shifted key, a CTRL-key or CBM-key sequence, to mean any sequence of commands you need. It is done by writing a REM line, which becomes part of your BASIC test but is not executed and will not interfere in running a program on POWERless system.

```
14 REM"CTRL−A=FORJ=
   ATOB:PRINTA(J);:NEXT
15 REM"SHIFT−O=
   OPEN6,4::CMD6:LIST
```

Magic: whenever I want a listing I press shifted-O, the word OPEN and all words following it are written on the screen. I hit RETURN and the printer prints.

Instant phrases can be used in a running program in an INPUT situation. When you debug a program requiring some answers to an input prompt, you can define a key to provide the answer. This saves a lot of typing of the same stuff over and over.

"Instant subroutines" are used when commands can't fit on one line, or when you want to use statements not permitted in direct mode. In this case REM macros define keys to mean "perform a subroutine coded beginning in line 200", thusly:

```
21 REM"CTRL—D backarrow 200
```

When CTRL—D is pressed the code runs and returns. This is very powerful. It can be used for all sorts of utilities. You can customize the REM macros to a particular debugging situation you are faced with. Keep in mind that you need not save your "debugging" utilities each time you edit the program. They can be merged in. They are there to help you write a better program, but in no way do they interfere.

Instant phrases and instant subroutines are written in a program as a special kind of REM line. This is fine for many applications, but if programs are changed by loading, those features vanish. So a KEY command has now been included in MOREPOWER.

You can define any key you wish (I normally use CTRL-somekey) to do something for you. There are three variations on the theme. It can print on the screen and you push return to execute. Or it can be like an instant subroutine, where just pressing the two keys makes it go to work. Or it can be a phrase and the computer pushes its own carriage return (honest!) and it is done. Or it can call a machine code routine and cleanly come back.

KEY definitions live at top of the memory, out of the way. POWER keeps track of them. Changing the top-of-memory pointer does not clobber anything.

One of the features of the instant subroutines I like is that whether defined as REM macros or KEY macros, upon finishing the job the cursor stays in place without mangling the screen saying READY. This is vital if, for instance, you need to preserve a screen display for some reason. The subroutine code can be either BASIC or machine code. POWER doesn't care, it just helps you use it.

MOREPOWER includes all the familiar disk function commands: disk directory listing on the screen, sending disk utility commands (scratch, replace, initialize and so on) and interrogating the error channel. The syntax is not quite the same as the DOS-support commands, but is straightforward anyway. These commands are missing from BASIC2 systems, such as the Commodore 64, so programs such as MOREPOWER are vital in filling the void.

MOREPOWER cannot only list a disk catalogue to the screen. It can list a program or a sequential file. Program listing is most useful in debugging as you can easily check a previous version of a program. If what you see on the screen pleases you, if it has relevant line numbers and so on, it can be entered into a program the lazy way: pushing the return key. I don't know if Brad intended for us to be that lazy, but that's what I do.

Being able to read a sequential file without having to write a file-reading routine is a timesaver both in debugging files and in being able to quickly look at what has accumulated on the disk.

MOREPOWER can also perform a true MERGE of a program file: subroutines, data lines or whatever you need. This feature was described above.

MOREPOWER permits flipping the commands from one disk device to another. It permits a LOAD/RUN sequence.

For those who hate to type in quotes, the important disk commands are available by pushing just one key, such as FUNCTION-2 for a directory, or FUNCTION-4 to read the error channel. Most useful! FUNCTION-8 loads and runs a BASIC program.

Finally, MOREPOWER includes an assortment of handy BASIC-Aid-type utilities such as finding the loading address of a file or its size. it can un-new a NEW'ed program, convert from hex to decimal and back and so on.

Expansion

The POWER package can grow. Just as MOREPOWER has been hooked to POWER and POWER has been hooked to the resident BASIC, so we can expect to see even more useful things if we can think of what is needed. I can't at the moment. The variety of commands already in the system covers just about all of my needs.

You may add commands you need using POWER's hooks. POWER provides a place and a careful method to hook up your own things, beyond those already permitted in the macros. You may invent your own BASIC commands and attach them to the system by the method described in the book. Hence, you can customize your computer to whatever needs you have. Bear in mind that the instant keys permit this to a great extent, and can be used via BASIC, without any knowledge of machine code. Adding new commands to the computer requires knowing some machine code, but Brad made it easy by including the assembly listing of a canned command-adder routine. Hence, even if you're just a struggling beginner you might be able to expand the system to something unique.

Documentation

The documentation provided with POWER is superb. The POWER manual is clear, concise and correct. It is delightful to read and learn from. The commands are explained in simple language. Numerous coding examples are shown. One or two forms of explanation are given so you're bound to understand them.

Learning to use POWF.R is simple, because the book is good and POWER syntax is intuitively simple. A nice side effect is that as you learn how to use POWER you pick up a good deal of information about the computer itself. Things you might have not been sure of before suddenly become clear.

Nothing is hidden. Things are easy to find. Each page is marked on top with the chapter of which it is a part, for instance "6:FINDING/ CHANGING TEXT". This is handy.

My demo version has a missing index, but the nice people at Pro-Line tell me that it will be back in the book in no time. Anyway, the table of contents is comprehensive enough that it is not a problem.

The book was written by Jim Butterfield after his use of the POWER chip for several years. This is good. The manual is based on actual working of the program rather than a bunch of abstract spec sheets. And, of course, it shows in the contents of the book.

The manual contains a list of a few things that might go wrong in the system-POWER interaction. Since POWER adds to the invariant, existing configuration, occasionally strange things may pop up. There is nothing unusual in that. What is unusual is to be told exactly what to expect and how and why it happens. This makes the book a strong and useful reference.

One of the unique features of the book is the documentation of the POWER code itself. Much of the information is quite technical and intended for people with knowledge of the Commodore 64 system and machine code. But even if you don't know much machine code, it is worthwhile to read this section. There is much to be learned—a necessary knowledge if you plan to expand your computer's incredible capabilities even further.

First of all the key addresses where such expansion can be made are listed and several hints of how to do it are provided. Secondly, some of the code in POWER has been written as subroutines. These subroutines are documented in the book. What they need and what they return is clearly listed, offering a gold mine of useful utilities to be used in programs or your extensions of POWER.

PAL and POWER

Since POWER is written by the same person (Brad Templeton) as an assembler called PAL, several features useful to PAL and machine language programming are documented, since the systems coexist together and can share routines. Specifically, all commands that help BASIC programs also are used to edit PAL programs. The subsystem load/run sequence is valuable in the PAL environment. If you have, and use, PAL you will find POWER to be a nice addition.

Conclusion

POWER is here to stay. POWER is soundly built, works well and the manual is excellent. POWER has proven itself in several years of use to be an asset.

A magnificent machine such as a Commodore 64 is greatly enhanced by the presence of POWER. I can do no more than recommend to everyone I know to get and use POWER. It is a great system. **C**

Home Computer Meteorology
With the Home Automatic Weather Station

by Steve Finkel

Maybe you can't do anything about the weather, but with the Home Automatic Weather Station and your Commodore 64 or VIC 20, you stand a good chance of at least knowing what the weather will be doing.

Television weatherpeople often deliver the weather forecast with corny one-liners, annoying small-talk and other behavior that would be viewed as peculiar or unpleasant in the real world. Have you ever wondered whether it would be raining in the afternoon, turned on the radio and had to suffer through a barrage of commercials, sports scores, traffic reports and every tidbit of information other than the weather? HAWS (Home Automatic Weather Station) will allow you to turn the sound down on obnoxious weathermen and endless radio chatter, confident in the knowledge that you are able to derive weather information on your Commodore system.

An unknowing public has been duped for too long by fast talking, overly technical meteorologists into believing that they are unable to deal with the intricacies involved in understanding and predicting weather. HAWS can help transform you from spectator to weather maven without any meteorological training! HAWS is a sensory unit that plugs into the back of your Commodore 64 or VIC 20, with software for either machine available on cassette or disk. The HAWS sensory unit measures the temperature, humidity and air pressure, and the software translates the input into useful, easy-to-comprehend weather information.

HAWS comes with a plain-language manual that explains the terminology in straightforward terms. The software presents the user with a menu that specifies nine different functions. Once the HAWS unit is installed (plug it in the left rear control port and choose "LOAD HAWS" in the menu), you are able to obtain information based on the parameters of temperature, barometric pressure, wind, humidity, cloud formation and precipitation. All these factors and their effects are explained in the manual.

The menu provides several other options in addition to LOAD HAWS. One function the menu lists is the DISPLAY program. DISPLAY gives continuous readings of temperature, relative humidity and air pressure, as well as calculating the dew points. These values are updated on the screen display every 15 seconds. COMFACTOR determines the comfort factor, which is based on temperature and humidity. The comfort factor is a measure of suitability of climate, based on statistics of human judgement of weather conditions.

The CLOUDALT option estimates the altitude of cumulus clouds, determined from temperature, relative humidity and dew point values. CHILLFACTOR calculates the wind chill factor, measuring wind speed and temperature. CALC is a program that functions as a calculator (the HAWSCULATOR), translating temperature and pressure values from English to metric and vice versa. The HAWSCULATOR also determines the dew point from temperature and relative humidity values.

The TREND program allows you to collect temperature, pressure, relative humidity and dew point values over a period of 1 to 36 hours and store the data on disk or tape. These values are very useful in understanding and predicting the weather. The GRAPH function plots and analyzes the data gathered with TREND. The weather values of temperature, pressure, relative humidity and dew point are plotted on four separate graphs. Along with each graph, numerical values are given for average, mean, high and low data values.

The PRINT command reorganizes the data from TREND and allows those with printers to obtain a printout of the data. The FORECAST function uses the information from TREND to make a weather prediction, presenting the data in an amusing slot machine format. Each function on the menu is presented in a different manner. Some, such as the calculator representation of CALC and the slot machine in FORECAST, are fun to use as well as informative.

A dimension of the HAWS program that increases understanding of the weather and is fun to play around with is inputting data values not derived from the HAWS unit. This allows the user access to the functions using any data values, to better understand the relationship between weather functions, to determine weather in other parts of the world (how hot IS it in New Zealand right now, anyway?) or just have fun "creating" one's own weather conditions.

HAWS also offers the user the ability to create additional weather programs. HAWS was developed by Vaisala, Inc. and is available for $199.00 with cassette or disk software for either the VIC or the 64. Whether you're a meteorology student, or someone who just wants to know if he should wear a sweater tomorrow, HAWS makes knowing and understanding the weather as clear as a cloudless day. **C**

new books

From: International Council for Computers in Education

University of Oregon
1787 Agate Street
Eugene, OR 97403

Three booklets for educators:
Computer Literacy Activities for Elementary and Middle School Students.
LOGO in the Classroom.
Computer Metaphors: Approaches to Computer Literacy.

From: Alfred Publishing Company, Inc.

Sherman Oaks, California

The Personal Computer Glossary by George Ledin, Jr. $2.95

From: Chilton Book Company

Radnor, PA 19089
215-964-4000

Handbook of Computer Applications for the Small or Medium-sized Business by Howard Falk. $19.95
First Family Computer Book by Ed and Stevie Baldwin. $17.95
Solving Marketing Problems Using VisiCalc by Cochrane Chase *et al.* $15.95
First Byte: Fundamentals of Microprocessing by Lyn and Robert Taetzsch. $19.95

From: Birkhauser Boston

380 Green Street
Cambridge, MA 02139

An Introduction to the Commodore 64 by Nevin B. Scrimshaw and James Vogel.
The Commodore 64 Music Book by Nevin B. Scrimshaw and James Vogel.
Discover Your VIC 20 by David Kahn, Jr. and Nevin B. Scrimshaw. $10.95
The Commodore Puzzle Book by Gordon Lee and Nevin B. Scrimshaw. $7.95

From: Garland Publishing, Inc.

136 Madison Avenue
New York, NY 10016

A Dictionary of Minicomputing and Microcomputing by Philip E. Burton. $17.95

(continued on page 124)

new books

From: New American Library

1633 Broadway
New York, NY 10019
212-397-8000

The Computer Phone Book (a guide to personal computer networks) by Mike Cane. $9.95

From: Osborne/ McGraw Hill

2600 Tenth Street
Berkeley, CA

Your Commodore 64 by John Heilborn and Ran Talbott.
Commodore 64 Fun and Games by Ron Jeffries, Glen Fisher and Brian Sawyer.

From: Micro Text Publications, Inc.

Prentice-Hall, Inc.
Englewood Cliffs, NJ
07932

Using the Commodore 64 in the Home by Hank Librach and Bill L. Behrendt.

From: Microcomputer Applications

827 Missouri Street
Fairfield, CA 94533

High-Tech Consulting: A Guide to Making Money as a Computer Consultant by John Zarrella. $18.95

Commodore Bestsellers at B. Dalton Bookstores

See your local B. Dalton bookstore for these popular titles. All are in the top twenty best-selling computer titles at Dalton.

Your Commodore 64 from Osborne/McGraw Hill
Teach Computer Programming on the Commodore 64 from David McKay
Commodore 64 Programmer's Reference Guide from Howard Sams, Inc.
How to Use the Commodore 64 from Alfred Publishing
Commodore 64 Users Handbook from Weber
Commodore 64 Games Book from Distributors
Commodore 64 Reference Card from Nanos
More Than 32 BASIC Programs for the Commodore 64 from dilithium Press
101 Programming Tips and Tricks for the VIC 20 and Commodore 64 from Arcsoft
Cosmic Games for the Commodore VIC 20 from Addison-Wesley **C**

that does not compute...

"Programming Multiple-Voice Music in Machine Language"

Issue 26 (October/November, 1983)

On page 24, line 2330 needs to be fixed. It should read:

```
2330   LDA FRQTAB+1,Y
```

"Getting the Most Out of (and Into) Your Disk Drive"

Issue 27 **that does not compute...**

The corrected program in this issue needs one correction. Line 20 should be:

```
20 PRINT "[CLEAR]";
```

"Millions and the Microchip"

by Jim Gracely
Issue 27

I want to thank Royal E. Dunlap of Cypress, Texas, for being the first to let me know that I don't know how many ounces are in a pound of gold. Spending most of my time around computers and not precious metals, I didn't even realize that an ounce of gold is really a troy ounce. Of course (?) there are only 12 troy ounces in a pound, and therefore only 48000 ounces in two tons of gold: At $395 an ounce, this is $18,960,000. So, if you took my quiz in Issue 27 ("Millions and The Microchip") of this magazine, C was the correct answer (and not A as given).

Now, for my rebuttal. Because I said only "two tons of gold", and not troy tons, I have a way to get out of this mess. A troy ounce is 480 grains. If gold is $395 an ounce (troy) it is $.8229 a grain. How many grains are in two tons? Well, let's start with the fact that there are 437.44 grains in an ounce (avoirdupois). We multiply by 16 (this time I'm sure) to get pounds (6999 grains). Two tons is 4000 pounds (27,996,000 grains). At $.8229 a grain, this comes to $23,037,908. If you will allow me this curveball, the correct answer to the quiz is still A. **C**

new products

The following information is taken from new product announcements sent to us by independent manufacturers and is provided only to help keep our readers abreast of developments. Commodore does not endorse any of the products mentioned, has not tested them and cannot vouch for their availability. If you have any problems with any of the products listed here, please write to us.

Company:

Micro-Ed, Inc.
P. O. Box 24156
Minneapolis, MN 55424
800-Micro-Ed

Product:

Inexpensive Laser Disk Controller—For the Commodore 64 and Pioneer Laser Disk player. An inexpensive interface allows the Commodore 64 to control the functions of a laser disk player and TV monitor display through a series of POKE statements. Uses a single monitor for both the laser disk image and the image from the 64. Right now Micro-Ed is developing educational software using this unique technology.
Price: Contact company.

Company:

MicroSpec, Inc.
P. O. Box 863085
Plano, TX 75086
214-867-1333

Product:

Rental Property Manager—For the Commodore 64 and single disk drive. For use by those who own or operate rental homes or apartment complexes. The system allows the user to maintain up to 200 unit records on a disk. Extensive data on each unit is included in each record. A printer setup function allows the user to produce hard-copy reports.
Price: Contact company.

Company:

Basic Byte, Inc.
c/o Software Author's
Services Intl.
13108 Ludlow
Huntington Woods, MI 48070
313-545-6779

Product:

Portfolio Manager: Volume 1, Stock Management—For the Commodore 64 or VIC 20. An investment management program for home use that provides an easy method of recording and computing stock transactions.
Price: $29.95.

Company:

LAWCO Ltd.
P. O. Box 1337
Cupertino, CA 95015
408-733-0739

Product:

BEC™—Break-even computations for use on the Commodore 64 or VIC 20. Computes the break-even point for new products. Data can be saved and updated. Multiple break-even points can be computed by using different variables. Allows printed reports.
Price: $79.95 tape; $89.95 disk.

Company:

Cosmopolitan Software
Services Ltd.
Box 953
Dartmouth, N.S. B2Y 3Z6
Canada
800-268-6364
(B.C.:112-800-268-6364)

Product:

TAXPACK™—Canadian income tax calculation for use with a VIC 20. Guides the user through the 1983 Canadian T1 General tax form.
Price: $29.95 tape.

Company:

Kobetek Systems Ltd.
1113 Commercial Street
New Minas, N.S. B4N 3E6
Canada
902-678-9800

Product:

SPP (Statistics for Personal Computers)—For Commodore's 4000, 8000, and 64 series computers. Includes descriptive statistics; cross-tabulations; correlations; linear, cubic and quadratic regression; multiple regression and more. Allows data editing, transformation, election and manipulation. Data can be displayed as histograms, scatter plots, normal probability plots, box diagrams and tables.
Price: Ranges from $350 to $500 for complete packages.

Company:

Savergy, Inc.
1404 Webster Avenue
Fort Collins, CO 80524
303-221-4200

Product:

Computer Interface Module 112—Computerized home controller for the Commodore 64 or VIC 20. Regulates lights, appliances and energy use. Package

includes interface cartridge, wall-mount switching unit and software on disk or cassette.
Price: $450.

Company:

Soltech
1538 Ohio Avenue
Virginia Beach, VA 23454
804-425-7792

Product:

Passive Solar Design Program for Homeowners—For use with the Commodore 64. Helps a user with little or no design experience design a solar addition to a building by calculating fuel savings, useful heat, heat losses, ventilation needs, amount of heat storage material, night insulation and more. Allows the user to change variables to see what effect it has on heat output and money saved. Creates graphs for easy interpretation of results. Provides a 30-year financial analysis that considers fuel inflation, solar tax credits, interest rates, etc.
Price: $99.95 on tape or disk.

Company:

SLED Software
P. O. Box 16322
Minneapolis, MN 55416
612-926-5820

Product:

Educational Software for Learning Disabled—For use with PET and Commodore 64 computers. Individualized instruction in spelling for students with specific language learning disabilities or anyone with spelling deficits. Covers five basic spelling rules in 26 lessons, providing repetitive exercises and reinforcement.
Price: Complete 26-lesson package on disk or tape $199. Programs also available individually.

Company:

Sierra Digital Research, Inc.
100 Washington Street,
Suite 104
Reno, NV 89503
702-323-3856

Product:

Three Useful Hobby Programs—For CBM 8032 or Commodore 64. *Mineralogy Pacs I* and *II* help the user identify minerals. Given identification parameters such as color, streak, hardness, fracture, etc., the programs search a data base and report the names of minerals matching the input data. *Small Arms Ballistics Parameters* for target shooters and handloading hobbyists generates a complete ballistics table from data supplied by the user. This program requires an 80-column printer.
Price: $50 each on tape or disk (2031/1541).

Company:

Generex, Inc.
P. O. Box 1269
Jackson, NJ 08527

Product:

Two Educational Programs—For the VIC 20 with 16K expansion. *Run for Office* uses a two-player game format to test the students' knowledge of the presidents. Whichever player knows the most is elected president. *Espanol* is a teaching aid program for Spanish language students. The teacher enters the words to be learned into a permanent file that is then used in the practice sessions of the program.
Price: Run for Office $19.95 tape; Espanol $24.95 tape.

Company:

Smoky Mountain Software
54 West Main Street
Brevard, NC 28712
704-883-2595

Product:

Bible Fun 1—Nine Bible programs for the Commodore 64 include *New Testament Jobs, Old Testament Jobs, Bible Mates, Old Testament Guess Who, Gospels Guess Who, Acts Guess Who, Books of the Bible, Hidden Words,* and *Bible Trip.*
Price: $95 disk.

Company:

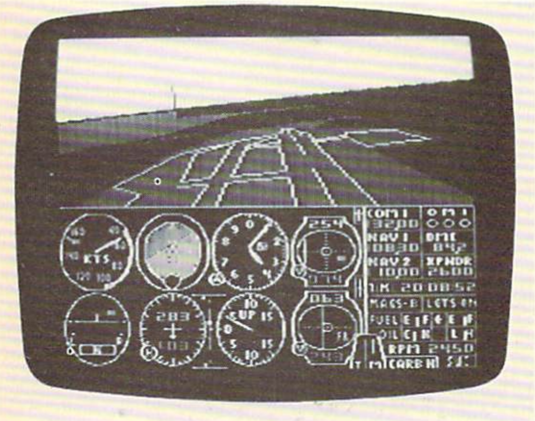
Academy Software
P. O. Box 9403
San Rafael, CA 94912
415-499-0850

Product:

IFR Flight Simulator—For the VIC 20. Designed for everyone from 13 year-olds to experienced pilots. A realistic simulation of instrument flight in a light plane written by a private pilot.
Price: $39.95 cartridge.

Company:

SubLOGIC Corporation
713 Edgebrook Drive
Champaign, IL 61820
800-637-4983



Flight Simulator II

Product:

Flight Simulator II—For the Commodore 64. Simulates the controls of a Piper 181 Cherokee Archer with full flight instrumentation and realistic panoramic view. Let's you practice take-offs, landings and aerobatics. Outside scenery and weather conditions are user-adjustable.
Price:$49.95 disk.

Company:

Computer Management
Corporation
724 Church Street
San Francisco, CA 94114
415-431-8747

Product:

BridgePro—For the Commodore 64. Allows one or two people to play the card game of bridge. BridgePro shuffles, deals, bids and plays the other hands. Features random card dealing, machine language speed, replay of hands and complete Contract Bridge scoring. Includes six different new play options. Score can be saved at the end of any hand and play continued at another time.
Price: $35 disk.

Company:

Matrix Software
315 Marion Avenue
Big Rapids, MI 49307
616-796-2483

Product:

*Astro*Talk*—For PET and Commodore 64 computers. Creates an accurate personal horoscope and explains what the stars are saying. Menu format allows the user to ask questions about specific areas of interest such as career, love life, etc. Additional features include a "Student Astroguide" that acts as a tutor for beginners and introductory material for those who are new to astrology.
Price: $39.95

Company:

Micro Information Processing
Systems
6810 Seagull Lane, Suite N
New Orleans, LA 70126
504-242-4128

Product:

Musical Greeting Cards—Each

COMPUTER ASTROLOGY
Astro*talk
Astro*talk
By
Matrix software
User-Friendly Computer Astrology for the Apple II, IBM PC, PET & Commodore 64
$39.95

new products

card contains a tiny microprocessor that plays a tune when the card is opened. Cards are available for Christmas, birthday, anniversary, get well and special friendship.
Price:$5 each or $48 per dozen.

Company:

Compucards
P. O. Box 894
Stone Mountain, GA 30086
404-299-0713

Product:

Continuous Feed Greeting Cards—Designed so your personal message can be printed inside using your word processor and printer. Cards have continuous tab feed with a clean-edge perf. Available for Christmas, birthdays and various other occasions. Artwork by nationally known artists.
Price: 20 cards and envelopes $9.95; 100 cards and envelopes $45; 300 cards and envelopes $130.

Company:

The Wizards
P. O. Box 7118
The Woodlands, TX 77387

Product:

Omnipotus—Game of thought for the Commodore 64. The player assumes the role of a computer and the computer becomes Professor Omnipotus, your creator. You have just returned from a "compare and categorize" mission in the outside world and the "professor" asks you questions about what you have observed. The program introduces the user to the basics of philosophical thinking and sometimes provides insights into the world you live in.
Price: $13.95 tape or disk.

Company:

Don's Designs
1728 Womer
Wichita, KS 67203
316-942-4477

Product:

Design-A-Quilt—For the VIC 20. Automatically designs over 450 million patterns suited for quilts, wallpaper, fabric and graphics.
Price: $29.95 tape.

Company:

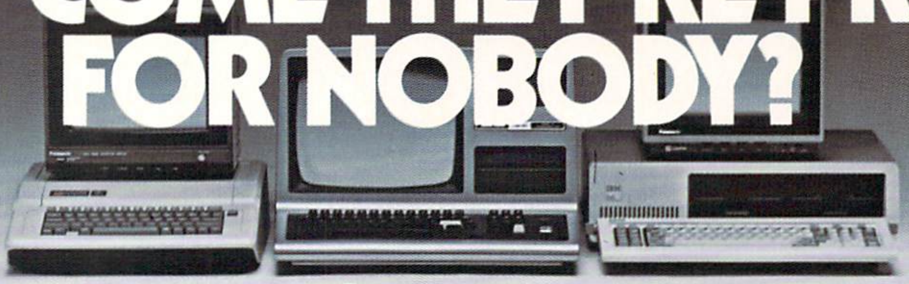
F/22 Press
P. O. Box 141
Leonia, NJ 07605
201-568-6250

Product:

Darkstar™—For the Commodore 64. Solves problems commonly encountered in the photographic darkroom. It works with all black-and-white and color materials to provide exposure data for changes in print density, magnification, lens opening, paper type and more. It also adjusts color-printing filter-pack composition to correct print color balance, eliminate neutral density and compensate for color paper emulsion changes and performs other calculations necessary to darkroom work.
Price: $49.95 tape. **C**

advertisers' index

IF PERSONAL COMPUTERS ARE FOR EVERYBODY, HOW COME THEY'RE PRICED FOR NOBODY?



$1395* APPLE® IIe 64K

$999* TRS-80® III 16K

$1355* IBM® PC 64K

A personal computer is supposed to be a computer for persons. Not just wealthy persons. Or whiz-kid persons. Or privileged persons.

But person persons.

In other words, all the persons whom Apple, IBM, and Radio Shack seem to have forgotten about (including, most likely, you).

But that's okay. Because now you can get a high-powered home computer without taking out a second mortgage on your home.

It's the Commodore 64. We're not talking about a low-priced computer that can barely retain a phone number. We're talking about a memory of 64K. Which means it can perform tasks most other home computers can't. Including some of those that cost a lot more. (Take another look at the three computers above.)

By itself, the Commodore 64 is all the computer you'll ever need. Yet, if you do want to expand its capabilities some day, you can do so by adding a full complement of Commodore peripherals. Such as disk drives. Modems. And printers.

You can also play terrific games on the Commodore 64. Many of which will be far more challenging than those you could ever play on a game machine alone.

And as great as all this sounds, what's even greater-sounding is the price. It's hundreds of dollars less than that of our nearest competitor.

So while other companies are trying to take advantage of the computer revolution, it seems to us they're really taking advantage of something else:

Their customers.

*Manufacturers' suggested list prices.
Monitor included with TRS-80 III only. Commodore Business Machines–P.O. Box 500R, Conshohocken, PA 19428;
Canada–3370 Pharmacy Avenue, Agincourt, Ont., Can. M1W 2K4.

commodore
COMPUTERS

Apple is a registered trademark of Apple Computer, Inc.
TRS-80 is a registered trademark of Tandy Corp. IBM is a registered trademark of International Business Machines Corp.

SATURN
ORBIT
WEIGHT
AGE
RADIUS
MOONS
DAY
YEAR
SONY
COMMODORE 64
SOFTWARE
THE COMMODORE 64.™ UNDER $300.*
You can't buy a better computer at twice the price.

ISBN 0-88731-006-0