The best games, applications, and utilities ever published for Commodore home computers by COMPUTE! and COMPUTE!'s Gazette.

Best of COMPUTE! & GAZETTE

$14.95
1988
$20.95 Canada 02707
Display until Jan. 29, 1988

Disk Inside

For Commodore 64 & 128

68 Of Our Best Programs Ever INSIDE ON DISK!

0 71486 02707 2 85

WHEN YOU'VE SOLD YOU MUST BE DOING

No Brag, Just Fact! Over 1,800,000 programs sold to date—and each program includes:

- **Free Customer Technical Support (For all *registered* users)**
- **A Money Back Guarantee (If you can find a better program, we'll buy it for you)****
- **A very liberal Upgrade & Exchange Policy (Which means you never have to worry about obsolete software)****

S.R.P. C64—$49.95
C128—$69.95

Word Writer 3™

Word Writer 3 has more features, more power, and is easier-to-use than any other C64 word processor!

Features:

- **An 85,000-Word Spell Checker**—plus, unlimited sub-dictionaries.
- **An Integrated Thesaurus** with over 60,000 synonyms and alternatives.
- **An Integrated Outline Processor** that quickly organizes notes, facts, and ideas into a convenient outline format.
- **An 80-Column Print Preview Mode**
- **Highlighting:** Prints out your text incorporating underlining, **boldface**, *italic*, superscript, subscript, and more.
- **Headers and Footers**
- **Automatic Program Set-Up:** Configures WORD WRITER 3 to your choice of printer codes, screen colors, and more.
- **SwiftKeys*** access commands quickly, using a minimum of keystrokes.
- **Compatible with GEOS***

WORD WRITER 3, DATA MANAGER 2, and SWIFTCALC INTERFACE TOGETHER FOR A COMPLETE PRODUCTIVITY SYSTEM!

The Critic's Choice

"SYLVIA PORTER'S PERSONAL FINANCIAL PLANNER is a class act from start to finish."
Run Magazine

"SWIFTCALC is an excellent product at an outstanding price."
Commodore Magazine

"Using WORD WRITER is an absolute joy."
Run Magazine

"PARTNER may become the most valuable utility you have…Timeworks has another winner, and deservedly so."
Compute's Gazette

"DATA MANAGER 2 is easy to use, well documented with an excellent tutorial…and, powerful enough to make life easier for many homes and small businesses."
Ahoy!

S.R.P. C64—$49.95
C128—$59.95

PARTNER 64™

A cartridge-based product with eight instantly accessible, *memory-resident* desktop accessories.

Accessories include:

- **Appointment Calendar & Date Book**
- **Name, Address, and Phone List**
- **Auto Dialer** • **Memo Pad**
- **Label Maker & Envelope Addresser**
- **Calculator** • **Typewriter**
- **Screen Print**

Other features include:

- **SwiftDos:** Allows you to access Commodore disk drive commands any time.
- **SwiftLock:** Enter your secret code before you leave your computer. The keyboard will be locked until you re-enter the code.
- **Extended Printer Control:** Enables you to send commands directly to your printer at any time.
- **SwiftLoad:** Allows your Commodore 1541 disk drive to load as fast as the 1571 disk drive.

S.R.P. C64—$39.95
C128—$69.95

Data Manager 2™

A highly flexible filing and recordkeeping system that stores, retrieves, sorts, evaluates, and updates large amounts of information.

Features:

- **A Report Writer:** Generates customized data reports. You specify the title, location, and sequence of each column.
- **A Label Maker:** Prints your name and address file onto standard mailing labels, and transfers and prints text information onto labels and tags.
- **Quick Access to important information:** Retrieves and prints items by name, date range, index code—or any category stored in the system.
- **Calculates numerical data from column to column and field to field:** Allows you to perform spreadsheet tasks such as payroll calculations, cost estimates.
- **Compatible with GEOS**

*Reg. trademarks of Commodore Electronics, Ltd., Berkeley Software, Inc., Timeworks, Inc.

1,800,007 PROGRAMS, *SOMETHING* RIGHT.

SYLVIA PORTER'S **Personal Financial Planner**™

All the computer tools you'll need to manage your money on a day-to-day basis, and plan your financial future, too!

S.R.P. C64–$49.95
C128–$69.95

For Your Day-to-Day Affairs:

- **Maintains your electronic checkbook** and credit card transactions.
- **Writes your checks** and balances your checkbook.
- **Prepares and monitors your budget**
- **Classifies and tracks your taxable income and expenses**
- **Calculates your net worth and generates customized personal financial statements**
- **Tracks your financial assets**—and your insurance policies.

For Your Financial Future:
Leads you step-by-step through a series of questions regarding your financial goals, and your current financial condition. Your answers will enable your computer to determine and print a summary of the amounts you must save each year to meet your financial objectives—in both real and inflated dollars.

SwiftCalc™

A powerful, easy-to-use electronic spreadsheet designed for home and business use.

S.R.P. C64–$39.95
C128–$69.95

Features:

- **Sideways:** Prints all your columns on one, continuous sheet...sideways.
- **250 rows and 104 columns** provide more than 25,000 cells (locations) in which to place information.
- **Performs mathematical functions,** up to 12 digits. Allows the use of minimum and maximum values, averages, sums, integers, absolute values, and exponential notation.
- **Performs financial analysis functions,** calculates the present and future value of a dollar and the present and future value of a constant amount (annuity).
- **SwiftLoad:** Allows your 1541 Disk Drive to load up to 5 times faster.
- **Produces "What If?" Reports:** Applies mathematical functions and algebraic formulas to any spreadsheet-type analysis, such as budgeting, financial planning, or cost estimating. Calculates these formulas and produces complicated "What If?" Reports at the press of a key.

The Timeworks Desktop PUBLISHER™

Where you once needed a typesetter, designer, and paste-up artist, you can do it all yourself with THE TIMEWORKS DESKTOP PUBLISHER. Includes everything you need to produce professional-quality printed documents on your C64 computer.

S.R.P. $59.95

Features:

- **Automatic Page Style Formatting:** Set up a page style only once. The computer follows it automatically, inserting rules, headers, and footers.
- **A Full-Featured Word Processor**
- **A Wide Variety of Font Styles in Multiple Sizes**
- **A Built-In Text Editor** that allows you to write headlines, taglines, and captions to fit any space.
- **High Resolution Graphics:** Imports illustrations and graphics from leading graphic programs.
- **Automatic Kerning:** Opens or tightens word and letter spacing for a truly professional look.
- **A Wide Selection of Built-In Patterns, Textures, and Shades**
- **A Complete Set of Built-In Drawing Tools:** Draw lines, boxes, circles, and polygons automatically—or, draw free hand.
- **Use THE TIMEWORKS DESKTOP PUBLISHER to produce:** Newsletters, Brochures, Flyers, Forms, Reports, Bulletins, Menus, Certificates, Letterheads, Newspapers—the possibilities are endless!

Available at your favorite dealer, or contact Timeworks today.
To Order Call: **1-312-948-9202**
For Update Information Call: **1-312-948-9206**

TIMEWORKS®

MORE POWER FOR YOUR DOLLAR

Timeworks, Inc. 444 Lake Cook Road,
Deerfield, Illinois 60015 312-948-9200

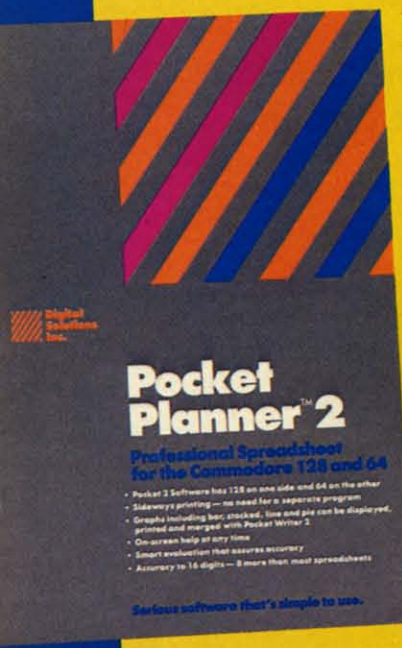
**Details on every Timeworks package.

These programs are also available for C128 (128K, 80 column)

"...one of the very best*..."

Write • File • Organize • Analyse • Track • Compare • Search • Graph • Report • Create • Sort
Solve • Calculate • Integrate • Express • Cut & Paste • Modify • Proof • Display • Report

You can do almost anything.



You can do it. You really can.

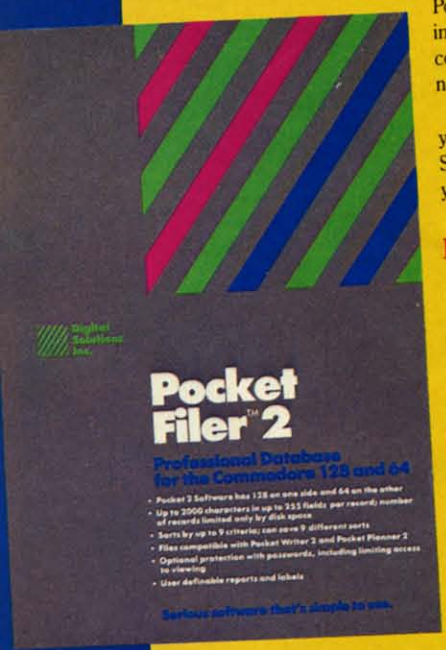
These are all of the wonderful things you can do with your Commodore 64 or 128 and Pocket™ 2 Software. This is the software that can make you an ace in 30 minutes . . . *even if you've never touched a computer before.*

The reason? Pocket™ 2 Software is menu driven. It has all of the help and instructions you need right on the screen. All of the power of your computer is right there at your fingertips. It's such an exhilarating feeling, you might find yourself humming the theme to 'Rocky'.

Fast, flexible and affordable.

At Digital Solutions, we're committed to making our software work for you. Naturally, all Pocket™ 2 software is fully integrated and GEOS™ compatible. It's people logical . . . not just machine logical.

All those wonderful things you've heard about Pocket™ 2 Software are true. Just wait until you get your hands on it.



Digital Superpak™ 2 . . . A super idea

One of the beauties of Pocket™ 2 Software is that all three programs work beautifully together. Working together, they're more powerful and give you a whole new range of options and possibilities.

Not only that, when you order Pocket Writer™ 2, Pocket Planner™ 2 and Pocket Filer™ bundled together in Digital Superpak™ 2, you can save a bundle of money. Just **$99.95** U.S.

Pocket Writer™ 2 The Award Winner

Experts call Pocket Writer™ 2 the best word processor ever. With its menu-driven, on-screen instructions, you'll fly through letters, presentations, memos, lists, reports and articles. In fact, if you need to write it, Pocket Writer™ 2 will make it easier. You can format your words the way you want them. You can change your mind. You can even stay on top of your spelling with Spelling Checker. Only **$59.95** U.S.

Pocket Filer™ 2 . . . like a giant file drawer

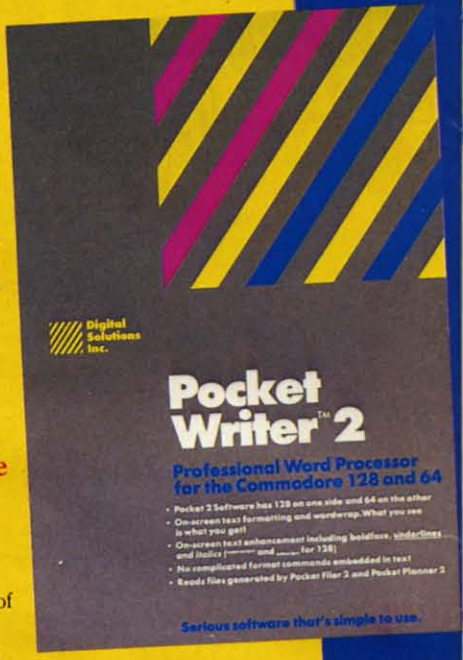
Pocket Filer™ 2 is a full-featured database that will help you to keep track of just about anything. No matter what kind of information you have to store, Pocket Filer™ 2 will help you to organize it better, process it in more ways and retrieve it faster when you need it.

At home or at the office, now you can organize information the way *you* want it. Only **$59.95** U.S.

Pocket Planner™ 2 It figures

Pocket Planner™ 2 is the answer for number crunchers. This powerful electronic spreadsheet is the right tool for collecting, storing, analysing and reporting information . . . financial or numerical. It's a common-sense solution for managing your checkbook, budget, installment payments, worksheets or investment portfolio. Pocket Planner™ 2 lets you put some real bite into your number crunching. Only **$59.95** U.S.

**Commodore Magazine, July 1987*



If you can't find Pocket™ 2 Software at your dealer, use this coupon or phone (416) 731-8775 to order.

METHOD OF PAYMENT (no Purchase Orders or C.O.D.)
Money Order ___ VISA ___ MasterCard ___ Amer. Exp. ___ Check ___

CHARGE CARD # | | | | | | | | | | | | | | | | |

EXP. | | / | |

Credit card orders must be signed

VISA, MasterCard and AMEX accepted. No shipping and handling charges. Ontario residents add 7% sales tax. Please allow 4-6 weeks for delivery.
Order yours today.

Individual Pocket™ 2 programs are just $59.95 U.S. each.
Digital Superpak™ 2 is just $99.95 U.S.
Send me Pocket Writer™ ☐, Pocket Planner™2 ☐,
Pocket Filer™2 ☐, Digital Superpak™2 ☐ Total $ _____

Name: _____
Address: _____
City: _____ State/Prov.: _____
Country: _____ Zip/Postal Code: _____

Digital Solutions Inc.

2-30 Wertheim Court
Richmond Hill, Ontario
Canada L4B 1B9
(416) 731-8775

Best of COMPUTE! AND GAZETTE

STRATEGY GAMES

ACTION GAMES

PROGRAMMING UTILITIES

GRAPHICS UTILITIES

APPLICATIONS

METABASIC

SPEEDSCRIPT

THE TURBO SERIES

Publisher	James A. Casella
Editorial Director	Richard Mansfield
Managing Editor	Kathleen Martinek
Associate Publisher	Selby Bateman
Production Director	Tony Roberts
Editor, COMPUTE! and COMPUTE!'s GAZETTE	Lance Elko
Editor, COMPUTE!'s Atari ST Disk & Magazine and COMPUTE's PC Magazine	Tom R. Halfhill
Editor, COMPUTE's Apple Applications Magazine	Gregg Keizer
Features Editor	Keith Ferrell
Technical Editor	Ottis R. Cowper
Assistant Technical Editors	Dale McBane, Jim Fuchs
Assistant Editors	Todd Heimarck, Rhett Anderson, John Shadle, Randy Thompson, Cliff Karnes
Programming Supervisor	Patrick Parrish
Editorial Programmers	Tim Victor, Tim Midkiff, William Chin, George Miller
Buyer's Guide Coordinator	Caroline Hanlon
Copy Editors	Karen Uhlendorf, Karen Siepak, Jill Champion
Submissions Reviewer	David Hensley
Programming Assistants	Troy Tucker, Joyce Sides
Executive Assistant	Debi Nash
Administrative Assistants	Julia Fleming, Iris Brooks, Sybil Agee
Receptionist	Anita Armfield
COMPUTE!'s Book Division Editor	Stephen Levy
Assistant Editors	Tammie Taylor, Robert Bixby, Lynne Weatherman
Programming Assistant	David Florance
Director of National Sales	Joseph W. Hatcher
Production Manager	Irma Swain
Art Director	Janice R. Fary
Assistant Art Director	Lee Noel, Jr.
Assistant Production Manager	De Potter
Artists	Robin Case, Kim Potts, Scotty Billings, Tony Jacobson
Typesetting	Terry Cash, Carole Dunton
Illustrator	Harry Blair
Director of Advertising Sales	Peter Johnsmeyer
Advertising Director	Bernard J. Theobald, Jr.
Production Coordinator	Kathleen Hanlon
Customer Service Manager	Diane Longo
Dealer Sales Supervisor	Jose Cruz
Individual Order Supervisor	Cassandra Green

James A. Casella, President
Richard Mansfield, Vice President, Editorial Director
Richard J. Marino, Vice President, Advertising Sales
Ilene Berson Weiner, Vice President, Production

Editorial Board
Richard Mansfield, Kathleen Martinek, Selby Bateman, Lance Elko, Tom R. Halfhill, Stephen Levy
Robert Lock, Founder and Editorial Consultant

Editorial offices:	324 West Wendover Avenue Suite 200 Greensboro, NC 27408 USA
Corporate offices:	825 7th Avenue New York, NY 10019 212-265-8360
Customer Service:	800-346-6767 (In NY 212-887-8525) 10:00 AM–12:30 PM; 1:30 PM–3:00 PM Monday–Friday
Dealer Sales:	800-638-3822 (In NY 212-887-8566) 9 AM–5 PM Monday–Friday

Advertising Sales Representatives

New England & Mid Atlantic
Bernard Theobald, Thomas Link: 212-315-1665

Midwest & Southwest
Jerry Thompson, Lucille Dennis: 312-726-6047 (Chicago), 713-731-2605 (Texas), 303-595-9299 (Colorado), 415-348-8222 (California)

West, Northwest & British Columbia
Jerry Thompson, Lucille Dennis: 415-348-8222

Southeast & International
Harry Blair: 919-275-9809

Send all advertising materials to:
Kathleen Hanlon
324 West Wendover Ave., Suite 200
Greensboro, NC 27408

Editor's Notes

Since its first issue in 1979, COMPUTE! has published programs for Commodore computers. Back then, the PET—grandfather of the VIC, 64, and 128—was the reigning Commodore machine. In the years since, we've continued to offer some of the finest software for Commodore computers available anywhere.

Collected here, in this special issue, is the best of COMPUTE! and GAZETTE—our best games, best programming utilities, best graphics utilities, and best applications programs. You'll find *SpeedScript*, an exceptional word processor; "Campaign Manager," an engaging simulation of the presidential election campaign; "Laser Beam," a fast-paced game that really tests your joystick prowess; "Meta-BASIC," a powerful BASIC-language extension that makes programming much easier; and dozens more.

Using The Programs

We recommend that you copy the programs from the enclosed disk to working disks before you run them. Then store the original disk safely away as a backup. You can use the "Unicopy" utility on the disk to quickly back up all the files.

When originally published, these programs were provided as printed listings that had to be typed into the computer before they could be used. In this collection, the programs are packed onto both sides of a 5¼-inch disk, ready to be loaded and used.

Most of these programs work on the Commodore 64 and on the Commodore 128 running in 64 mode. Several, however, are written specifically to take advantage of the 128's special features. These programs are labeled with the characters *128* on the program disk.

In general, you'll find the program names identical to the article titles. View the disk directory by entering

```
LOAD "$",8
```

and then **LIST**. If you don't see the file you're looking for, turn the disk over and repeat the process on the other side.

In most cases, there's nothing more to do but load and run the program, but do read the article that accompanies each program for specific loading instructions and an explanation of all the program's features.

We know you'll enjoy these programs. Readers of COMPUTE! and GAZETTE have been using them—and telling us how much they like them—for years.

COMPUTE! Publications, Inc.
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019

Best of COMPUTE! and Gazette for Commodore 64 and 128 is published by COMPUTE! Publications, Inc., 825 7th Ave., New York, NY 10019 USA. Phone: (212) 265-8360. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408.

Bring your Commodore® to life with a FREE modem and software!

Now you can plug your Commodore into an exclusive network that's useful, easy, fun and inexpensive. Join Q-Link now and we'll give you a free modem and software to get you started.

Enjoy **unlimited access** to a wide range of Q-Link "Basic" services for a set fee of only $9.95 a month. Get help from Commodore experts — use the Hotline to ask Commodore questions, or search the database of information from Commodore to answer your questions on the spot. Access Grolier's Academic American Encyclopedia™, the latest news and entertainment updates.

Special Q-Link "Plus" services let you expand your software library with **more than 10,000 public domain programs** — games, graphics, business and educational software and much more! Also, you can preview new copyrighted software before you buy. Get answers to your software questions from Berkeley Softworks, Electronic Arts and Activision, to name a few. Participate in full-color, multi-player games like Casino (includes Blackjack, Poker, Bingo and the Slots), Chess and Bridge. "Plus" services cost only an extra 6 cents per minute — **and your first hour of "Plus" service usage each month is free!**

Offer valid in the continental U.S. and Canada for new members only. Expires 6/30/88. Prices subject to change without notice. There is a communications surcharge for connection to Q-Link from Canada: 6 cents (U.S.) per minute from Montreal, Vancouver, Ottawa, Toronto, Quebec, Windsor, Kitchener and Calgary using the Tymnet network, and 15 cents (U.S.) per minute from over 85 other locations using the DATAPAC network.

With Q-Link, you can sharpen your computer skills... connect with people... discover new ways to get smart... save money every day... plus have a lot of fun!

No-risk guarantee. Try Q-Link without risk. If, at any time, for any reason, you are not completely satisfied, you may cancel your membership.

Order your Q-Link software and modem today by mailing this coupon to Q-Link, 8619 Westwood Center Drive, Vienna, VA 22180 or for faster service call toll-free: **1-800-782-2278 Ext. 1207**

Q-LINK RESERVATION FORM

Choose one:

☐ **I need a modem.** Start my Q-Link membership by charging me now for my first 4 months at $39.80, and send me the FREE Q-Link software and a FREE Commodore 300 baud auto-dial modem (model 1660—retail value $69.95).

☐ **I already have a modem.** Send me my FREE Q-Link software and start my Q-Link membership by charging me now for my first month of membership at $9.95.

Full name _____

Address _____
(No P.O. Boxes)

City _____ State _____ Zip _____

Home phone _____

Choose your method of payment:

☐ Please charge my credit card. ☐ Check enclosed.
☐ Master Card ☐ Visa

Acct# _____ Exp. _____

Signature _____

Call toll-free 1-800-782-2278 Ext. 1207 or mail this coupon to:
Q-Link, 8619 Westwood Center Drive, Vienna, VA 22180

Q-link℠

Chess

John Krause

Try to outwit your computer with this fast, multilevel chess game for the 64. A joystick is required.

The world was amazed, in the late eighteenth century, by a machine that had the astonishing ability to play a good game of chess. It entertained kings and queens. It defeated Napoleon, a master tactician. Hundreds of people paid to compete against it, but eventually it was revealed that a small man was hidden inside the machine.

A chess-playing machine remained only a dream until the late 1950s when the first computer chess game was played. Now, the World Computer Championship, held every three years since 1974, attracts almost as much publicity as the human championship matches. Why has there been so much interest in machines that play games?

One reason is that chess can be used to measure a computer's intelligence. Chess is easy to play, but difficult to master. So difficult, in fact, that some experts believe that a computer would have to be almost as intelligent as a human to become world champion.

Of course, another reason is that chess is just plain fun, but not if you can't find an opponent. To be an entertaining opponent, a computer chess game should be fast, easy to use, and capable of playing at several different skill levels. "Chess" has all these features and more. Although it's really no match against the best commercial chess games, it *has* managed to defeat these giants of the microcomputer chess world on rare occasions.

Joystick Input

The program is found in two parts on the disk. To start the game, enter LOAD "CHESS",8 and then RUN. The first program will automatically load the second (CHESS2) at the appropriate time. Both files must be present for Chess to work. After running the program, you will be asked to specify several play options. You can choose among five skill levels; start a new game or set up any position; play against the computer or watch it play against itself; or play either the white or black pieces. All of these options will be discussed in greater detail later, but for now, type 1 at each prompt. This puts you in command of the white pieces versus the computer on level one, the easiest level.

The first time the program is run, you need to wait a few seconds while the computer gets its brain in order. Then the board will be displayed with your pieces on the bottom of the screen and the computer's pieces on the top. You should see a frame around the square in the lower-left corner of the board (the VIC version uses a blinking square). This is the cursor which takes the place of your hand to move pieces around the board.

Use the joystick (plugged into port 2) to move the cursor atop the piece you wish to move. Press and release the joystick button. Now move the cursor to the square you want to move to and tap the button again. Your piece moves to the new square, and the computer responds almost instantly with its move.

A Spectacular Blunder

Did you make a foolish move? No problem. One of the most valuable features of Chess is the ability to change the position by adding or deleting pieces. This feature is especially useful for those of us who frequently manage to maneuver into a superior position, only to throw it all away in a single, spectacular blunder.

A piece can be deleted by positioning the cursor on the piece and pressing the space bar. To add a piece or change a piece to a different one, move the cursor to the appropriate square and press P, N, B, R, Q, or K for pawn, knight, bishop, rook, queen, or king, respectively. This will put one of *your* pieces on the square. To add one of the computer's pieces, hold down the SHIFT key while pressing one of these editing keys.

To take back a move, use the editing keys to delete your piece and put it back on its original square. Don't forget to take back the computer's move, too.

The editing feature also enables you to make special moves which cannot be made with the joystick alone such as castling and *en passant* captures. For example, castling can be accomplished by deleting the king and putting it on its new square, and then moving the rook as you normally would with the joystick. Although *you* can make these special moves, the computer will never castle or capture *en passant* because, due to their complexity, these moves were not included in its thinking routine.

Strange Chess

Although the computer will always make a legal move, it doesn't check to see that you do the same. You are free to move any of your pieces to any square without so much as a contemptuous buzz from the computer. If you're an experienced player, this shouldn't be a problem. If you're a beginner, however, you may want to familiarize yourself with the basic rules of chess lest you end up playing strange chess, a personal version which bears little

How *Chess* Thinks

You've probably heard that if a monkey sat down at a typewriter and pecked randomly at the keys for a long enough period of time, it would eventually type the complete works of Shakespeare. Theoretically, this is indeed possible—given enough time. There's the rub. At a brisk typing speed of 50 words per minute, it would take that poor monkey billions of years just to type "To be, or not to be." Nevertheless, there is power in trial and error.

The Minimax Algorithm

Substitute the monkey with a high-speed computer, and this technique becomes a practical method of imitating intelligence. In fact, it has been used with great success in the field of artificial intelligence. This program uses a popular trial-and-error technique known as the *minimax* algorithm.

The computer looks at the present board position and mentally moves the pieces through all the possible combinations of future moves and countermoves up to a certain point, say three moves ahead. For each combination, it calculates a score based on which pieces were captured during the combination. Each piece is worth a certain number of points depending on its general importance: 1 point for a pawn, 3 for a knight or bishop, 5 for a rook, 9 for a queen, and 46 for a king. (Of course, since you lose the game if your king cannot escape capture, the value of a king is actually infinite, but 46 is high enough to convince the computer that it's a bad move.)

When, in a move being examined, the computer captures an opponent's piece, the value of that piece is added to the score. Conversely, when one of the computer's pieces is captured, its value is subtracted from the score. Thus, a high score is considered good for the computer, and a low score is good for its opponent.

The task is to find the combination that represents best play for *both* sides. This combination is not necessarily the one with the maximum score, because while the computer is trying to maximize the score, its opponent is trying just as hard to minimize it. The best combination gives maximum scores during the computer's moves, and minimum scores during the opponent's moves.

After the best combination has been found, the computer's best move in the present position is simply the first move in the combination. The problem has been reduced from analyzing a chess position to finding the maximum and minimum of a series of numbers, which is much better suited to a computer.

50 Million Combinations On Level 5

Like most algorithms based on trial and error, this one requires sifting through an enormous number of combinations to find the best one. Fortunately, a few tricks can be used to reduce the combinations to a manageable number. This algorithm uses a technique called *alpha-beta cutoff*. It makes the computer search more intelligently, giving it the seemingly paradoxical ability to find the best move without looking at all the possible combinations. On level 5, for example, instead of having to search through roughly 2 billion combinations, it looks at only 50 million.

Even so, it would take BASIC from now till 1986 to generate that many combinations. That's why the algorithm is programmed in machine language. An advanced programming technique known as *recursion* (making a subroutine call itself) is used to generate all the possible combinations of moves. Capable of analyzing about 5000 combinations per second, this routine provides a moderate challenge at a reasonable playing speed.

resemblance to the real game. On the other hand, if you like to fudge a bit, the computer will make it easy. It will politely acquiesce to your most surreal moves.

When a pawn reaches the other side of the board, it's automatically promoted to a queen. If you would rather have a knight, bishop, or rook, you can easily make the change using the editing keys.

Checkmate

The computer thinks by analyzing thousands of possible moves and countermoves and choosing what it considers to be the best move based on the relative value of the pieces (see "How *Chess* Thinks"). Most positions don't have just one best move but several which are equally good, in which case the computer chooses among them at random. This random factor insures that every game will be different, and makes for varied and interesting play.

Play continues until one side is either checkmated or stalemated. The computer will then stop play and indicate which side has won.

There are a few quirks in the way the computer determines whether checkmate has occurred. On levels three through five, it announces checkmate prematurely. When this happens, the computer has determined that it's impossible to avoid checkmate on the *next* move or two, assuming both sides make the best moves.

Also, the computer doesn't know the subtle difference between checkmate and stalemate. Consequently, when stalemate occurs, it will announce checkmate although, in fact, the game is a draw. Since the computer tries as hard as it can to checkmate its opponent, it will also try to achieve stalemate, possibly forcing a draw when it could have won. Fortunately, this rarely happens because the conditions for stalemate exist only in unusual circumstances such as when one side has only the king remaining.

Also, the computer won't give you any hint when your king is in check (not checkmate). So be extra careful that you don't leave your king in check or move into check. Otherwise, your king would be in check during the computer's turn to move—a highly unorthodox if not illegal position. The computer's reply to such a position is unpredictable, but it usually announces checkmate, forcing you to restart the game.

In any case, when the computer announces checkmate, press the joystick button to start a new game. If you want to try out some of the other play options without waiting till checkmate, you can start

a new game at any time by pressing RUN/STOP-RESTORE and running the program again.

Play Options

When you choose the black pieces, the board will revolve so that you still play from the bottom. Since the player with the white pieces always moves first, you must wait for the computer to move before you will be allowed to make your first move.

If you become mentally exhausted after several bouts against the computer, give your brain a rest and watch the computer play itself. When you select this option, just set the joystick aside and sit back and watch the action. Beginners will find this feature an excellent way to learn some good strategies to use against the computer.

You don't have to begin a game from the starting position. If you choose the option to set up a position, an empty board will be displayed and you can use the editing keys to place pieces on the board in any position. When the position is set up, the computer will start thinking after you make your first move.

This feature is especially useful for continuing a previous game or creating a problem for the computer to solve. It also allows you to experiment with hypothetical or downright ridiculous positions. Live out your fantasy by giving yourself ten queens versus the computer's lone king. The position doesn't even have to be a legal one. You could invent your own type of chess by giving each side two kings, for example, although the computer may get confused trying to determine when checkmate has occurred.

One of the advantages of a computer opponent over a human is that you can tell the computer exactly how hard you want it to try to beat you, and it will obediently play at that level of difficulty. This is important because it's no fun if you always lose or always win effortlessly.

You have five skill levels to choose from. The difference between one level and another is the number of moves ahead that the computer looks. On level 1, for example, it looks two moves ahead (its move and your reply). Each succeeding level looks ahead one more move than the previous level.

Alas, the smarter play on the higher levels doesn't come without a price. The further ahead the computer looks, the more moves it must examine and, hence, the longer it thinks. The thinking time varies greatly depending on the level (about one second per move on level 1; about two *hours* on level 5).

Here's a rundown of the five levels:

Level 1: Beginner. Thinking time: one second. Look ahead: two moves. Fast but dumb.

Level 2: Intermediate. Thinking time: five seconds. Look ahead: three moves. Provides a reasonable challenge for impatient players.

Level 3: Tournament. Thinking time: two minutes. Look ahead: four moves. Since the usual time limit for tournament play is 40 moves in two hours, an average of three minutes per move, this level is best suited for serious players.

Level 4: Mate in two. Thinking time: 30 minutes. Look ahead: five moves. Capable of solving most mate-in-two problems.

Level 5: Postal chess. Thinking time: two hours. Look ahead: six moves. Simulates postal chess games where there is no time limit. Can avoid checkmate in two moves.

The thinking times given here are average times. The actual time ranges from half to twice the average time depending on the position.

Level 4 can be used to solve mate-in-two problems such as those published in many newspapers. Just select the following options: level 4, set up position, computer versus itself. Enter the position using the editing keys, and then make a do-nothing move by positioning the cursor over a white piece and pressing the joystick button twice. After several minutes of deep thought, the computer should respond by moving one of the white pieces (the solution) and announcing checkmate. The only mate-in-two problems that the computer cannot solve are those which involve castling, *en passant* captures, or pawn promotion. ©

Power Poker

Jud Bleser

This is one of the most addictive games we've ever published. It adds a new dimension to the traditional game of poker, and is fun for one player or in competition with friends. For the Commodore 64.

Poker is a game that's just as popular today as it was a century ago. Even though there's always a random element at play, it usually requires careful thought and a knowledge of probability. The variations created by the cards you're dealt and how you arrange them makes poker unpredictable. "Power Poker," written for the Commodore 64, adds a new twist to the game—rather, another dimension.

A Double Purpose

Think of this game as two-dimensional poker. You play on a five-by-five grid and try to make the hands that gain the most points. Each card serves two hands, so placement must be done carefully. (If you're new to poker, see "Poker Hands.")

When you run the program, you'll see a table of the number of points awarded for each kind of hand. After a pause of a few seconds, you're ready to begin. A five-by-five grid is displayed, each position identified by a letter A–Y.

The computer randomly selects a card and displays it. Place it in the grid by pressing the appropriate letter. After the card is placed, a new one is chosen and displayed, and so on, until all 25 cards have been placed. Choose your moves carefully, and remember: There are 52 cards in the deck, but you'll only have 25 to play with. The goal is to make the most points possible. Scoring is based on the hands you build. After a column or row is completed, points are totaled and added immediately to your score. (High score is displayed at all times on the screen also.)

Here are the values for each hand:

Hand	Points
Royal flush	400
Straight flush	300
Four of a kind	160
Straight	120
Full house	100
Three of a kind	60
Flush	50
Two pair	30
One pair	10

To remember the value of each hand, you can press f1 at any time during the game to see the table of values. Press it again to resume play. Poker players may notice that some of the hands are out of order. Normally, a flush would be much higher on the list. But remember that you're drawing 25 cards and the odds for getting two or three flushes are very high. Higher point values have been given to hands that are less likely to occur.

Hands do not need to be in sequential order. For example, "5,6,4,7,8" is a valid straight. However, "roll-over" or "round the corner" straights such as "3,2,A,K,Q" are not allowed. Straights using an ace as low (A,2,3,4,5) or high (10,J,Q,K,A) are acceptable.

If you're playing Power Poker on a black-and-white TV, change the value of variable TV from 1 to 0 in line 100.

Poker Hands

If you've never played poker, it's very easy to learn. There are 52 cards, divided into four sets (or *suits*) of 13. The suits are hearts, clubs, spades, and diamonds, and each suit consists of cards numbered 2–10 with a jack, queen, king, and ace. The object is to make one of the following hands (examples are in parentheses):

Royal flush: 10,J,K,Q,A—all of the same suit

Straight flush: a sequence of five of the same suit (9,10,J,Q,K—all diamonds)

Four of a kind: four of the same value (2,2,2,2)

Straight: five in sequence (4,5,6,7,8)

Full house: three of a kind plus a pair (10,10,10,4,4)

Three of a kind: three of the same value (9,9,9)

Flush: five of the same suit (2,K,8,A,5—all clubs)

Two pair: two groups of two, each of the same value (A,A,6,6)

One pair: two of the same value (10,10)

Program Construction

Line #	Function
100–380	Initialization
390–420	Print High Score
430–460	Print Score
470–490	Call Screen #2
500–550	Shuffle/Grid Set-Up
560–650	Game Play Routine
660–670	Test for Completed Row
680–690	Test for Completed Column
700–780	Game End Routine
790–890	Data Collection - Row
900–990	Data Collection - Column
1000–1120	Data Evaluation
1130–1150	Print/Flash Poker Hand
1160–1290	Data: Redefined Characters
1300–1310	Data: Cards/Suits
1320–1340	Data: Screen Location Table
1350–1730	Data: ML Routines

Variables

A$	"{16 DOWN}"
B$	"{1 DOWN}{27 RIGHT}"
C$	poker hand scored
CA	card (J1)
CK	shuffle check
CO	color
CS	total value of row/column
E	box used
H	box selected
HS	high score
K	same kind
L	same suit
NH	new high score
P	card to flash
R	straight
SC	score
SM	screen memory
SU	suit (J2)
TV	color or b/w television

Machine Language Routines

49166	White Background
49180	Clear Card
49194	Grid Set Up
49297	Draw Card
49396	Flash Off
49422	Flash On
49449	Save Color
49502	Restore Color

SOLITAIRE

Ben Elizer

Looking for something different to do with your computer? Like the conventional game of solitaire, this computerized version requires you to think ahead at all times.

"Solitaire" is an electronic version of the familiar card game. Like the original, this game challenges you to put a deck of cards in order using the fewest possible moves.

Unshuffling The Deck

As you probably know, Solitaire has a very simple object. After shuffling a deck of playing cards, you must put them back in order, following a few simple rules. Though there are several different variations of the conventional game, here are the rules for this version:

When you run the program, the computer deals out four rows of 13 cards, then removes the aces, leaving four empty spaces. Your goal is to rearrange the cards into four rows of the same suit, putting the cards in each row in ascending order from the lowest (2) to the highest (king), without leaving any empty spaces between cards. That sounds simple enough. But since you must move a card into one of the four empty spaces, your choices for any given move are limited.

Your position on the screen is shown by a blinking cursor. Press the M key to move from the current position to another empty space. When you press P, the computer moves a card into the current space: Which card it puts there depends on which card is immediately to the left of the space. Whenever possible, the computer uses the next card in suit. For example, if the card to the left of your current position is the 2 of hearts, pressing P puts the 3 of hearts in the current space and puts a space where the 3 of hearts was before. If you press P on a space to the right of the queen of diamonds, the king of diamonds moves from its current position to that space, and so on. Each time you press P, one space is filled and another is emptied.

In this way you can gradually move cards into the right order. When you press P on a space at the beginning of a row, the computer asks which suit to play (hearts, clubs, spades, or diamonds). This determines the suit for that row. While it's possible to win on only one deal, most games require two or more deals. When no moves are possible (every empty space is followed by a king or another space), the computer automatically shuffles the remaining cards and deals them out again. Of course, it does not disturb cards that are already in correct order. You'll find that it takes considerable foresight to win consistently in only two or three deals. Completely random play results in an average of nine or ten deals.

Sea Route to India:
A Historical Simulation For The 64

M. J. Winter

Here's your chance to make history on the "Sea Route to India." Following in the wake of Portuguese explorers, you can find gold and adventure, if you don't starve, or get sunk by pirates, or capsize in a terrible storm.

One of the earliest games for PET computers was *Westward Ho*, in which the player becomes a turn-of-the-century pioneer, trying to cross the country in a covered wagon. Decisions must be made about purchasing food, supplies, and ammunition. Various experiences—hunting, Indian attacks, settlements—occur on each leg of the journey. By repeatedly playing the game, the user learns where to spend money, how to hunt, and whether to trust strangers. Luck, however, is a major factor in success. PET users of all ages played the game over and over until they finally reached the West Coast.

Westward Ho was an abbreviated version of *Oregon Trail*, in which the game's designers took pains to produce an accurate simulation. They used prices from contemporary catalogs, and calculated frequencies and likely locations of Indian attacks by studying historical accounts. The result was a game that was both interesting and informative.

Sail The Bounding Main

"Sea Route To India" uses a similar technique, drawing on the voyages made by Portuguese explorers in the fifteenth century.

The subroutine beginning at line 15000 introduces the game and gives you the rules.

Your goal is to sail from Lisbon around Africa to India. During the voyage, you encounter the same dangers faced by the real explorers: hunger, thirst, pirates, natives, weather, mutiny, and attack by Arab traders.

Your journey is charted in weeks on a map displayed on the screen. Lines 500–800 contain the loop for each week. The miles you sail depend on the weather. Each week your store of water, food, and supplies decreases by one unit. If your voyage lasts more than 30 weeks, the crew's happiness also decreases by 1.

Each week you have a new experience; line 560 sends the program to the appropriate event. In the early part of the voyage, you sight whales and other ships, and sail into terrible storms. But after you pass the Cape of Good Hope and pick up your Indian pilot, you might be attacked at any time by Arab dhows.

Check Your Progress Every Week

At the end of each week, the program assesses your situation. If you sailed far enough to visit the Canary or Cape Verde Islands, then your water, food, supplies, and crew happiness are restored. The ship's log is updated, and the game map shows your progress. Lines 91–93 define DT$ (dots); three characters are needed for each dot. One dot on the map represents 200 miles (line 1002). Then, if there have been no fatal shortages, the voyage continues for another week.

Your ship "sails" across the screen in line 15155. In the race (lines 3093, 3096), the ships are placed at the right of the screen and a string of DELETEs is printed several times. (If you win the race, the crew is happier; they become disgruntled by a loss.)

Lines 1000–1250 contain the whale hunting routine. The whales are within a long string (F$) of *shifted spaces*, which are cyclically rearranged (line 1210) and the leftmost 40 characters printed each time. The program checks the keyboard, then moves the whales until you press H, which drops the harpoon. The program then alternately moves the whales and lowers the harpoon.

To check whether the harpoon hits a whale, the screen is opened for INPUT (line 1100). The entire row of the screen to the right of the harpoon is input. If the first character is not a shifted space, a whale has been hit.

Landfall To Gather Supplies

The subroutine beginning at line 4000 describes the sighting of a river mouth. Landing offers you a chance to get food and water, and to cheer up the crew. Sometimes (line 4060) natives appear. As many early explorers discovered, they are unpredictable. Sometimes they are friendly and trade gold for trinkets (cheering up the crew); sometimes they attack.

If they attack, you must type RUN and press RETURN quickly. The clock is set to 0 in line 4320, to time how fast you typed in RUN. After you press RETURN, the program looks at the clock. If more than 200 jiffies have passed (line 4340), the natives attack and kill you.

The same timing technique is used when the Arab dhows attack. The Arabs are fiercely determined to protect their trading routes. Vasco da Gama himself was nearly trapped by them more than once.

Campaign Manager

Todd Heimarck

This two-player national election simulation ranks as one of the best games we've published. With the right strategy, your candidate can make it to the White House. For the Commodore 64.

The Democratic delegates are gathered in Moscone Center, wearing straw hats, carrying balloons and signs. The floor fights are done. The time has come to nominate.

"Maryland?"

"Mister Chairman—the great state of Maryland, The Free State, Home of the World Champion Baltimore Orioles, casts all of its votes for the senator from Arizona."

The chairman pounds his gavel. The din of cheers and jeers subsides. The convention is deadlocked. And you control a large block of uncommitted delegates. It's all up to you.

The vice president from Rhode Island has good charisma and intelligence, but you know his health is poor. The reverend from Arkansas is attractive, but a bit conservative. Although the senator from Arizona is experienced, he's not very smart. Perhaps the New Jersey doctor? No, the Ohio senator has the best combination of personality and issues, plus you'll get a home region advantage in the populous Heartland.

Now it's the Republican's turn. Of the five choices, the woman from South Carolina is the best all-around candidate. She has high charisma and fundraising appeal, which translates well into television ads.

It's time to hit the campaign trail.

The Democratic senator starts with $9 million and 59 health points. He rests two days (to build up his health), then spends two days fundraising. Campaign stops in Illinois and Texas sway the voters slightly to the Democratic side.

The Republican campaigns in her home state of South Carolina. She then moves on to North Carolina, Virginia, and Florida, followed by a couple of days resting.

As the campaign progresses, the Democrat concentrates on personal appearances in the industrial northeast, plus forays into the larger states such as Texas, California, and Florida. The Republican candidate does less actual compaigning, preferring to spend more time on fundraising to pay for the (expensive) television ads.

In the crucial eighth week, both candidates rest and fundraise in prepration for the last minute campaigning. The Democrat does a media blitz in the Pacific, Southern, and Atlantic states. The Republican hits the Heartland, Arklatex, and the Urban Northeast.

Initial returns from New England show the Republicans sweeping the region, but the large states of New York and Pennsylvania went Democratic. The Republicans won most states from Ohio to the Great Plains, but the Democrats picked up the Southern Atlantic states (except Florida). Texas voted for the GOP, while the rest of the region went Democratic. The Rocky Mountain States were solid Republican. The Democrats won the Pacific States.

The final results show the Republicans winning six of nine regions and capturing the presidency, with 315 electoral votes to the Democrats' 223. Three of the four biggest states voted Democratic, but Ohio and Illinois (with 47 electoral votes between them) made the difference. The TV ads in the last week moved these two key states into the Republican camp.

Nine-Week Campaign

Written entirely in machine language, "Campaign Manager" pits you against an opponent. Each of you manages the campaign of your candidate. The player who makes the right decisions gets his or her candidate elected.

You have nine weeks to campaign. Each week you plan your moves and enter them via the menu on the itinerary. You have two defensive moves, resting and fundraising, and two ways to gain votes, campaigning (personal appearances) and advertising on television.

At the beginning of each turn you see a medium-resolution map of the U.S. which indicates which way each state is leaning. The MAP option allows you to move a cursor around the country, to identify which states are which. If the Republicans are ahead, the state is red. Democratic states are cyan (light blue). If you're using a black and white television, the Republican states are the darker ones. You may notice that states occasionally switch back and forth, even though neither candidate campaigned or advertised there. This indicates that

the voters in that state are split down the middle, and because of slight errors in polling, seem to be leaning one way or the other.

Since you only have 63 days (nine weeks of seven days), you have enough time to campaign in each state once or twice. But in terms of electoral votes, California (with 47) is far more important than some of the smaller (three vote) states like North Dakota or Vermont.

Generally, it makes more sense to campaign more heavily in the ten biggest states, sometimes called "megastates".

State	Electoral Votes
CA	47
NY	36
TX	29
PA	25
IL	24
OH	23
FL	21
MI	20
NJ	16
NC	13

Winning the election requires 270 electoral votes (of a possible 538). The ten biggest states account for 254, just 16 short of a majority.

At the beginning of the campaign, each state has a large pool of undecided voters. As the game progresses, they make up their minds and the pool diminishes. It's possible, but unlikely, for all of the state's voters to decide before the end of the campaign. You would have to go to the state at least eight times before the undecided points were used up.

Each state has a built-in bias toward one party, based on past elections for president, senator, governor, etc. The District of Columbia, for example, is staunchly Democratic, so the Democratic candidate will automatically get seven campaign points there, compared to a Republican's two.

Since the Republicans have won three of the last four elections (including a landslide victory in 1972), you might expect them to begin the game with a huge advantage. But if you look at non-presidential elections, you will find a lot of states that elect Democratic governors, senators, and representatives and then vote for a Republican president. And a lot of those basically Democratic states were split by third-party campaigns (Wallace in '68, Anderson in '80).

To even things up, and make the game more playable, the Democrats begin with an electoral vote advantage of 282 to 256, although four of the megastates (PA, OH, FL, and NC) are barely leaning to the Democratic side. The Republicans have the advantage of beginning with 29 of the 51 states (since DC has three electoral votes, it counts as a state). Most of the states west of the Mississippi are Republican, while the Democrats have most of the industrial Northeast and the South.

In addition to the natural political leanings, each state believes certain things about five general issues:

1) unemployment/inflation,
2) poverty/crime,
3) agriculture,
4) education, and
5) defense.

(The issues are based on census reports, almanacs, etc.) A very urban state might be conservative on crime, but not care much about agriculture, for example. Each candidate has certain stands on these issues. When you campaign or advertise in a state, you can get up to three extra campaign points for each issue, if you agree with the citizens there.

Finally, the candidate you choose has a campaign effectiveness rating based on charisma and intelligence. This factor translates to votes each time you campaign in a state.

To start the game, choose which party will go first. You might want to flip a coin, the winner choosing either a party or to go first or second. In testing, we found that the second player has the very slight advantage of making the last move. Next, decide if one of you will start out as the campaign manager for the president running for a second term. Being incumbent gives you some extra campaigning strength, and is not recommended if you want an even game.

Note that all choices can be made with a joystick in either port. Move the pointer to a menu item and press the fire button twice to make your choice. If you don't own a joystick, use I, J, K, and L for up, left, down, and right respectively. Press M in place of the fire button.

Players then pick which candidate will represent their party. Five randomly chosen candidates are available. To the right of the candidate's stats is the YES/NO counter. Before making your choice, pick NO for each possibility until you have seen all five. They will cycle around again so you can make your choice.

The heart of the game is the actual campaign, but in some ways the convention is more important. Nominate a terrible candidate and you'll spend most of your campaign trying to catch up.

A candidate's personality greatly affects the outcome of the election. In the lower left corner you'll see a list of five attributes, each associated with a number from one (worst) to eight (best). With a couple of exceptions, the ideal candidate is the one with straight eights.

First is charisma (CHAR), which is personal magnetism, panache, the ability to influence and excite people. This is the most important personality trait because it is part of both campaign effectiveness and advertising effectiveness.

Stamina (STAM) rates your candidate's health. A candidate with low stamina will have to rest frequently to regain health and strength.

Intelligence (INTL) adds points to campaign effectiveness and last minute campaigning.

Experience (EXPR) helps you with fundraising. If your candidate has lots of experience, he or she has more contacts and connections for raising money. Since experience comes with age, it counts against your health, although stamina counts for more health points.

Appeal (APPL) also contributes to fundraising appeals. But if you have maximum appeal (eight) you may be tainted by your affiliations with special interest groups, and there is a backlash when you advertise. It's best to have an appeal of six or seven.

The candidates' attributes are generated by adding three random numbers, so candidates are more likely to have a middle number (four or five) than one of the extremes.

The personality traits translate into these five campaign factors:

Campaign Effectiveness (CHAR*2 + INTL): the key factor in campaign stops.

Strength/Health (STAM*4 + 9 − EXPR): determines the effectiveness of a rest day.

Fundraising Appeal (EXPR*3 + APPL): determines how much money can be raised in a day.

TV Ads (APPL OR 8 + CHAR): translates into votes when advertising.

Last Minute Campaigning (INTL + STAM): wins last-minute votes to your side after the ninth week.

The significance of each factor is discussed later.

Campaign Issues

Next to the personality factors are the candidate's stands on various issues. You see five issues, each with a sliding scale of one (at the far left, representing liberal) to six (conservative). A Republican who wants to get tough on crime, for example, will have a rank of six. A Democrat who wants to solve the unemployment problem will have a rating of one.

Candidates will range from two to five on the issues of agriculture and education. On the other three issues, the Democrats will have stands from one to four; the Republicans will go from three to six.

You will generally get more votes with middle of the road beliefs. Look for a candidate with twos or threes if you're the Democrat. Fours and fives are best for the Republican. The exception is agriculture and education, where you do best with a three or a four.

Common sense tells you which issues are important in most states. Agriculture is a major issue in the farming states. Your stand on defense makes a difference in states with a lot of military-related industry.

The candidate's personality is generally more crucial than the stands on issues. If you have a lot of charisma, intelligence, and appeal, it doesn't matter that you may have radical views on one or two issues.

If you have five very bad candidates, press RUN/STOP-RESTORE and try again. It's not much fun to run a campaign you are destined to lose.

After the nominees have been chosen, the first week begins. You may notice that some states have changed colors. That's because each nominee gets the equivalent of campaigning once in each state. Some people make up their minds before the campaign even starts. If one candidate is much more charismatic, or happens to hit the right issues, a state may jump over to his or her side. In addition, each gets a home state and home region advantage.

You should develop a strategy. If your appeal and charisma are strong, concentrate on television ads. If your candidate has a strong anti-crime stance, visit the more urban states. At the very least, you should plan to visit each of the megastates.

You begin in your home state where it is traditional to campaign once (but not twice). And the first week usually means some fundraising and resting as purely defensive moves.

Under the week's itinerary are two numbers representing money and health. At the beginning of each week, your treasurer tells you how much money you have, up to a maximum of $25 million. Your personal physician figures out how healthy you are. At most you'll have 255 health points.

If you fall below $4 million any time during the week, television advertising will be useless until you replenish the campaign coffers. If you have less than one million, you won't be able to pay the pollster (the bar graph to the left of the map will disappear). When your bank account falls to zero, the campaign is paralyzed until you sponsor a fundraiser. You can't even afford to pay your doctor or staff.

It takes time away from campaigning, but you have to raise money once in a while. Each fundraising point (experience times three plus appeal) is worth $200,000.

Campaigning takes a lot out of you, so you have to occasionally take a day to rest and relax. When you decide to catch some Zs, the itinerary will be filled with (you guessed it) Zs. Each day of rest adds double your strength factor, plus campaign effectiveness, plus the number of states you are winning to the health you have. A high campaign effectiveness gives you optimism; you rest better. If you're behind, you lose sleep worrying about it. Resting two days in a row gets you 16 extra health points.

There are two reasons to keep your health up. First, when you campaign in a state, you get an extra campaign point for every 32 health points you possess. Second, if your health falls below eight you look haggard and stutter; campaigning does you no good.

The treasurer counts dollars, the doctor counts your health, and your pollster counts votes.

The pollster does three things. First, you get a bar chart that shows how many electoral votes would go to the Democrats and Republicans if the election were held at that time. You can see it to the left of the map. The gray bar marked U represents undecided states too close to call. Second you have a map of the U.S. to show you, at a glance, which way each state is leaning. Republican states are red; Democratic states are blue. These first two services are part of the pollster's contract, and cost you nothing. Of course, if your money drops lower than one million, you have to stop paying the pollster; all you get is the map.

The third service is the most important—regional polls. To get a poll of all states in a region, move to POLL on the main menu and press the fire button twice. You'll see a bar chart showing which way each state in the region is leaning, from one (half a character wide) to four (two characters). The poll reflects the political situation at the beginning of the week; whatever campaigning you have planned for the week is not included. A state with a thin bar can usually be taken with a single campaign stop.

Don't use polls in the first couple of weeks because most states start out fairly even and you won't learn much. But polling can be a powerful tool towards the end of

the game. If New York is firmly committed to you, forget about further efforts in that state. And if you find a whole region weakly supporting your opponent, you can hit them with TV ads and score a few dozen electoral votes.

Regional polls cost $100,000 and are not available if you begin the week with less than $1 million.

The final character (although transparent) in your entourage is the jet pilot. Your jet can carry you on short hops within a region for almost nothing. But if you travel to a new region, you shell out $100,000 for fuel, maintenance, etc. As long as you're in a region, you might as well stay there a few days to avoid a lot of travel expenses. Again, you don't actually move to a new region until you have campaigned in one of the states. You can use the travel option to conduct regional polls; you'll pay $100,000 for the poll, and another $100,000 if you decide to campaign in a region. If you travel to a region to poll and decide not to campaign, you won't be charged for traveling.

Benjamin Franklin once said that after three days, guests and fish begin to smell. The same principle applies to campaigning.

Campaign once and you gain some votes. Stay for a second day and the voters of a state are flattered; you gain a couple of bonus votes. But stick around for a third or fourth day and you have overstayed your welcome. Do not campaign in a state more than two days in a row.

Voter Points

Each state begins with 255 undecided voter points. Your main goal is to use campaigning and television advertising to sway the undecided. And you have to maintain your health and money.

The effects of a personal appearance can vary. You get up to three points for each issue (if the state agrees with you), one point for every 32 health points, and up to 24 for your campaign effectiveness (intelligence plus double charisma), and a two point bonus if it's your second day in the state.

If your money is down to zero, you get no campaign points. If your health is below eight, you get a single vote.

Each campaign stop decreases your health and money. It's possible to run out in the middle of the week, making each succeeding visit ineffective until you rest or raise money. Let's say you go to Connecticut and impress 23 of the 255 undecideds. The pool of available voters is reduced by that number. Half of 23 (11 points) is charged against your health. Half again (5 points) times $100,000 is subtracted from your money. In addition, each state has some people who don't agree with you, so a quarter of your total (five points) goes to your opponent as a reaction against your speech. If you had previously been in a different region, travel expenses of $100,000 are subtracted.

Television advertising is a little different. It affects every state in the region, and quickly swings voters to your side. To advertise, first travel to the region and make at least one campaign stop to establish your presence. You can then place the cursor on TV ADS and press the fire button twice. After campaigning once, advertise as much as you like.

Unlike resting and campaigning, the effects of advertising do not accumulate from day to day. If you advertise two days in a row, you don't get bonus points. Advertising does grow in strength from week to week, however, and will be more effective towards the end of the campaign.

If you flood the region with ads, it's possible to bring a whole section of the country to your side. But it is costly. In each state, advertising credits you with half your campaign effectiveness, half your TV ads effectiveness rating, points for issues, plus two times the week number (in week seven, for example, you get 14 extra campaign points).

The cost is the usual one-fourth of campaign points gained, plus double the TV ads' effectiveness. The large regions can cost a lot. Going on TV in the Atlantic States (all nine) or in the rocky Mountains (eight) can deplete your treasury.

On the day you plan to advertise, you must have at least four million dollars. If you don't, you waste the day and gather no new votes. So, if you begin the week with $5 million, and campaign in six states, it's likely you'll have less than $4 million by Saturday. Your ad campaign will do you no good.

There is one more item you can choose: RECONSIDER. If you make a mistake, this option wipes your itinerary clean so you can start the week anew. Your choices are not permanent until you fill out the seventh day and press the fire button. (If you pull down on the joystick, your slate will be wiped clean—a quicker way to reconsider.)

Last-Ditch Efforts

The ninth week is usually the most hectic. If you sponsored some fundraisers in week eight, you will want to spend a lot on TV advertising in the regions where you have a chance. Polls can tell you which states are most vulnerable.

After both candidates have finished their last week of campaigning, a couple of things happen. The last region to be visited by a candidate gives a few extra votes to him or her. And the last-week routine goes into action, as all the undecided voters make up their minds. Each candidate gets his or her last-minute campaigning points (intelligence plus stamina) added to each state in the country. The undecided voters are split between the candidates and ties are resolved (based on the built-in bias to one party or the other).

The map is drawn for the final time. The final bar chart appears to the left (which should indicate at a glance which candidate won). Beginning with region one (New England), the electoral votes are displayed, with region totals below.

The winner is the candidate with the most electoral votes. There is a slight chance that there will be a tie, in which case you'd have to flip a coin. If you want to play again, press RUN/STOP–RESTORE and type RUN.

Here are a few rules of ettiquette which help to make a fairer game.

First, if you're playing with two joysticks, try to avoid interfering with your opponent's choices. Remember, the joystick routine

reads *both* joysticks.

Second, when you have filled out your itinerary and the prompt PRESS FIRE BUTTON TO CONTINUE appears, let your opponent study what moves you made, and he or she can then press the fire button.

Third, since polls cost money, they should be kept private. When the other player is taking a poll, avoid looking at the screen.

Main Menu Command Summary

CAMPAIGN—allows you to make a personal appearance in one of the states of the region you're visiting. Results depend on campaign effectiveness, built-in party bias of the state, health, and issues. Does not work if you have zero health or money, or if all undecided voters have been claimed. Gains votes, costs health and money.

TV ADS—blankets the region with advertising. Reduces health and costs a lot of money, but can quickly deliver a big chunk of votes. Net votes based on TV advertising effectiveness, campaign effectiveness, and issues. Does not work if you have less than $4 million.

FUNDRAISE—raises money for your campaign based on fundraising ability. Takes a day, gains no votes, costs nothing.

REST—builds up your health points, according to strength factor. Extra points if you rest two days in a row. Gains no new votes, costs nothing.

MAP—moves the cursor around the map, prints the state name, electoral votes, and region number. For information only, costs nothing.

POLL—provides a bar graph showing which way the states in the region are leaning. Costs $100,000 (immediately). Not available if money falls below $1 million.

RECONSIDER—erases the week's itinerary if you make a mistake.

TRAVEL—takes you to a new region of the country. Costs $100,000 (not charged to you until you actually campaign there). ©

Pool

Joseph T. Woyton
Version by Kevin Mykytyn

Chalk up your cue stick and sharpen your skills with this exciting simulation of pocket billiards. For one or two players.

The rules of "Pool" are simple—you try to sink the billiard balls on the table by aiming and shooting the white cue ball.

To play Pool, load it from disk using a secondary address of 1: **LOAD "POOL",8,1**. **SYS 49152** starts the program.

The title screen comes up first, with a ragtime melody playing in the background. You choose a one- or two-player game. The screen clears and the pool table appears.

At the start of the game, the white cue ball is at one end of the table and six balls are arranged at the other end. There are six (rather than 15) because only eight sprites are available on the 64. Using the joystick, position the cue ball in the "kitchen," behind the scratch line. When you're ready to shoot, press the joystick button once. A crosshair appears on top of the cue ball.

Move the crosshair in the direction you want to shoot. Pressing the joystick button starts the cue ball rolling.

The distance between the cue ball and crosshair determines the strength of the shot. The farther away, the harder the shot. There is a limit on how far you can move the crosshair (approximately two-thirds the length of the table). On the initial break, you'll probably want to shoot hard. On later turns, the strength of the shot will determine how far the cue ball travels after a collision. Strategic soft shots can help you set up the table for the next shot.

The goal in the one player game is to clear the table in the fewest number of shots (the record here at COMPUTE! Publications is eight). When all balls are in the pockets, you're ranked according to your ability, from Pro (the best) to Pool Shark, Amateur, and Novice.

In the two-player game, you try to outshoot your opponent. When you sink a ball, it's placed on your side of the screen. With six balls in play, tie games are possible.

A scratch occurs when you knock the cue ball into a pocket, or when the cue ball doesn't hit anything before coming to a stop. You lose your turn, one of the balls to your credit is put back on the table, and the cue ball is placed in the starting position. Your opponent can then put the cue ball anywhere behind the scratch line.

Pool does not completely follow the laws of physics, although it offers a realistic simulation. The sprites are moved pixel by pixel, but the movement is calculated in 256ths of a pixel for increased accuracy. ©

Bingo 64

Richard L. Witkover

"Bingo 64" is a cleverly written computer version of the classic game. It makes good use of the 64's graphics and sound capabilities to provide you and three friends with many exciting games of bingo. A joystick is required.

Few people have not known the anticipation, heard the click of the balls, the call of the number, and finally, the excited shout of "Bingo!" Here's a four-player version of this world-famous game written for the Commodore 64.

Setting Up The Game

Before you begin playing, you have to choose your mode of play from a menu. *Manual ball feed* allows you to control the pace of the game. If you opt for *auto ball feed*, the computer automatically picks the next number. Choose *manual cover* if you want to cover the spots on your card yourself (using the joystick). If you want your 64 to cover the spots, pick *auto cover*.

After you make your choices, four blank cards are displayed on the screen. Each column on a card corresponds to one letter of the word BINGO. The five numbers in each column are selected from 15 possible values and are checked to prevent duplication. This process is repeated for all four cards. The selected numbers are displayed on the cards and stored in the three-dimensional array C%. A cover token is then placed over the free box in the center, giving it the status of a called number.

Ready To Play

The cards are on the screen, and you are ready to begin playing. At the bottom of the screen are five balls (sprites), each labeled with a letter in the word BINGO.

Numbers are selected through a random graphics routine. The lettered balls jump up and down like kernels of popcorn. The height a ball reaches is random. If it jumps above a line on the screen, it is selected. If not, it falls back with a plop, and another ball is given a chance.

After the column letter has been selected, a random number is chosen. To prevent duplication, all called numbers are entered in the two-dimensional array N%. The newly selected number is compared to the numbers in the array. If a match occurs, the number is discarded and another one generated. When a unique number is found, it is printed on the enlarged sprite in the middle of the screen, along with its letter.

Checking For A Match

Next, each number in the appropriate column on each card is checked for a match. A chime sound signifies a match. If the number doesn't appear on any of the four cards, a sour bong sounds.

If you selected auto cover, the computer covers a matched number by printing it as a red reversed character. If you chose manual cover, you must use the joystick to position the marker and press the fire button to cover your number. If you make a mistake, you lose the number.

When a number is covered, the program checks the row, column, and diagonal for a bingo. Since the covered numbers have been printed in reversed character mode, this is easily done from screen memory. The program needs to check the five boxes to see if they all contain reversed characters (screen codes greater than 127). If a winner is found, the covers change color and music plays. Each card is checked for the possibility of multiple winners.

One Machine Language Routine

The first DATA line contains a short machine language routine which initializes screen color and sprite memory locations. The remaining DATA statements set up sprite data and music. The program contains extensive REMarks to make the logic flow easier to follow, and the variable names are listed below, with their functions.

Variable	Function
A1-4	Parameters of pattern for winner scan
AU	Mode flag for ball feed and cover
BO	Bingo flag
B0,1	Digits under joystick cursor
C%	Card numbers array
CC,CM	Joystick cursor position
CD	Card number
CL	Column number
CO%	Ball color array
DU%	Duration of notes array
FB	Fire button switch
FH%,FL%	Music frequency high and low bytes
FR	Joystick row number
HB	High byte of screen color memory
J0-3	Joystick direction switches
L	Ball letter array
LN	Length of ball number string
N%	Called number array
NC	Ball color index
NM	Called number
NN	No-Number match flag
NU	Value of called number within column (1–15)
N1,2	Digits of called number
OB	Reversed number flag (logical variable)
PD	Joystick memory register contents
P1	Screen memory location of N1
RN	Row number on card
S	Start of screen memory for cards
SD%	Sprite data array
SN	Screen memory location of box digit
SS	Sound chip memory location; also, screen-to-color memory offset
V	Video chip memory location
WI,WJ	Indices of box to be checked for winner
WM	Memory location of box to be checked for winner
SP	Contents of WM
X	Ball X-position array
Y	Ball Y-position array
YM	Maximum ball height
Z	Present box number value

Sleuth

Paul D. Farquhar
Version by Kevin Mykytyn

This intriguing game challenges your problem-solving abilities. A crime has been committed, and you must question suspects to discover who's responsible. What makes things difficult is that one of the suspects has problems telling the truth. And what gives the game added appeal is that it's different every time you play.

When you were invited to a small dinner party at Lord Crumbly's secluded mansion, you never thought you'd be accused of murder. In addition to yourself, there were three other guests and two servants. You knew his lordship was not a well-liked man, but you were certainly not expecting murder.

Late that evening, while you were alone in the library, a shot broke the silence. You rushed to the room of Lord Crumbly and discovered his lifeless body. A moment later, the others arrived to see you bending over the dead man. The police were summoned, but have not yet arrived. As the prime suspect, you must quickly discover the murderer or risk being accused of the crime.

The murder can be solved by questioning the suspects and keeping track of who said what about whom. But, to complicate matters, you know that one of them is a compulsive liar.

Sifting Through The Evidence

Three questions must be answered affirmatively before you make an accusation. First, did the suspect have a motive—a reason to dislike Lord Crumbly? Second, did the suspect have access to a gun? Third, where was he or she when the shot was fired—with someone else, near Crumbly's room, or somewhere unknown?

Your task is to narrow down the list of suspects until you find someone who had a motive, a weapon, and an opportunity. Obviously, if Professor Bard had no gun, he must be innocent. You may find more than one person who had motive or opportunity, but only one will fit all three categories.

Four of the five will always tell the truth, while one will always fib. The liar is not necessarily the murderer, although it is possible. How do you distinguish between the true and false statements?

At first, you don't know who's trustworthy, so don't believe anyone. Just write down what they say on the program's a four-page notebook. You can look at your notebook by pressing one of the function keys. The f1 key is the first page, f3 is the second, f5 is page 3, and f7 page 4. You may find it convenient to split up motive, weapon, and opportunity on the first three pages and use the fourth for your conclusions. After selecting a page, type in the message you want. You can also flip between the pages of the notebook by pressing the other function keys. To get back to the game, press the back arrow (the upper-left character on the keyboard).

When someone says something, write it down on one of the first three sheets. For example, if Snipe says Larue was being blackmailed by Lord Crumbly, write it under Motive. If Bard says Chauncy was with someone when the shots were fired, write it under Opportunity (Bard says Chauncy had no opportunity).

Paring Down The List Of Suspects

Who's telling the truth?

If Snipe tells you that Whiggins did not have a gun, and Larue says the same thing, you can conclude that both Snipe and Larue are truth tellers. Since there's only one liar, if Snipe or Larue was a liar, the other would have to be telling the truth, and they would say opposite things. But they said the same thing, so they must both be telling the truth. Once you know they're OK, you know everything they have said (and everything they may say in the future) will be correct.

By similar logic, if two suspects say contradictory things, you know one of them is lying. Bard says Snipe hated Lord Crumbly (a motive), but Chauncy says Snipe had no motive. Either Chauncy or Bard is not telling the truth, and since there's only one liar, Whiggins, Larue, and Snipe must all be truth tellers.

As you collect your data, look for someone who had motive, weapon, and opportunity. As soon as you find a suspect with all three, you can solve the case. Another way to nail the murderer is to rule out four suspects. For example, if Whiggins had no gun, Snipe no motive, Chauncy no motive, and Bard no opportunity, then that leaves a single possibility: Larue.

After uncovering the answer to the mystery, you'll see how many clues you were given (you can usually finish with about ten clues). You'll then be asked if you want to play again.

Page Flipping

The electronic notebook is created using page flipping, which means relocating the screen display to a different section of memory. This technique is often used for animation on the Apple and Atari. While one screen is displayed, the computer draws the next picture on an alternate (invisible) screen. Then, a few POKEs cause the screen to change to the next frame. You don't need to know how it works to play

Sleuth, but if you'd like to use page flipping in your own programs, here's a brief explanation.

Location 53272 controls several things, including the video matrix base address (where the screen starts, in plain English). The upper four bits of this location point to the start of the screen. Normally the bit pattern is 0001, meaning that the screen starts at 1*1024. If we want to change the location of the screen, we must be careful not to disturb the remainder of this byte. This can be done using the following statement:

POKE 53272,PEEK(53272)AND15 OR S*16

The variable S holds the number of the 1K block where the screen starts. It must be a number from 0 to 15 because the video chip can access information such as screen memory, sprite definitions, and character definitions from only one of four 16K blocks. So to change the start of screen memory to 12288 (12*1024), type:

POKE 53272,PEEK(53272) AND15 OR 12*16.

But POKEing to 53272 is not enough. There are a couple of problems to overcome. Even though the computer is now displaying screen memory from a new location, the BASIC editor is still printing to the old screen.

Location 648 tells the operating system where to find the screen if it needs to print something. It contains the start of screen memory divided by 256. Since the screen normally starts at 1024 this location is normally 4 (4*256 is 1024). To change the start of screen memory to 12288 enter:

POKE 648,48.

Even after telling the computer where to start its screen display memory and telling the editor where to print, there is still a problem. Locations 217 to 242 hold a screen line link table necessary for proper formatting of anything printed to the screen. If you type more than 40 characters on a line, the computer uses the line links to find out where to start the next line After flipping pages, this table will still be set to work with the old screen. The line link table must be rebuilt to work with the new screen. BASIC has a routine that will do all this for us; the clear screen routine. Once the screen is cleared, all the line links are set for the new screen. But in Sleuth, it's necessary that the screen not clear when the pages are flipped. The machine language routine at the start of the program does the relinking without clearing the screen and sets the color RAM to blue.

It's possible to flip screen memory pages, but there is only one location for color memory (starting at 55296). This makes it necessary to set the color memory when a page is flipped. Otherwise, the characters could be on the new screen, but might be the same color as the screen and would not be visible. A second method is to save the color RAM in another portion of memory and move it back when the page is displayed.

Finally, the new screens that you create should be protected from BASIC by POKEing location 56 and doing a CLR. Also, when you're typing on an alternate screen and press RUN/STOP-RESTORE, location 53272 is reset but location 648 is not. Display memory is now at 1024, but your typing is still being printed on the other screen. This can be fixed by typing:

POKE 648,4

and pressing RETURN (even though you can't see what's being typed).

If you don't want to write your own page flipping routine, you can use the one in Sleuth. Include lines 1–9,36,320,and 485–540 in your program (you can renumber them, of course). Line 1 protects the new screens from BASIC and reads the machine language routine, from lines 2 to 9, into memory. This should be done at the start of the program.

Line 36 calls line 540, which clears all the alternate screens. Then the position array is set to 0. This array, labeled C, keeps track of which row the cursor is on when a page is flipped and returns the cursor to that row when the page is restored. This routine should also go at the beginning of the program. Line 320 checks for the press of the function keys. It should be included as part of your key input routine. Lines 485–535 are the main page flipping routine and can be placed anywhere in the program. C

Switchbox

Todd Heimarck, Assistant Editor

Here's a challenging game of strategy for the Commodore 128 that looks easy at first, but takes time to master and permits many variations.

Playing "Switchbox" is like putting dominos in place for a chain reaction—either you're setting them in position or you're knocking them over. Winning requires skill and a sense of when to go for points and when to lay back and wait for a better board. The goal is simple: You try to score more points than your opponent by dropping balls into a box full of two-way switches. Each switch has a trigger and a platform. If the ball lands on an empty platform, it stops dead. But if it hits a trigger, it reverses the switch and continues. In many cases dropping a single ball creates a cascading effect—one ball sets another in motion, which sets others in motion, etc., all the way down.

Box Of Switches

Switchbox is a tale of twos: Each switch has two parts, two positions, two states, two paths in, and two paths out. The two parts are the platform and the trigger. A switch can lean to the left (platform left, trigger right) or to the right (platform right, trigger left):

The trigger is weak, and always allows balls to pass. But the platform is strong enough to hold a single ball. So the platform either holds a ball—it's full—or it does not and is empty. When a ball sits on a platform, the switch is said to be loaded, or full.

Figure 2 shows a full switch over two empty switches. The platform holds a ball and leans to the left. The trigger extends to the right. Note that the switch on top has two pathways leading in, the left path and the right, and that the right path leading out is the left path into one of the switches below. The left path of the top switch leads into the right path of the other, the switch below and to the left. If you drop a ball down the righthand path, it hits the trigger and flips that switch to the right. Then it continues down, hits the lefthand trigger below and flips that switch as well.

Figure 1. Trigger States

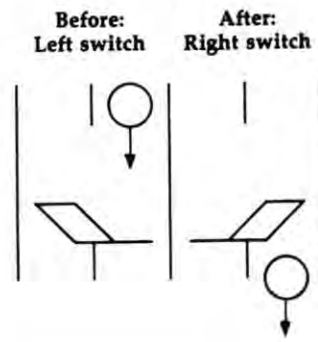
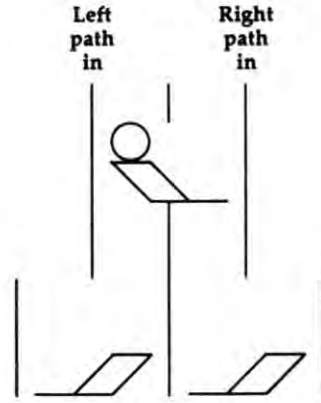


Figure 2. Loaded Trigger



In the meantime, the ball on the platform is set in motion (when the switch is flipped) and then hits the trigger. The top switch is reset to point to the left. The second ball then drops a level to the platform below, where it stops. The playing field is composed of five levels, with four switches in the first level and eight in the bottom level. At the beginning of the game, there are no balls on the field—all platforms are empty—and the position of each switch is chosen randomly.

Moving Down The Path

Players alternate dropping balls into one of eight entry points. These balls (and others) may or may not make it all the way through the switchbox, to one of the 16 exit paths. Balls fall straight down (with one exception), so a ball's movement is always predictable. When it hits an empty switch, one of two things can happen. If it lands on the empty platform, it stops dead in its tracks. But if it lands on a trigger, it falls through to the next level below.

Moving balls always make it through loaded switches. Triggers allow balls to continue, and move the switch to the other position. If it's loaded, the dead ball on the platform is put into motion and it hits the trigger that just moved over.

This makes the switch go back to its original position, but with an empty platform. So when a ball hits the trigger of a loaded switch, its motion continues unabated. The switch moves, the ball on the platform begins to fall and it hits the newly placed trigger. The newly emptied switch moves back again, and the two balls drop to the next level.

There's one more possibility: a ball dropping onto a platform that already holds a ball. A platform can't hold any more than one ball, so when this happens one of the balls slides over to the trigger. So the ball does not move straight down—it slides over to the next pathway. This is the exception to the rule that balls drop in a straight line. Of course, when the ball hits the trigger, the switch changes position, causing the other ball to drop and hit the trigger.

The Chain Reaction

At the game's start, all platforms are empty, so four of eight entry paths are blocked. Remember that your turn ends when a ball hits an empty platform and stops. As the switches fill up, the chances increase that a ball will descend through several levels. The goal is to score points by getting balls to pass all the way through the maze of the switchbox. The best way to collect a lot of points is to cause a chain reaction.

A ball that hits a loaded switch from either side continues on its way. And the previously inert ball on the platform starts moving. One enters, two exit. If both of those balls encounter full platforms, four drop from the switches. The pathways are staggered, so the effects can spread outward, with more and more balls cascading toward the bottom.

Rather than taking an easy point or two, it's often worthwhile to build up layers of loaded switches. Watch out for leaving yourself vulnerable, though. Because players take turns, you'll want to leave positions where your opponent's move gives you a chance to create a chain reaction. The best strategy is to play defensively. Look ahead a move or two, and watch for an opening that allows you to score several points at once.

Four Quarters

A game of Switchbox always lasts four rounds. In the first (equality), each exit counts for two points. Your goal is to score ten points. The second quarter has more points available, as well as a higher goal. If you look at the exits, you'll see that the further away from the middle, the higher the point value. The numbers increase in a "Fibonacci" sequence: 1, 2, 3, 5, 8, and so on. Each number is the sum of the previous two (1+2 is 3, 2+3 is 5, 3+5 is 8, etc.). The target score in round two is 40.

In round three the numbers are a bit lower. They increase arithmetically (1, 2, 3, 4, up to 8 in the corners). A goal of 20 points brings you to round four, where you can score big. Here the numbers are squares: 1, 4, 9, 16, 25, all the way to 64 at the edges. In rounds two through four, it's sometimes prudent to leave a middle path open for your opponent to score a few points, in order to gather a high score on the big numbers to the left and right.

Each round lasts until one player has reached the goal. At that point the other player has one last turn before the round ends. It's possible to win the round on this last-chance play; watch out for barely topping the goal and leaving a chain reaction open for the other player. An arrow points to the scoreboard of the player whose turn it is. On the other side of the screen, you'll see a number where the arrow should be. That's the goal for the current round (the Amiga version displays the goal on both sides of the screen, below the scoreboards).

Bonus points are awarded at the conclusion of each round. Four numbers appear below the scorecards. The first is simply the total so far. The second is the total plus a bonus of the goal for the round if the player's points are equal to or greater than the goal. For example if the goal is 20 and you get 18, there's no bonus. If you score 22, the bonus is the goal for that round (20) and you'd have 42 points. The third number under the scoreboard is the difference between scores for the rounds. If you win by two points, two is added to your score (and two is subtracted from the other player). The final number is the grand total of the first three scores and bonuses. Rounds one and three are fairly low-scoring with low goals. You may want to seed the field with extra balls during these quarters, so you can collect more points in the second and fourth quarters.

Variations

Although the goal of the game is to score the most points, there's no reason you couldn't agree to play for low score. In a "lowball" game, you would try to avoid scoring points. You wouldn't necessarily play backwards, you would have to adjust the strategy of where to place the balls. Fill up the board as much as possible and leave your opponent in a situation where he or she is forced to score points.

The DATA statements at the beginning of the program (the *Setup:* routine in the Amiga version) determine the goal for each round and the point values for the exit paths. You can prolong the game by doubling the goals; this also dilutes the value of a big score at the beginning of a round, preventing one player from winning on the first or second turn. An interesting variation is to assign negative values to some slots. If some paths score negative points, you are forced to think harder about where the balls will drop.

In addition to the numbered keys (1–8), the plus (+) and minus (−) keys are active. Pressing plus drops a ball at random down one of the eight entry paths. Pressing minus allows you to pass your turn to your opponent.

Once you've mastered the regular game, you can add some new rules. Each player gets three passes per half, similar to the three timeouts in a football game. If you don't like the looks of the board, press the minus key to use one of your passes. After one player has skipped a turn, the other player must play (this prevents the possibility of six passes in a row). It's also a good idea to make a rule that a player can't pass on two consecutive turns. You can also give each player two random moves to be played for the opponent. In other words, after making a move, you

could inform your opponent that you're going to give him one of your random moves and you would press the plus key.

Here's one more change you could make: Instead of alternating turns, allow a player to continue after scoring. When a player drops a ball and scores some points, the other player would have to pass (by pressing the minus key). If the first player scores again, the opponent passes again, and so on until no more points are scored.

Playing Solitaire

To drop a ball, press a numbered key (1–8). If you're using a 128, ST or Amiga, the numeric keypad is convenient for choosing a move. By using the pass and random turn options, you can play against the computer. Here are the rules for solitaire play:

1. The computer always scores first. At the beginning of every round, the computer plays randomly until at least one point is acquired. Press the plus key for the computer's turn. You must continue passing (skip your turn with the minus key) until the computer puts points on the board.

2. After the first score by the computer, you can begin to play. When the computer has a turn, press the plus key for a random move.

3. Whenever you make points, you must pass again until the computer scores. When the computer gets more points, you can begin to play again. This rule means you should hold back on the easy scores of a few points; wait until there's an avalanche available.

4. If you're the first to reach the goal, the computer gets a last chance. Don't make this move randomly; figure out the best opportunity for scoring and play that move for the last-chance turn.

In the interest of keeping the program at manageable length, no attempt has been made to provide an "intelligent" computer opponent. Once you become familiar with the game, you might find it an interesting project to try adding some routines that give the computer a rational basis for picking one move over another. ©

Q-Bird

Mike Sedore

This delightful and colorful arcade-style game for the Commodore 64 challenges your character, a defenseless (but nimble) baby bluebird, to survive among a crew of nasty, hungry enemies. A joystick is required.

Other birds say you're paranoid, but you're not—everyone really *is* out to get you. You often ask yourself how long a defenseless baby bluebird can hope to survive when a host of voracious predators are looking for a meal. If only you could leave this place to find a new home. But alas—you're too young to fly. The best you can manage is a flapping long jump. But you'd better be careful not to jump too far: You could fall a long way down.

Q-Bird is played on a 6 × 7 grid. There are 15 levels of play from which to choose. As you play, you advance level by level by completing grids. A grid is complete when the color of each square matches the goal color shown at the bottom of the screen. To change the color of a square, simply hop on it. You have to jump on each square once in level 1, twice in level 2, and so on up to level 15. A bonus life is awarded each time you complete a level. If you reach level 15, you should feel satisfied. It doesn't get any harder than this. But don't relax; it doesn't get any easier either.

You start with five lives. Choose your starting level by pushing forward on the joystick (which must be plugged into port 2). If you pass the level you want to play, pull back on the stick to reverse the level counter. Press the fire button to begin play. To hop, simply move the joystick in the direction you want to go. For a super jump, push the fire button as you hop: You'll leap over a square and land on the next. If you time it right, you can leap right over your enemies. But be careful not to jump over the side of the grid. That costs one life.

Press SHIFT–LOCK to pause the game. To restart a game, press RUN/STOP–RESTORE.

Leapin' Lizards

Three purple lizards live on the grid. They randomly leap off of their tails from square to square, landing on any bluebirds careless enough to get in their way. But they're the least of your worries. There's a king cobra that relentlessly chases you. He's got a hole in his stomach just the size of a baby bluebird. If he doesn't get you, then perhaps the low-flying and hungry hawk will. And while you're looking out for all of these villains, try to dodge the runaway balls that roll down the grid. Any one of them could turn you into a bluebird pancake in a moment.

Fortunately, you do have an ally. Occasionally, a flashing egg appears on a random square. If you hop on it, you momentarily stun all the grid inhabitants. You can now go anywhere you please without harm. Unfortunately, this doesn't last long. When the safe time is nearly through, your character, the bluebird, begins flashing. All action returns to normal after the third flash, so be ready for the frenzy to continue. ©

Hex War

Todd Heimarck, Assistant Editor

You float high above a distant planet, controlling robot armies below. Can you take control of the priceless mining turf planetside, or will your opponent's robot crews prevail? To win at this thoughtfully designed, engaging strategy game, you'll need foresight and conceptual skills rather than a quick hand on the joystick.

"Hex War" is a two-player strategy game that can be played five different ways, and there are limitless variations. But the basic premise is always the same: You and an opponent move armies on a field of hexagons, attempting to capture territory.

The goal of the first two games is simple: capture the capital city of the other player. In game 1, the capital cities are far apart; you must devote some of your armies to defending your own capital while attempting to breach the walls of the other capital. Game 2 puts the capitals near each other, so offense and defense tend to merge in this scenario. Most of the action takes place within a small area of the battlefield.

Games 3 and 4 spread the action over a wider area. In the third game, your object is to occupy eight of the twelve cities on the game board. Six cities occupy the periphery, and six are in the center of the playfield. Game 4 requires actual control of six cities; you must have an army in the city, one that's not involved in a battle, before you're credited with control (this version will probably take the most amount of time to play).

Although the first four scenarios encourage a commitment to battle, you employ different tactics in the fifth. The goal here is to acquire 40 of the 61 hexes, so you need some free armies to move around. As soon as you claim 40 hexes, you win the game.

When you first run Hex War, the computer pauses to set up the screen, then displays a menu of five choices. The five different games are explained in detail below. If you're new to the game, press the 1 key to choose game 1. There will be another short pause while the variables are initialized, and then you'll see a playfield with 61 hex shapes, containing four armies on each side.

Hexes And Hexadecimal

A chess board has 64 squares arranged in a rectilinear grid. Hex War gives you a playing field of 61 hexagons (almost as many as a chess board), but they're part of a six-sided honeycomb field. If you've played war games before, you may recognize the hexes.

Before playing, plug the joystick into port 2. At first, the cursor movement may seem unusual. The cursor travels not up-down/left-right, but northeast-southeast/northwest-southwest. To make the movement less confusing, turn your joystick 45 degrees clockwise, so that what was up becomes northeast, and so on.

Each hex has six neighbors, so an army can move in six possible directions. To travel left and right, you'll have to push the joystick twice (for example, up and right on the joystick to move one hex to the right, which counts as one movement).

Army strengths are listed in hexadecimal (base 16) numbers, so the four armies labeled 40 actually have strengths of 64 (the hexadecimal value 40 equals 64 in our everyday decimal numbering system). At the beginning of a turn, any army has exactly three movement points. It requires one point to move an army into a neutral or enemy-controlled zone. To move *through* the same zone also requires a point. To move into and through a friendly hex requires a total of one point. This means you can move a single army through two neutral or enemy hexes in any one turn, but the same army can move through up to three friendly zones during a turn.

Select an army by moving the cursor onto it. Click the joystick button once, then position the cursor on a neighboring hex and click again. If you wish to stop, click again, and two plus signs (++) will appear, signaling that no more movement can occur. Otherwise, position the cursor on another neighboring hex and click.

Zones Of Control

Each army controls the six contiguous hexes surrounding its resident hex. If you enter an enemy's zone of control, you forfeit any additional moves and must prepare for battle. In addition, an army that begins the turn in a zone of control cannot move until the battle is resolved.

Robots Vs. Robots

In this game, you aren't really on the planet, but parked high above it in a remote mothership. You've landed some robots to explore the area, and they've encountered robots belonging to another explorer. Your robots, or *bots* as you call them, follow your orders to advance toward the other bots. Each bot has a mining laser which can stop or disable the other bots. Also, your bots have disruptor beams which can daze another bot, temporarily confusing it. When two bot-groups come close to each other, they shoot lasers and disruptors until one army of bots is disabled.

Three things can happen to a robot which suffers a hit. If the robot suffers a direct hit in its logic unit by a laser, it is vaporized. It is destroyed forever and never reappears in play.

The second thing that can happen is injury. If the laser beam is deflected, the robot is out of commission until it can be transported back to a botspital. An injured bot is frozen in place until the battle is finished, after which the victorious army carts away the injured bots to be repaired and reused.

Thus, winning a battle means you evacuate both the friendly injured and the enemy injured. After all of the injured bots recover, *they join the force in whose botspital they were healed.* In effect, injured bots eventually become members of the army which won the battle in which they were damaged.

The third possibility is confusion: The robot is temporarily disoriented for two turns. When the time has passed, the robot is ready again.

Reprogramming Bots

Moving the cursor onto an army of robots brings up a status window in the upper-left corner of the screen. The number in reverse video is unimportant; it's the army number (which may change as the game progresses).

The four numbers underneath are significant, however. The first is the army's active strength (in decimal). The second is the number of injured robots, which will be transported to the botspital of whichever side wins the battle. The third—on the line below—is the number of disrupted robots who will be available for combat in the next turn. The fourth number is how many robots can join the active force two turns from now.

If one side is able to reduce the other player's active force to zero, two things happen. The winner sends all injured bots away to be repaired. The winning side also collects all enemy bots (injured or dazed) and sends them to the reinforcement center to be reprogrammed. Eventually all these bots will available to the winner of this particular battle for future engagements.

Reinforcements And Mergers

At the start of the game, you'll see some armies positioned outside of the hex field. These are reinforcements and reserves in transit to the battle. Player one's reinforcements enter at the bottom right corner; player two's enter at the top left. The line of new armies moves counterclockwise; the army next to the entry point is the next to enter the battlefield.

However, the reinforcements cannot enter the battlefield if an army (friendly or enemy) is blocking their way. Keep your armies off your own reinforcement point, and try to block your opponent's armies from this area if you can. If the entry hex is owned but not occupied by your opponent, you'll lose some reinforcements.

After completing a turn, you are credited with additional reinforcements according to how much territory you own. Passing over a hex allows you to claim it; the hex changes color to indicate ownership. Each piece of property provides enough ore and energy to build a new robot, available for use two turns in the future. The numbers in the line of reinforcements are updated after you move to show additional robots being built.

Winning a battle also provides additional armies in the line of reinforcements. As mentioned above, a victorious army captures any dazed enemy bots, which are reprogrammed and available in three turns. At the same time, the winner evacuates injured bots of both sides. Transportation and repair take five turns for friendly bots, seven for enemy bots. The two additional turns are needed for reprogramming the opponent's forces.

If you're losing a battle, the number of injured robots (displayed in the status window) will begin to rise. Remember that, if your opponent reduces your active strength to zero, he or she will capture all of your injured bots; they'll be reprogrammed and added to future reinforcements. To prevent this from happening, you're allowed to bring in a second army for merging. Simply move another army on top of the army with which you want to merge. There's just one rule: One or both of the armies must have a strength less than 32 decimal (1F or less in hex).

Customizing The Scenarios

The five built-in scenarios provide plenty of variety, but if you'd like to add more challenges, here are some suggestions.

First, a note about the logical organization of the grid. The variables T and B, CT and CB, and HT and HB are used to locate the coordinates on the playing field (see figure). The first number is T (or HT or CT), the second is B (or HB or CB). These coordinates are also used in the three-dimensional MAP array (where level 0 of the array is the army number, 1 is the current owner and 2 keeps track of whether or not a city is located there); they're also part of the ARMY array. By varying the starting position, number of armies, reinforcement strengths, and location of cities, you could simulate historic battles.

To add or subtract cities from the field, change the value of CN in line 50. You'll also have to change the DATA statements in lines 270 and 280. The numbers there are the T and B coordinates of the cities.

The strengths and locations of

the armies can be changed as well. The DATA statements starting at line 1540 determine the strength (64) and T/B coordinates for the armies at the beginning of the game. If you wish to start with more armies (or fewer), you'll have to change the inner FOR–NEXT loop (with the index of K) in line 1500. In that same line, change NX(J) to one number higher than the number of armies on each side. For example, if you want six armies apiece, change NX(J) to 7. The subroutine at line 1600 sets up the reinforcements; if you don't like the random patterns, change the formula here.

Variables defined in lines 70–90 control the play of the game. PN determines which player goes second; it can be either zero or one. Variable ME controls the maximum merge strength. If you'd like to be able to merge any two armies, change it to a high value (512, for example). To remove the merge option altogether, change ME to zero.

The movement points are defined by MM in line 80. Movement across friendly territory takes one point, across neutral or hostile territory two points. Increasing MM will give your armies more mobility. The three variables KA, KB, and KC affect the outcome of individual battles. KA determines how many bots are vaporized, KB controls the number injured, and KC affects how many are dazed. If you make the fractions smaller (1/24, for example), the battles end more quickly. The subroutine starting at line 2600 resolves current battles.

Running Hex War

To play Hex War, type this line in direct mode (without line numbers):

```
POKE 44,64:POKE 64*256,0:NEW
```

Be sure to press RETURN after you type the line. Now load and run Hex War as usual. It is very important that you perform this step before running the program: If you don't, the screen will be jumbled and impossible to decipher.

You may find it easier to let the 64 handle this chore for you. "Hex War. Loader" is a short loader which performs the setup, then loads and runs Hex War. To use the loader, you must have the main program saved with the name HEX WAR on the same disk as the loader. ©

Space Gallery

Jeff A. Lapkoff

Swirling frisboids, pellets, and heat seekers are headed your way. Can you fend them off with your cannon? A colorful arcade-style game written entirely in machine language for the Commodore 64. Joystick required.

The year is 2023, and you're a solo space gunner on an earth-orbiting colony. You must try to defend against hostile forces, and as the colony's sharpshooter, you must remain stationed at the video control center. For days, the radar has shown an approaching force, and based on the direction and speed at which they're coming, you know what to expect: the dreaded frisboids, curious but deadly creatures. They come in waves of four, but you have only one cannon. And they shower you with lethal pellets, which can destroy your cannon.

The object of "Space Gallery" is to shoot as many frisboids as you can while dodging the falling pellets. To fire at a frisboid, press the fire button on your joystick (plugged into port 2). For each one you hit, you earn five points. Your cannon has unlimited ammo; however, the fewer shots you take, the higher your shot bonus at the end of each level of play.

To dodge the pellets, move your cannon left or right with the joystick. If your cannon is hit by a pellet, it will explode. The game ends when you've used up five cannons.

A Bonus And An Option

Shooting down five waves of frisboids advances you to a higher level and, if you didn't use too many shots, rewards you with a shot bonus. Once you get to level five, you receive an extra cannon.

Space Gallery also features a heat seeker option, but beware—it makes the game much more difficult. The heat seeker is a normal-looking frisboid that drops blue pellets, which home in on your cannon. At the beginning of the game, you can activate the heat seeker by toggling on the gray square at the bottom of the screen. ©

Whirlybird

Philip I. Nelson

This hyperactive bird lays some rather unusual eggs. A fast action game written entirely in machine language for the Commodore 64. A joystick is required.

"Whirlybird" is a fast, updated version of the classic game *Breakout*. It's written entirely in machine language. You control the Whirlybird, who flies back and forth at the top of the screen, spinning continuously. The goal is to clear out the layers of colored bricks at the bottom by bouncing eggs against them. When an egg moves back in your direction, move the Whirlybird into its path to bounce it back down. The game ends when you clear the field of bricks or run out of eggs.

Plug a joystick into port 2, then load and run Whirlybird as if it were a BASIC program. *Do not try to start the program with SYS.* The playfield appears immediately, with the Whirlybird twirling across the top of the screen. On either edge is a sparkling row of sidebeams, and below are multicolored rows of bricks. At the upper right is the number of eggs you'll have in this game. Though the usual number is ten, you can get a smaller or larger number (up to 255) by pushing the joystick backward or forward. (Don't select zero—you'll simply return to the startup screen.)

Next you must pick the skill level. Press f1 to play a normal game or f3 for an expert game. At the expert level, the Whirlybird is only half its normal size, making it harder to hit the eggs. After choosing the level, the prompts disappear and play begins.

Press the fire button to release the first egg. You earn one point every time you hit a brick, and ten points every time you bounce an egg back down with the Whirlybird. Letting the egg fly past you into space subtracts ten points from your score (unless it's already under ten). When that happens, the sidebeams sparkle continuously until you press the fire button again to release the next egg. Play continues until you lose all your eggs or clear the field of bricks. A 50-point bonus is awarded for clearing every brick from the screen.

In addition to normal bricks, the playfield contains a few round ones. When the egg hits one of those, the Whirlybird swoops down to deposit a sidebeam somewhere above the playfield. This doesn't affect your score, and you get a free egg as well (press the fire button to release it). However, the extra sidebeams add an extra element of uncertainty since the egg rebounds sideways from them (not up and down) and they may or may not disappear when struck.

Whirlybird displays your current score as well as the highest score attained during the current session (which may include more than one game). When a game ends, both scores are displayed until you release the first egg in the next game. Then the current score is cleared to zero. Since the high score is stored within the program code, you can record it for future comparison by resaving the game after each session ends. (It's a good idea to resave it with a different filename—like "BIRDHIGH"—to distinguish it from the original copy, which then serves as a backup.) The next time you load and run Whirlybird, it displays the previous highest score.

The Joystick Accelerator

It may take some practice to master the joystick motion in Whirlybird. Rather than zooming immediately to the spot you want, the bird moves in an elastic manner that simulates the inertia and momentum of a natural object. It takes a bit of pushing to overcome the bird's inertia and get it moving. The longer you push in one direction, the more speed and momentum you gain (up to a point). After you let up on the stick, the bird slows down and eventually stops by itself. Hitting the sidebeam reverses your direction immediately. Reversing direction in midscreen may take a long or short time depending on your speed.

You might think of the joystick in this game as a gas pedal in a car. Keeping your foot on the accelerator makes the car move faster and faster. Let up and the car gradually slows down. Learning to control the acceleration takes some time. You may find it more effective to move the Whirlybird with short taps on the joystick, rather than long periods of acceleration.

Egg Juggling

The direction an egg bounces when it hits the Whirlybird depends on which way you're pressing the joystick. If you're not pressing it left or right, the egg rebounds as if the bird were a solid wall. If you're pressing

the stick left, the egg bounces to the left; if you're pressing right, it bounces to the right. This lets you control the direction of every bounce. But don't move the stick so much that you miss the egg completely.

The result of a bird-egg collision also depends on what part of the bird you hit. The safest strategy is to hit the egg with the bird's feet. Bouncing it off a wing is more dangerous, since the bird never stops turning. If the egg arrives when the wing is outspread, you'll usually get a normal collision. But if the Whirlybird happens to be facing sideways—making it very slim in profile—the egg may miss completely.

Occasionally a wing collision causes the bird to juggle the egg briefly. Juggling gives you extra points and may also change the angle of rebound slightly (a boon when you're short on eggs and need to clear the last few bricks). But it's also dangerous, since the bird may juggle the egg upward and out of play. Juggling occurs less frequently at the expert level; with smaller wings it's more difficult to get multiple collisions.

Rings And Filters

Whirlybird's unusual sound effects are created with two voices of the 64's sound chip. Voice 3 is used as a random number generator (to pick a random screen position and direction for each new egg) and doesn't produce any sound. Voice 1, set for a pulse waveform, makes a "boing" sound when an egg bounces, and voice 2 creates low, thrumming tones in the background with a ring modulated triangle wave. The vibrato-like effect results from ring modulated overtones that slowly pass in and out of phase with the frequencies of voices 1 and 2.

These sounds are made more interesting by passing them through a combined lowpass/bandpass filter and sweeping the filter's cutoff frequency up or down in conjunction with other game events. When you start the game or when the egg is flying freely, the cutoff frequency sweeps up and down, creating a gradual *meow-meow* effect. When the egg hits something, the cutoff frequency is set high and swept rapidly downward to accentuate the bouncing noise. ©

Prisonball

John Scarborough
Version by Kevin Mykytyn

Nearly everyone has played Pong *or* Breakout, *two computer-game classics. "Prisonball" creates an intense, two-player action game by drawing features from both of those games. Either a pair of paddles or two joysticks are required.*

"Prisonball" is a two-player action game that combines the best elements from two classic computer games, *Pong* and *Breakout*. The object of the game is simple—knock out as many bricks as you can in the allotted time.

Break To The Center

The game begins by displaying five colored walls running vertically down the center of the screen. Each player controls two paddles located at the left and right sides of the screen. Three balls appear at a random location and start bouncing around the screen. When a ball is on your side of the screen, move one of your paddles into its path to deflect the ball toward the walls. You can only hit a ball when it's moving toward your paddles (away from the interior walls). Balls travelling from the opposite direction go right through your paddles. If you happen to miss a ball, it wraps around the screen and appears on the other side, giving your opponent a chance to score.

At the beginning of the game, all three balls are a neutral color. Each time you hit a ball, it changes to the color of your paddle. You score whenever a ball of your color hits one of the five interior walls. The score depends on which wall you hit. The center wall is the hardest to reach, so it yields the most points. The two intermediate walls are worth less than the center wall. The outermost walls are easiest to hit and score the fewest points.

Every time a ball hits one of the walls, a brick is knocked out of the wall at the point of impact. By aiming your shots carefully, you can bore a path through a wall and move a ball into the interior space between two walls. When this happens, the ball bounces wildly back and forth between the walls, scoring many points in a short time.

An additional bit of strategy has to do with the redrawing of walls. Whenever a wall has been destroyed, it is immediately redrawn. Some of the highest scores result when you trap one or more balls behind a wall when it is redrawn. Since the wall is new, the trapped balls may hit it many times before they break back out to the exterior.

Either joysticks or paddles can be used to play this game. To play with paddles, plug a pair of paddles into port 2. Each game lasts for three minutes. The screen border flashes briefly as a warning when only 20 seconds remain on the timer. Bricks from the center wall are worth 30 points, those from the two adjacent walls are worth 20, and the outermost bricks each score 10 points. ©

Quickchange

Kevin Mykytyn and Mark Tuttle

You'll need a good strategy and fast reaction time to succeed in this mind boggler. Three games in one for the Commodore 64. A joystick is required.

Are you ready for a challenge? "Quickchange"—three games in one—requires logical thinking, manual dexterity, and a good strategy. If you work quickly and stay cool, you should be able to survive for a few rounds. None of the games is easy to master. The three games use the same board and have the same basic format, but each has a different twist.

Use the command LOAD "QUICKCHANGE",8,1 to load the program. When it's finished loading, enter SYS 49152. Be sure to have a joystick plugged into port 2.

You're first presented a menu with a selection of three games: Flip Flop, Missing Pieces, and No Turning Back. Move the arrow with your joystick to select a game, and press the fire button to start. It is suggested that you start with Flip Flop and progress to No Turning Back.

Flip Flop

The playing field is a 120-square grid, 8 rows by 15 columns. You're represented by the black circle starting at the center of the grid. Eight black pieces, one in each row, start at the left and move across the grid. Each moves at a different rate of speed to the far right edge and then returns, and so on. Contact with one of the black squares costs one player (you begin with three).

The object of the game is to move your piece around the grid and change the color of all the squares from pink to blue (different colors in successive levels) while avoiding the moving black squares. And all this must be accomplished in 60 seconds. Moving into one square instantly reverses its color. It's important that you choose your course carefully. Backtracking over blue squares changes them back to pink. When you complete the grid (change all squares to blue) within the allotted time, you advance to the next level. For each successive level, the speed of the black squares increases. It doesn't get any easier.

One point is awarded for each square you flip to blue. However, once a square is flipped, you do not receive any additional points for flipping it again. You lose a point for changing a square to the wrong color and gain back the point when it's changed back to the right color, so your score can move up and down. You receive 100 bonus points for each level completed. Throughout the game, current score, high score, current level, time remaining, and number of players remaining are displayed. The game can be paused at any time by pressing SHIFT or SHIFT LOCK. Release it to resume play. The game is over when you've used all three players or when time runs out.

Missing Pieces

The play in this game is nearly identical to Flip Flop. But there's a twist. Several pieces—randomly selected each time you play—are removed, thus creating empty spaces—*holes*. If you happen to leap into one while hurrying along, you lose one of your players. Unfortunately, the black squares are unaffected by these empty spaces. They move through them as if they weren't there.

The scoring and rules from Flip Flop apply to Missing Pieces.

No Turning Back

This is the most exciting—and most difficult—of the games. The playing board is the same as Flip Flop, but the object of the game is different. Once you move away from a square, it's erased. The goal is to eliminate the squares (you are credited with clearing the board if three or fewer remain).

Moving into an empty space is not allowed. But, unlike Missing Pieces, the black squares cannot move into the voids. They'll reverse direction if they encounter one of these spaces. This can be used to your advantage. You can cut off a black square by removing the squares horizontally adjacent to it. This will erase a black square and free up an entire row. For each black square removed, 100 points are added to your score (plus 1 point for the space previously occupied by the black square).

This game is trickier than the others. In haste, it's easy to isolate your own player. Remember, the object is to eliminate the squares, and you can't enter a voided space. Also, be aware that when you lose a player—provided it's not your last—your next player is positioned in the center of the screen. If there are no available pieces adjacent to the center, you've eliminated yourself from the game.

There's A Way To Win

The key to success in each of these games is to find the right pattern. There are several that will work. If you randomly move around the maze, you'll find that although you can accumulate points, you'll probably run out of time.

Quickchange may be played competitively for highest score or just for the challenge of completing a level or two. ©

Saloon Shootout

David Hensley, Jr. and Kevin Mykytyn

You'll have your hands full in this action-packed, arcade-style game. It's one of the most challenging and graphically charming games we've published. For the Commodore 64, Plus/4, and 16. A joystick is required.

If you're a sure-shot, dead-eye, trigger-happy gunslinger, you'll be suited to this wild and woolly action game. In "Saloon Shootout," the pace is frantic and it never slows up. The object of the game is to accumulate the highest number of points before time runs out or before you run out of bullets.

Of Mice And Mugs

There's a lot to contend with in Saloon Shootout. A gun, which appears on the bar, is your weapon. It's controlled by your joystick (plugged into port 2). By moving the joystick up and down, you control the distance your bullet travels. You'll notice the gun barrel shift up and down as you move the joystick. Also, an arrow on the right of the screen will help you gauge the distance your shot will fire.

A main objective of the game is to shoot as many mugs as you can before they fall off the bar. For every ten of these you shoot, a running mouse appears. Each time you shoot this critter, you receive a bonus score and it reverses direction and speeds up. Every time you hit him, the point total escalates and so does his speed. (This is where you can get a lot of points for those potential high scores.)

This may sound pretty easy so far, but your job at the saloon is much more complicated. You must also keep your eye on the three windows on the back wall of the saloon. They're sometimes opened by a masked desperado who will steal ten of your bullets unless you shoot him before he disappears. But don't be too hasty—sometimes good guys appear (the good guys smile), and shooting them results in a stiff penalty—100 points subtracted from your score. If your supply of bullets is running low, you can replenish them by shooting the cards that appear on top of the player piano. However, you must shoot the cards in order: 10, J, Q, K, and A. Be careful here—one card hit out of order will reshuffle the deck and you'll have to start over. Now you can see that this is no simple task.

There are a few features that help to calm your nerves, however. If you reach a score of 500, you receive a bonus time of 50 seconds. To help you keep track of the time remaining, the screen flashes when there are 25 seconds left. As noted above, the arrow on the right side of the screen helps make sure that those valuable bullets you're shooting are not wasted. And finally, the player piano cranks out ragtime tunes to help relieve the tension. To change the tune, shoot one of the white keys on the player piano. To turn the music off, shoot a white key twice.

Game Strategy

When the game first runs, you have a few seconds before the mugs start to appear. It's wise to use this time to shoot a few of the cards (in order, as we mentioned) in case your bullets run low during the game. Try to make the mouse appear as many times as possible to receive higher scores. When it appears, try to shoot it as many times as possible. Unless your bullet count is high, it's a good idea to set your sights on the windows as they start to open. And remember to keep an eye on your bullet supply and the time remaining.

To play Saloon Shootout, load the program using the format LOAD "SALOON",8,1 Start the game by typing SYS 10240 and pressing RETURN.

Statistics And Point Totals

Time:	**2 minutes**
Bullets:	**35**
Scoring	
Mugs:	**10**
Missed mugs:	**−5**
Cards:	**10**
Back of card:	**−10**
Good guy:	**−100**
Bad guy:	**25**
Mouse:	**20,40,60,80,100**

Props

Philip I. Nelson

"Props" is a fast-paced, nonviolent game for the 64 with six levels. Animated in machine language, it uses all eight sprites, programmed characters, and all three SID voices for sound effects.

Included in the article is a detailed program discussion which offers a variety of excellent programming tips and techniques.

You are a lonely pigeon, lost in a dangerous sky filled with whirling propellers. Your mission is to return to your coop and your mate, for a brief rest before flying away again. To make matters worse, every time you leave, and at other uncertain intervals, your mate moves to a new coop.

While in flight, you must avoid getting pulled into the propellers. If that happens, you lose points. Unless you escape quickly, the props may pull you back again and again. The props start in orderly formation, but every collision will bump one out of line; so the worse you play, the more confusing things get.

To play, plug your joystick into port 2. The six skill levels range from leisurely to manic. Whenever you reach home, your score is displayed briefly. If you press the fire button during the score display, the game pauses to let you catch your breath. During the pause, you can change to a different skill level by pressing number keys 1 through 6. To quit, just pause and press the 0 key. If you score well at any given level, the game pauses by itself and lets you pick a new skill level.

The Animation Subroutines

Two main machine language (ML) routines are responsible for virtually all the animation. The first one reads the joystick, moves your bird shape accordingly, and flaps the wings of both birds. The second rotates the eight propeller sprites and moves them up or down. Two additional small routines help program a new character set and fill color memory with white values for the new-ROM 64s.

Let's look first at the bird-moving routine (Birdmove), which you could adapt for just about any graphics game. Birdmove animates our bird-shaped character. The routine keeps track of a variable, BIRDLOC, that represents the bird's current screen location. To move the bird around in screen memory (locations 1024–2023), first we put a blank space into BIRDLOC to erase the character.

Next we check to see whether any movement has been requested through the joystick. If so, we change the BIRDLOC variable to represent the new screen location. If not, BIRDLOC stays the same. In either case, we then plant a new bird shape in the updated BIRDLOC screen location.

Setting The Bird's Boundaries

To move the bird left or right, Birdmove will subtract or add 1 to BIRDLOC. To move the bird up or down on the 40-column screen, we subtract or add 40 to BIRDLOC. Before moving our pigeon around in memory, we need safeguards to prevent the bird figure from flying above screen memory into the BASIC program space, or below it into the sensitive zero page of memory, either of which could crash the computer.

Birdmove uses two techniques to confine the bird. The first compares BIRDLOC to absolute upper and lower limits. If you try to move lower (<1024) or higher (>2023) than the bounds of screen memory, Birdmove will terminate without changing BIRDLOC.

Collision Detection

The second safeguard is a collision-checker for sideways movement. When you move left, for example, Birdmove holds the updated BIRDLOC position in temporary storage. Before it moves a bird figure into the new location, the routine checks that spot to see which of the three possible characters is there.

If the desired spot contains a space, your bird can move left. If the new spot holds the coop character, the old BIRDLOC is restored and you exit Birdmove without changing position. If neither character is found, then the spot must contain the mate character, so the routine sets a flag to show that the bird has reached home, and ends with the wing-flapping display.

To modify Birdmove for your own games, just add more comparisons to check for as many possibili-

ties as you need. For example, your game might check the desired location and then branch to appropriate routines to score if you've hit a treasure, faint if you've hit a troll, rejoice if you've bumped into a friend, and so on.

The Joystick Flags

The joystick reader at the front of Birdmove is from the *Commodore 64 Programmer's Reference Guide*. It will store flag values in a memory location which you can then PEEK to determine movement. In "Props," the joystick flag values are in the cassette buffer, but you could put them in any safe memory spot. The right/left flag is stored in location 832, and the up/down flag in 833. The value in 832 will be 255 for left, 1 for right, and 0 for no movement. The value in 833 will be 255 for up, 1 for down, and 0 for no movement. Note that leftover flag values will remain in the computer's X and Y registers, though, so if your ML program goes from this routine to one that uses indirect addressing, you should clear the X and Y registers to 0 to keep things straight.

Programmed characters are used to make the birds' wings flap. In lines 62000–63000 of Props, we first copy the character set from the ROM chip into RAM memory beginning at location 14336. Then we create new shapes for characters 90–96 by POKEing new values into the right places in our RAM character set. Character 90 is programmed to serve as our coop character, and the other six are a series of bird shapes.

Each time we call the Birdmove routine, we also flip to the next character in this wing-flapping series to create the illusion of movement. To see all the programmed characters, first RUN the program and then press the STOP key while the instructions are displayed. Hold down the SHIFT key and press CLR/HOME to blank the screen. Now type in this line. You'll have to use abbreviations to fit it all on two lines.

```
PRINT"{CLR}"TAB(255): K=90:FORJ=
  1024TO1276STEP42:POKE54272+
  J,1:POKEJ,K:K=K+1:NEXTJ
```

Press RETURN and you'll see the coop character and six bird shapes in the upper left of the screen.

Flapping From BASIC

Now let's make our bird flap its wings from BASIC in immediate mode. Type this line and then press RETURN:

```
FORK=1TO100000:FORJ=91TO96:POKE
  1024,J:FORL=1TO30:NEXTL,J,K
```

The bird should be flapping at top left. Press STOP when you've seen enough. While we're at it, let's to the same job with our ML routine. To set things up, type this line and press RETURN.

```
POKE251,0:POKE252,4:POKE834,91
```

This puts information in memory locations which the ML routine uses to position the bird and start the wing-flapping character series. Now type this line and press RETURN:

```
FORJ=1TO100000:SYS49608:FORK=
  1TO30:NEXTK,J
```

Using The Routine's Modules

As before, press STOP when you've seen enough. The entire Birdmove routine starts at location 49408 in memory, with its flap portion toward the end of the routine (49608). At certain points during Props (the reunion or a pause), we want the birds to flap their wings without moving. So we just bypass the movement parts entirely, starting at location 49608. If all we want is to place the mate somewhere, without any moving or flapping, we can jump in even later, at 49615. By structuring our ML program in distinct modules, we're able to get maximum use out of what we've written.

Now let's call the whole Birdmove routine to let our bird fly free. First, type this line and press RETURN.

```
POKE834,91:POKE835,0:POKE836,4:
  POKE837,230:POKE838,6:POKE251,
  255:POKE252,5
```

We just positioned the bird and set limits to keep it on the screen. Now enter this as one line.

```
PRINT"{CLR}":FORJ=1024TO2008
  STEP41:POKEJ,90:POKEJ+54272,1:
  NEXT:FORJ=1TO100000:SYS49408:
  NEXT
```

You'll see the bird wrap around the side of the screen when its way is clear, but stop when it hits a coop character. The up-and-down movement routine contains no collision-checker, though, so moving in those directions will erase any character you encounter.

Vary The Difficulty With Delay Loops

Running at full ML speed, Birdmove is fun to play with, but too fast to be practical. Props uses a variable delay loop (pegged to skill level) to slow things down to a manageable speed.

Spritemove, the second big ML routine in Props, handles the sprite animation, moving the eight propellers up or down at the correct speed and twirling them in unison.

Look at lines 2–6 of Props and you'll see something odd. The game works by cycling through these lines, calling the Birdmove routine over and over with the statement SYS 49408. But Spritemove is called only once (SYS 49152) in line 1, while we're setting things up. Yet the sprites move continuously as long as we're playing. How can we make Spritemove work all the time without calling it repeatedly? Easy—just let the computer do it along with its other housekeeping.

Harnessing The Hardware Interrupt

In addition to executing your programs, your computer's processor chip has continual housekeeping to do like updating timers and scanning the keyboard. But it can do only one thing at a time. So occasionally the computer stops doing your work and takes time out for its own. You never notice these *interrupts*, because they happen about 60 times every second.

Like Birdmove, the 64's hardware interrupt routine is just another ML program, starting at location 59953 ($EA31) in memory. By changing one pointer (vector), we can have the computer perform our ML routine first, then on to do its housekeeping as usual—60 times a second.

Memory locations 788–789 ($0314–0315) are specially reserved to hold the address where this interrupt routine begins. When you turn on your 64, it automatically sticks the normal (default) address in these locations. The first

part of Spritemove just changes this vector to point the computer to the beginning of our ML program.

At the end of our ML routine, we send the computer on to its normal interrupt program at $EA31, rather than returning to the program as we do in a conventional ML subroutine.

Watch It In Isolation

Such an *interrupt-driven* ML routine will seem to run independent of BASIC. To watch Spritemove in isolation, first RUN Props and press the STOP key when the props move. You'll see the blinking cursor and READY signal, which shows the computer has quit executing our BASIC program. We're back in BASIC immediate mode, but Spritemove is still working along with the interrupts, so our graphics and sound keep going.

We can do anything we'd normally do from BASIC, even call other ML subroutines as we did in the examples above, but there's a limit to how far we can take this technique. Grafting a lengthy ML routine onto our interrupts will make those "time-outs" so long that they slow our BASIC operations down to a crawl.

To stop Spritemove, first clear the screen of character graphics by holding down SHIFT and pressing CLR/HOME. Now type SYS49152 and press RETURN. The props and sound should freeze.

To restart the props, move your cursor up to the same line and press RETURN again. The interrupt vector now points to Spritemove again, and we're back in business. Spritemove is designed to alternately change and restore the interrupt vector, every time we call the routine, letting us turn it on or off at will.

The Sprites Are Still There

Note that stopping Spritemove doesn't erase the sprites. If we want them to disappear at certain points in Props, we have to disable their display with the statement POKE SP+21,0. When that's done, the sprites are all still moving in the sense that Spritemove keeps changing their location registers and shape pointers as always. But none of this is visible since we've commanded the computer not to show it on the screen.

Compared to the interrupt routine, the rest of Spritemove is simple. The BASIC setup portion of Props sets all eight sprites to fixed horizontal locations, giving each a track to run up or down in. Each prop always flies in the same direction—one space up or down on the sprite grid for every execution of Spritemove at skill level 1.

Each sprite has a register (memory location) containing its vertical location. To move the props, Spritemove increments or decrements every vertical location register one or more times, depending on skill level.

Believe it or not, this is simpler in ML than in BASIC. Let's say sprite 1 starts out at vertical location 100. If we start plopping bigger values into its vertical location register, sprite 1 will move down the screen.

Safe Increments Are Assured

In BASIC we'd have to program in a safeguard to make sure we couldn't POKE a value larger than 255 into the register, since that would abort our program with an ILLEGAL QUANTITY error.

But ML lacks the error-checking mechanics of BASIC, and simply won't let you put a number bigger than 255 into any memory cell. Trying to increment a register from 255 to 256 will just flip its value back to 0. Increment that register again, and it'll contain the value of 1, and so on.

The same thing works in reverse—decrementing a register that contains a 0 value will give us the value of 255. This characteristic of ML, which might seem a limitation, is used to advantage in Spritemove, which just keeps incrementing and decrementing the vertical sprite registers blindly. We know ML won't let us exceed the safe 0–255 range which, conveniently enough, the sprites also use for vertical location.

Animating The Propellers

You define a sprite's shape by pointing it to a block of shape information which you've placed in memory beforehand. To rotate the props, we just flip them through a series of related shapes, much as the birds are made to flap their wings. Spritemove points all eight sprites in unison to successive sets of shape data which was stored when we set up Props. Since props are bilaterally symmetrical, we can save memory space and get the effect of an eight-position rotation by flipping them repeatedly through a series of only four shapes.

Just as the computer looks in a special place to find the address of its interrupt routine, Spritemove checks and changes a special spot for the current shape pointer, location 828 (#033C).

We've used other memory registers in the cassette buffer to store things for our ML routines. Locations 832 and 833 hold values received from the joystick, as we've seen. Location 842 holds the home flag: The Birdmove routine will store a value of 1 here if the bird reaches home; otherwise, the register contains a 0.

Passing ML Values To BASIC

This is an example of how to use variables in machine language, and pass information back and forth from ML to BASIC sections of your program. In BASIC, of course, we'd name a variable something like HOME, and say that HOME=1 when home is reached, making sure that HOME=0 at all other times. But ML doesn't recognize names—just numbers inside memory locations. So, in Spritemove we choose a special memory location (842) to represent the condition of our home flag. Then we store a 1 value in 842 as a signal whenever home is reached.

Line 3 of the BASIC program uses the PEEK function to check that same memory location (HM=842) for a nonzero value, branching to the BASIC "home" subroutine at line 20 if that condition is satisfied. Once we've performed our home routine, we set the flag back to 0 in line 24, so that our bird can get lost again.

Synchronizing Sound And Action

Props also creates its filtered and ring-modulated sound effects by passing values from ML to BASIC.

When the bird flies around the screen, a soft musical tone is heard, changing constantly in relation to screen position. We start making this sound in line 2 by POKEing voice 1 on. In line 6 we change the pitch of voice 1 by PEEKing into location 251 which, you'll recall, is used by Birdmove to store our bird's screen location. In this simple way, we can link the bird's sound effect to its graphics action.

Voice 2 is always on during the game, set to the noise waveform to make a swooshing sound. The effect of fading in and out is created, not with the volume control (which affects all three voices equally), but with a filter, which we can set to affect any or all of the voices at a given time. In line 1002 of Props we POKE register 54296 with a value of 47. Besides volume, this register lets you select what *type* of filter you want. So we started with a value of 15 for maximum volume in all voices, then added 32 (15+32-47). This turns on bit 5 of the register to activate the *bandpass* filter, which will cut out all but a narrow band of frequencies in the tone of the filtered voice.

Next we have to tell the computer which of the three voices it should send through the filter. Also in line 1002, we POKE the value of 66 into register 54295, which sends voice 2 through the filter and selects a moderate amount of resonance. (If you've never played with filter resonance, try editing line 1002 to substitute the value of 226 instead of 66, to hear the more pronounced effect of maximum resonance.)

A Swoosh Is Filtered Noise

Now the filter's ready to use. Picking the noise waveform for voice 2 gives us a more or less random mishmash of all audible frequencies to work with. Setting the *cutoff* frequency low will *pass through* a narrow band of low frequency tones for a roaring or rumbling sound, and cut off all other tones. A high cutoff value gives us a narrow band of hissing, high-frequency tones. To make a swooshing sound, we just change the cutoff frequency at high speed, from low to high values.

To tie this sound to the graphics action, we let Spritemove change the cutoff frequency at ML speed. At the very end of SPRITEMOVE is a little routine that stores a value into the filter cutoff frequency register. This value is the same one used to control how many spaces the sprites move each 1/60 second. So at higher skill levels we add bigger numbers to the cutoff frequency register, to sweep the filter from low to high more rapidly.

As with sprite positioning, we can increment forever, without fussing over illegal quantity errors. What we get is a repeated low-to-high sweep in the range 0 to 255.

Filtering Voice Three

The echoing synthesizer tones heard while pausing, or when the bird's mate changes coops, are produced by applying similar bandpass filtering to voice 3. The technique is the same—we sweep the filter cutoff frequency upward, over and over. But instead of noise we're using a triangular waveform, ring-modulated by the pitch frequencies of voice 2 (line 51).

The pitch of voice 3 is linked to the bird's screen position by using the value found in location 251. And the pitch frequency of voice 2 is also swept down over and over, in our familiar 255-to-0 range, by the Spritemove routine.

Unlike the noise waveform, which contains tones at almost every audible frequency, the triangular waveform is rich in certain harmonic frequencies and totally lacking in others. So at certain frequencies the bandpass filter cuts out just about everything, causing silence. Adding ring modulation suppresses the fluty tone we'd otherwise get from a triangle wave, and adds new harmonics for an even stranger effect.

A Two-Voice Sound Effect

One final, important difference between this and the swoosh sound is in the ADSR (attack/decay/sustain/release) envelope. For the prop sound, we set voice 2's sustain value to the maximum of 240 (line 1082), and trigger the ADSR envelope only once at the beginning (line 11050).

With maximum sustain, the tone will never fade out naturally—it only seems to reach silence when our filter is set to its lowest cutoff frequencies. For contrast, we trigger the ADSR envelope for voice 3 every time we make the synthesizer sound, causing the slow, ghostly fade-out.

But you do fancy filtering without mastering ML. Take a look at lines 11050–11058, which govern the animation and sound of wings flapping during the instruction display. Here we're controlling the filter frequency from an entirely different source.

A Special Number Generator

Location 54299 (VM+3) is a very special register that can be made to produce four different number sequences which are handy for controlling sound. It can generate a 0-to-255 sweep like we've used up to now. Or it can sweep from 0 up to 255 and back down again. It can generate random numbers, and can also flip back and forth from 0 to 255 at varying rates.

You choose *which* number sequence you want by selecting one of the four waveforms for voice 3. You control the *rate* at which the numbers change within that sequence by setting the frequency of voice 3.

For a convincing wing-flapping sound, we want the filter to sweep up and then back down again. So we select the up-and-down number sequence by setting voice 3 to a value of 16 in line 11050. To time it to the beating of our birds' wings, we just fiddle with various pitch frequency values for voice 3 (H3 and L3) until we get it right. Note that you don't want to *hear* voice 3—you're only using its pitch values to control the output of voice 2. So W3 is POKEd to 16, which selects the triangle waveform without turning on the gate bit which would make the voice audible (that is, by POKEing W3 to 17).

Once you have Props working, you can learn a lot about the 64's SID chip just by changing the values used in this and other sound sections. ©

Powerball

William Chin, Editorial Programmer

This outstanding arcade-style game's moving obstacles, special features, and different game screens will keep you thinking and moving fast. For the Commodore 64. A joystick is required.

Not your ordinary version of *Breakout*, "Powerball" is an addictive, multifaceted arcade-style game with new features such as *capsules* and *slivers*—and ten different screens to master.

Each Powerball screen is populated by its own configuration of slivers, capsules, and bricks. After a few games you'll begin to develop strategies for each of the screens. As in *Breakout*, the object of the game is to destroy a series of walls brick by brick. In Powerball, this can be done in two ways—you may use your paddle to bounce balls against the wall, and you may shoot the wall down after catching a *fire* capsule. Unlike *Breakout*, you must quickly observe the characteristics of each type of screen object to earn high scores.

To play Powerball, enter LOAD "POWERBALL",8,1. Start the game with SYS 49152.

When the game begins, you'll see a paddle in the center of the screen near the bottom. Below the paddle you'll see an indication of the number of paddles remaining, your current score, and the high score of the current series of games. You begin each game with four paddles available (only one paddle can be active at any given time, however). Use a joystick in port 2 to move the paddle left and right. Press the fire button to release the ball and begin play.

Blasting Bricks

Your main weapon against the bricks is the bouncing ball. Use your paddle to keep the ball in motion. You'll lose a paddle whenever a ball gets past you to the bottom of the screen. The game ends when all paddles have been lost. When all breakable bricks on the current level have been eliminated, you advance to the next level. When you lose all your paddles, you are *not* automatically dumped back to the first level. Instead, you can press the fire button to start a new game at the current level. You can press SHIFT at any time to pause the game. (Use SHIFT-LOCK for longer pauses.)

Bricks come in three varieties: soft, hard, and solid. Soft bricks are green and are destroyed by being shot or by being hit by a ball. Hard bricks start out gray and change color each time they are hit. They must be turned green before they can be destroyed. Solid bricks are white. They can be destroyed only by a power ball. However, it's not necessary to destroy all the white bricks to advance to the next round.

Slivers And Capsules

Slivers and capsules are the special features that distingush Powerball from most other games of its type. Slivers are small flashing objects that change color as they float around the screen. Balls bounce off the slivers. This can be helpful at times (to deflect a descending ball), but, in general, slivers do more damage than good. Touching a sliver with your paddle sends the sliver to the top of the screen where it is least dangerous. There is no way to permanently destroy a sliver.

Capsules are the same shape as your paddle. They appear at random times and at random locations near the top of the screen, and then descend vertically down the screen. Touching a capsule with your paddle gives the paddle a special ability depending on the color of the capsule. (If you're using a monochrome display, each type of capsule also has a distinguishing label.) Below is a list of the capsule types and their corresponding powers:

Label	Color	Power
S	yellow	slows down all balls
C	green	paddle catches ball
L	blue	paddle becomes longer
F	red	paddle fires
P	purple	ball becomes power ball
3	lt. blue	ball splits into three balls
*	gray	extra paddle
N	orange	go to next screen

A standard paddle with no special powers is white. When a capsule is touched, the paddle takes on the color of the capsule to indicate its current power. A paddle can have only one power at a time; the previous power is lost each time a new capsule is touched. The special power is also canceled when the paddle is lost.

A few notes about the powers: When you've caught the green C capsule, your paddle will capture the ball so that you can reposition the paddle for an accurate shot. Press fire to release the ball. If you wait too long (about three seconds), the ball will be released automatically. The purple P capsule turns the balls into power balls, which destroy *all* types of bricks. After you capture a red F capsule, your paddle has the power to fire darts at the bricks as well as to deflect the ball. Press the fire button to shoot the darts. The light blue 3 capsule splits the ball into three individual balls (but only once each time one of these capsules is touched—you'll never get more than three balls). The split balls behave the same as the original ball, except that you don't lose a paddle until *all three* of the split balls have slipped past to the bottom of the screen. The gray * capsule adds an extra paddle to your total when touched. Touching an orange N capsule takes you to the next level, regardless of how many bricks remain on the current level. The gray * and the orange N capsules are rare.

Laser Beam

Mike Greenfield
Version by Kevin Mykytyn

This fast-paced arcade game is written entirely in machine language and challenges the quickest reflexes. A joystick is required.

"Laser Beam" is a fast-action arcade-style game with a simple premise—you score by moving. The more you move, the more you score. The object is to avoid several bouncing balls until just the right moment when you must grab them and stuff them into a basket.

To get started, plug a joystick into port 2 and type LOAD"LASER BEAM",8,1. Then type SYS 49152 and press RETURN.

The Highlight Zone

Move the joystick up or down to select the difficulty level—0 is the slowest and 9 is the fastest (in fact, 9 is so fast that it's almost unplayable). The game starts when you press the fire button.

Immediately you'll see seven colored balls bouncing around the screen. At the left is a highlighted "safe zone" occupied by a small stick figure—that's you. By moving the joystick, you can maneuver your figure around the screen. But if you venture out of the safe zone and bump into a bouncing ball, you're zapped. (You get three lives per game, as indicated by the figures at the bottom of the screen.)

Your goal is to render the balls harmless, grab them one by one, and drop them into the basket in the lower-right corner. To make a ball safe to touch, you have to shoot it with the laser gun. The laser is visible along the edge of the screen. To control it, first you must move your figure into the uppermost corner of the safe zone. *The laser gun is under your control only when your figure is at this spot.* Pushing the joystick to the right moves the laser clockwise around the edge of the screen, and pushing the joystick to the left moves the laser counterclockwise around the screen. Press the fire button to activate the beam. If you push the joystick in any other direction, you'll move your figure away from the top of the safe zone, and the laser gun will no longer be under your control.

Now, you can't shoot just *any* bouncing ball with the laser to make it safe to grab. You have to shoot the ball which matches the border color of the screen. As soon as you hit the ball, it turns white. Then you can maneuver your figure out of the safe zone, grab the white ball by touching it, carry it to the basket, and drop it in by pressing the fire button. Afterward you must scurry back to the safe zone before a collision with another ball.

For example, let's say the border color is red. First you move your figure to the top of the safe zone to take control of the laser gun. Next you push the joystick right or left to aim the laser at the red ball. When you have a clear shot, press the fire button to shoot the beam. If you score a hit, the red ball turns white. Then you can push the joystick in another direction to move your figure out of the safe zone. Grab the ball, stuff it into the basket by pressing the fire button, and make your escape—all while avoiding the other bouncing balls, of course. If you succeed, the border color changes to correspond to one of the remaining balls.

You continue with the process until all the balls are safely dropped into the basket. Then another round begins.

Bouncing Chaos

Sounds simple, right? Well, it's not. There are a few complications. Suppose you fire the laser and hit a bouncing ball that *doesn't* match the border color. It turns white, too. But it *isn't* safe to grab. If you touch it, you're zapped. This becomes a real problem when you accidentally shoot several of the balls and turn them white. Only one of them is safe, and you have to remember which one. It's not easy when three or more white balls are bopping all over the place.

There is an incentive for creating this chaos, however. The number of points you get for dropping a ball in the basket doubles for each white ball on the screen. If the only white ball is the one you're grabbing, you get only 5 points. If a second ball is white, you get 10 points; if a third ball is white, you get 20 points; and so on. If all seven balls are white when you drop the first one into the basket, you score 320 points.

Another complication is that your safe zone doesn't always stay safe for very long. After you shoot the ball that matches the border color, it begins shrinking from the bottom up. If you don't hurry out, you'll get zapped.

And there's yet another reason to move quickly: If you finish a round by dropping every ball into the basket before the horizontal bar at the bottom of the screen disappears, you get a 50-point bonus.

If all this action causes your brain to momentarily suffer a system crash, you can freeze everything by pressing and holding the SHIFT key. Press SHIFT LOCK to freeze the game for extended periods. Release SHIFT to resume play. ©

Bagdad

Bryan Files

Beware of the evil genies determined to knock you from your magic carpet. With a little strategy and skill, you can trap them in bottles—but work quickly before they re-emerge. A joystick is required.

Flying through Bagdad on your magic carpet, you're suddenly accosted by some rug-riding evil genies bent on knocking you off—literally and figuratively. Your only defense is to knock them off first, and this takes some tricky maneuvering. First, you must trap each genie in a bottle. Then collect the bottles before the genies have a chance to escape. If you fail to collect a bottle in time, the lid opens and your battle resumes.

The Calm Before The Storm

Ridding the city of enemy genies gives you a brief rest before the next wave of more numerous and determined genies. You begin with three lives, with a new life added for every 5000 points you earn. Only white genies are encountered during the first nine waves, but the tenth wave brings a new, more aggressive menace—the blue genies. If you manage to reach the fourteenth level, white, blue, and the dreaded purple genies emerge for a fast and furious battle.

Flying The Magic Carpet

After loading and running "Bagdad," push the joystick, which should be in port 2, forward to start the game. Move the joystick left or right to move your genie correspondingly. Press the fire button to fly up, and release it to allow gravity to pull you down. Hovering requires alternately pressing and releasing the fire button. If you reach the top of the screen, your genie will bounce, however, he will wrap around the screen at the left and right edges.

Strategic positioning is extremely important. When two genies hit at the same level, they bounce off each other. You have to use your carpet to hit the other genies. If you're at a higher level when you collide, the edge of your carpet knocks the genie off his rug. The falling genie then crawls back into his bottle, which falls to the ground (or to one of the intermediate levels).

Once in the bottle, the evil genie begins to work his way out again. The only way to get rid of him is to fly down and pick up the bottle.

If you hit a genie when you're at a lower level, you're the loser. You don't have a bottle, you simply lose one of your genies. You'll reincarnate at the bottom of the screen, providing you have at least one life remaining. During the three or four seconds the genie is flashing, he cannot be defeated by enemies. However, if no genies are nearby, you can terminate the flashing and begin play by moving the joystick.©

Arcade Baseball

Kevin Mykytyn and Mark Tuttle

Here's a computerized version of the once-popularmechanical baseball game found in the pre-electronic arcades. The only difference is you don't need any dimes or quarters to play. An exciting one- or two-player game.

Spring training is over and baseball season is under way. Here's a chance to do some swinging and pitching against a friend—or your Commodore 64. "Arcade Baseball" is modeled after the mechanical baseball arcade game that was popular before computerized games dominated the scene. But this computer game offers a few options that the mechanical versions did not: You can choose to practice or play, and you can select an opponent: another person or your computer.

A Few Choices

To play the game, load and run Arcade Baseball. First you're prompted to choose Practice (f1) or Play (f7). In practice mode, 20 pitches—a random mix of fastballs, changeups, curves, and sliders—are thrown so you can practice hitting. (To change the number of pitches thrown, change the value of RM in line 740.) No runners are displayed, and outs and runs do not accumulate. After all the pitches are thrown, you're asked again to select Practice or Play.

In Play mode, you're asked to select a one- or two-player game. Press the 1 or 2 key. If you wish to play the computer, select the one-player game. Now you're ready to begin.

The screen is divided into three sections. On the left is the playing field. Along the top of the field is a row of black holes. A batted ball will land in one of these holes, and each is marked with a result (single, double, triple, or out). To hit a home run, the ball must pass over the center of the rectangular ramp near the center of the field. (The crowd loves a home run and cheers when either team hits one.) At the bottom of the field is a specially designed bat.

The upper right corner of the screen is the scoreboard, which contains the inning, number of strikes, outs, and the current score. Players are represented as Visitor and Home. In the square below the scoreboard is a display of the baseball diamond, designed to show which bases are occupied by the team at bat.

Play Ball!

You must first choose a one- or two-player game. The one-player version—like the original arcade game—lasts for three outs. Your objective as batter is to score as many runs as possible before reaching three outs. The computer, as pitcher, randomly selects the type of pitch: fast ball, changeup (a slow moving pitch), curve (which breaks away, to the right of the plate), or slider (which breaks in, to the left of the plate). Swing at a pitch by pressing the space bar. This moves the bat left to right. Once the bat is released, you no longer have control of it, so you must time the release of the bat to meet the pitched ball. If the ball passes untouched, it counts as a strike.

Because different pitches cause the ball to travel at different speeds and in some cases change directions, timing the release of the bat is crucial to good hitting. The bat is designed as a half circle to allow you control of the angle of the batted ball. There's no such thing as a foul ball in Arcade Baseball; you can bounce the ball off the sides of the field.

When you get a hit, you'll see any movements on the base paths at the lower right of the screen. If the hit is a single, the batter advances to first base, and any other base runners move up one base. A double moves all runners up two bases, a triple three bases. A run scored is indicated by a chiming tone and an update on the scoreboard.

The two-player version offers more variety:

1. The game lasts for three innings. If the score is tied after three innings, play continues until one player wins. (The home team always gets to bat last, regardless of the score.) The visiting team is blue, the home team red.

2. You pitch as well as hit. Press one of the function keys to deliver a pitch: f1—fastball; f3—changeup; f5—slider; and f7—curveball. ©

Eagles And Gators

Philip I. Nelson, Assistant Editor

Featuring 16 sprites on a split screen, unique sound effects, and realistic animation, this two-player action game is programmed entirely in machine language. It's written for the Commodore 64, and requires at least one joystick.

In "Eagles and Gators," an intense two-player action game, even the title screen is animated. Though the Commodore 64 is ordinarily limited to 8 sprites, this game displays 16 moving sprites at all times and includes highly realistic animation as well.

The game begins with an animated title screen, asking whether you want to read the game instructions. Press Y to view the instruction screen or N to skip it. Next, you're shown the game screen and are prompted to choose one of the nine skill levels. If you've never played the game before, you'll probably want to press 1 to choose the easiest level.

Flyers Against Swimmers

When play begins, the screen is split into upper and lower color zones: sky-blue for the eight high-flying eagles and murky black for the eight gators swimming below. Each player controls a group of eight sprites. The game is written to be played with two joysticks, but you can substitute keypresses to control the gators, normally controlled by joystick 1. Press the CTRL key to move left, the 2 key to move right, and the space bar to fire.

Your sprites are in constant, realistic motion, flying or swimming up and down in their zones, rebounding when they hit the border between the two worlds. The flashing colored sprite in your group is the one controlled by your joystick. To switch control to a different sprite, move left or right.

The goal of the game is to expand your zone and push your opponents off the screen. This is done by convincing your creatures to push against the sky/water border. Press the fire button just as your flashing eagle or gator bumps the border, and it will "clamp" on and start to shove. Whenever you clamp successfully, your opponent's screen flashes red, and all of your creatures thrash momentarily with excitement.

Your zone expands whenever you have more sprites pushing than your opponent, and the game ends when one side or the other takes over the entire screen. That sounds simple, but it takes precise timing to clamp just one sprite. If you press the fire button at the wrong time, your sprite moves away from the border, regardless of its initial direction. If you press the fire button when your flashing sprite's already stuck to the border, it bounces back to the farthest end of your zone.

Your adversary can also unclamp one of your creatures by clamping one of his or her creatures directly across the border from yours. The ability to unclamp an opponent's creatures at any time adds an extra element of strategy and makes it possible to turn the tables on your opponent even when defeat seems almost certain. The game is designed to reward concentration and careful play, and to discourage wild tactics such as holding the fire button down constantly. At best, you'll prevent your creatures from approaching the border; at worst, you'll inadvertently unclamp those from your side who are already pushing.

When you win a game, the screen clears to your color, and a joyful, double-sized sprite from the winning side performs a victory dance across the screen. At this stage you can pick a new skill level for the next round or quit by pressing the Q key. After you exit, the game can be restarted by typing SYS 16384 and pressing RETURN. To stop in midplay, press RUN/STOP–RESTORE.

Displaying 16 Sprites

Space doesn't permit a detailed explanation of every routine in this program, but you may be curious about how to display more than eight sprites at a time, something often mentioned, but rarely seen in action. In order to understand how it's done, you'll need to know a little about raster displays.

Your TV or monitor makes a picture by scanning its screen with a moving electron beam called a *raster*. This is done line by line, from top to bottom, 60 times a second. The visible portion of a TV picture is made up of 200 raster lines, numbered from 50 at the top through 249 at the bottom. The same numbering scheme is used for locating sprites vertically on the screen, so raster line 100, for example, matches up with sprite vertical location 100. This correspondence between raster lines and sprite positions greatly simplifies the task of creating more than the usual number of sprites.

The Commodore 64 keeps track of the raster's vertical location in a special memory location (register) at 53266 ($D012). BASIC is too slow to make much use of it, but machine language (ML) is fast enough to let us check this register repeatedly and monitor the position of the raster beam as it scans down the screen. This information is used to create a split screen as well as to double the usual number of sprites.

Split Screens

To make the two-color split screen, we first establish an arbitrary "split" point somewhere near the midpoint of the screen (say, at line 100) and then repeatedly compare

the raster's location to that value. When the raster value is less, we know it's above the split, so we color the screen cyan to create the sky zone. When the raster value is greater, that means it has dipped below the split point, so we color the screen black for the water zone. Remember, moving downward on the screen increases the raster (or sprite) vertical location value. By changing the border color along with the screen background color, we can make the split extend the full width of the screen for heightened realism.

The same sort of monitoring makes it possible to display 16 sprites. When the raster's above the split, we display all of the 64's 8 sprites at vertical locations in the upper color zone. When it drops below the split, we display the same 8 sprites at locations in the lower zone. The two sets of sprites are never actually displayed at the same time. The program just changes their locations so fast that your eye sees 16 solid figures, without a trace of flicker.

Shadow Registers

If you stopped at this point, you'd have two identical groups of eight sprites—upper and lower—sitting motionless on a split screen. Nice, but boring. To exploit the effect in a game, we want the two sprite groups to look different and move independently. Unfortunately, the Commodore 64 still has only eight sets of sprite control registers for handling the sprites' shapes, colors, and so on. Remember, we're just flipping the same group of eight sprites back and forth at very high speed between different vertical locations. Thus, top sprite 1 doesn't just look the same as bottom sprite 1; it *is* the same sprite. The minute you change the shape or color of any sprite in the upper group, its lower twin immediately follows suit, and vice versa.

What we really need to write this game is 16 sets of sprite control registers. Commodore didn't design 16 sprites for the 64, but you can achieve much the same effect through software. Here's an outline of how it's done in this program. First, we set aside some free memory to use as "shadow" registers in place of the actual sprite control registers. Since none of the eagle or gator sprites move horizontally, we don't need to duplicate the 8 horizontal position registers. However, every sprite must be able to move freely up or down and change its shape and color at any time. Thus, for each group of 8 sprites we need 8 registers for vertical position, 8 for color, and 8 more to hold shape data pointers. Three sets of 8 make 24: double that for the second set of 8 sprites, and you now have 48 shadow registers.

To make each sprite appear in the correct spot, we must transfer the contents of each shadow register into the corresponding control register at precisely the right instant. When the raster beam is in the upper portion of the screen, we'll take data from the eagles' set of shadow registers and store it in the actual sprite control registers. When the raster creeps down below the split, we'll fetch gator data instead and plug that into the control registers.

Of course, it's not necessary to do this on a two-color split screen. This particular program changes screen colors at the split point to accentuate the division between sides. But in another application, you might prefer to leave out the color changing code. The split location is just an invisible reference point that tells you when it's time to flip the sprites from one zone into the other.

Coasting On The Interrupt

Expressed in this simple, schematic form, the job of displaying 16 sprites sounds quite simple. When the raster beam is above the midpoint, make the sprites appear in the upper area. When it's below the split, move them down into the lower area and give them new shapes and colors. The process is uncomplicated in theory. But don't bother trying to do this in BASIC, which is many times too slow to read the raster's position accurately. This program uses an interrupt-driven routine to manage the 16-sprite effect. Although interrupt handling goes beyond the scope of this article, the technique essentially wedges our custom sprite display routines into the computer's operating system along with its normal hardware-servicing routines, which do background tasks such as scanning the computer's keyboard. Conveniently, the interrupt-driven routine is executed 60 times per second, exactly the same frequency at which the screen is redrawn. As a result, the sprite display and color split are quite stable.

The advantage of driving an ML routine on the hardware interrupt is that it becomes completely automatic. Sixty times every second, the computer executes your custom routine along with its own housekeeping routines. An interrupt-driven routine seems to coast along all by itself, without slowing down other program events. In effect, we've enhanced the 64 by adding the ability to display 16 sprites, with the option of a two-color split screen. Because the custom routines operate in the background, the shadow sprite control registers seem to work exactly like the actual control registers. We can change the color, shape, or position of any of our 16 sprites by plugging different values into their shadow control registers. To move our border up or down, we just change the value for our split point, and everything else follows accordingly.

The price you pay for these new features is that the sprites can't leave their respective zones. If you try to move an eagle sprite down into the gator zone, the custom display routine automatically gives it a gator shape and color, and the sky is minus one eagle. The same thing happens in reverse if you move a gator into the upper zone. If you move any sprite so that it overlaps the border, you get a weird hybrid creature with an eagle-shaped upper body and a gator tail. It's easy to forget because the shapes look so real, but the 64 still has only 8 sprites built into hardware. Using similar techniques, it's possible to display as many as 64 sprites on the screen at once. All you need to do is divide the screen into additional zones, add extra shadow registers, and modify the display routines to handle them. However, the same general rules apply. Each group of sprites is confined to its own zone, and you may not have more than 8 sprites occupying the same screen line. ©

Face-Off

Kevin Mykytyn and Mark Tuttle

Do you like fast action and competition? Whether you play against a friend or your computer, this colorful simulation of Air Hockey offers both. For the Commodore 64 and 128 (in 64 mode). A joystick is required (two joysticks for two-player game).

If you've played Air Hockey, you already know how to play "Face-Off." Based on ice hockey, Air Hockey pits two players against each other, each trying to shoot a puck into his opponent's goal while defending his own goal. A center line splits the playfield. Neither player is allowed to cross this line. The object of the game is simple: The first player to score five goals wins.

To play the game, load it and type RUN. You'll see a message (READING DATA—PLEASE WAIT) and after a brief pause, you'll see a prompt, 1 OR 2 PLAYERS. Press 1 or 2 (it's not necessary to press RETURN). The one-player game pits you against your computer; the two-player game is for two human opponents and requires two joysticks.

Next, you're prompted to select the speed (1–3). Speed 1 is the slowest and 3 is the fastest. (It is suggested that you play your first game at the slowest speed, although speed 2 is probably the one you'll choose after playing a few times.) If you selected a two-player option, the game begins after you've selected a speed. If you chose the one-player option, there's one more prompt to answer: Skill Level (1–9). This determines the intelligence of the computer-controlled player. If you choose 1, the computer plays a pretty easy game. At 9, it's very wily. With a speed of 3 and a skill level of 9, the computer is next to impossible to beat.

When the game begins, you see a red puck, a cyan player on the left, and a yellow player on the right. The cyan player controls the puck to start. Contact with the puck starts the game. (Notice that the puck gradually slows down if it's not hit.) After each goal scored, the player scored against gets control. (You can knock the puck into your own goal, which awards a point to your opponent.) A total of nine pucks are placed per game. If you wish to change the number of pucks, change the value in line 510.

One Player Or Two?

When playing against the computer, plug a joystick into port 1. You control the yellow player, on the right. You can move anywhere up to the center line. (At slower speeds, one strategy is to play along the center line, like rushing the net in tennis. This keeps the puck in the computer player's territory most of the time if you can react quickly enough.) Current scores for each player are posted at the top of the screen. After nine pucks are used, the game is over. You're then prompted to press the fire button to play again, then to press up on the joystick to change play options or down to play with those of the previous game.

The two-player game has the same rules. The cyan player, on the left, must use a joystick plugged into port 2.

Space Arena

Bryan Files

This engaging two-player game, which could have been titled "Indirect Aggression" for its unique design, features superior graphics and playability. For the Commodore 64 and 128 (in 64 mode). Two joysticks are required.

In a distant galaxy, a fierce war between two great empires has reached a stalemate. The cost to both civilizations has been high with no gain to either side. With the assistance of the Timelords, the two sides have agreed to settle the dispute in a controlled environment—a Space Arena.

Enclosed in this arena are four asteroids and a ship representing each side. Both ships are equipped with photon blasters, which have the ability to push objects without damaging them. A ship is destroyed only by a collision with an asteroid. The highest score within a given time determines the victor. One point is given each time a pilot hits the opposing ship with a photon blast. If the enemy's ship is destroyed by running into an asteroid, 20 points are awarded.

In "Space Arena," two people compete with their joystick-controlled ships. The movement is much like the classic arcade game *Asteroids*. Rotate the ship by moving the joystick left or right. Push the joystick forward to activate the ship's thrusters, and press the fire button to release a photon blast. Remember that you have no brakes; to slow down, you must turn the ship around and thrust in the direction you're moving. Game options include speed (1–3, where 1 is the slowest and 3 the fastest) and time limit (1–5 minutes).

There are two basic strategies to playing Space Arena. You can try to push your opponent into an asteroid, or push the asteroids into your opponent. The effect of photon blasts on asteroids is much less than on ships because the asteroids are heavier. A single photon blast can send a ship flying across the screen, especially if the ship has to turn around before being able to slow down. On the other hand, if your opponent is faced with several fast moving asteroids, he may crash into one on his own.

To load the game, type LOAD "SPACE ARENA",8,1. To run it, type SYS 49152.

Kicker

John Krause and Mark Tuttle

There's no such thing as luck in this fast-paced two-player soccer simulation for the Commodore 64 and 128 (in 64 mode). Two joysticks are required.

Defense is just as important as offense in "Kicker," a two-player soccer simulation in which each player controls nine men. You need quick reflexes to protect your goal and to drive the ball toward your opponent's goal.

Speed And Endurance

Plug a joystick in each port. Before beginning play, you must select a speed. You're prompted to press S, M, or F (Slow, Medium, or Fast). Slow is the recommended level to start with so that you can get a good feel for the game, although the medium speed is probably the level you'll choose most often. Here, the speed is competitive and your wrists may last for the entire game. The Fast level should be attempted only by diehard Kicker fanatics.

Player one, the blue team, and player two, the red team, control three rows of three men each. But each player can access only one row at a time. Move the joystick up to select the top row, down to select the bottom row, or leave it centered for the middle row.

After choosing a row, you can move the joystick left or right to shift the entire row and get your men into position to kick the ball. When the ball comes near one of your men, press the fire button to kick the ball in the direction of your opponent's goal. (The game is designed so that you can kick the ball only in the direction of your opponent's goal. You can't accidentally kick the ball into your own goal.) Kicking the ball requires split-second timing. Holding down the fire button won't do. You must press the fire button at the moment the ball comes in contact with your player. Otherwise the ball will continue past your player. Remember, selecting a row will not allow you to kick the ball. You must activate the row by pressing the fire button.

When a row is inactive (the fire button is not pressed or you do not control that row) the men are represented by three dots. Use the row of men nearest your goal to block shots by your opponent. The scores of both players are displayed at the top of the screen. After each goal, a bell sounds and a red or blue light, representing the color of the scoring team, flashes in the goal area. The ball is placed in the center of the field at the beginning of the game and after each goal. The first player to score nine goals is the winner.

Heat Seeker

Jeff Wolverton

Your jet climbs upward to avoid the missile, then dives for the ground. But it's still on your tail. You can't shake a heat seeker. This fast-action game requires a joystick.

Heat seeking missiles are dangerous. They sense the heat from your jet engine and home in on you. They'll catch you, too—they're faster than a jet.

Your assignment: Eliminate the heat seeker base. It's easy enough to strafe the missiles on the ground, but if any are launched, you'll have to take evasive action.

Piloting The Jet

Use the joystick to control the movement of the plane. The controls may seem to be a little confusing at first. You pull back to loop upwards (counterclockwise) and push forward to loop down (clockwise), like a real airplane. The jet moves at a constant velocity—you can't speed up or slow down. Press the fire button to launch a missile at the heat seekers on the ground.

It does no good to fire at a moving heat seeker. They're equipped with an Improved Electronic Evasion (IEE) circuit which makes them impossible to hit. The only way to get rid of a seeker is to make it crash into the ground. When you're being pursued, dive for the ground and pull up at the last second. Seekers are faster, but they can't turn as quickly.

If you manage to eliminate all the heat seekers, you get to start all over again, with a new group of heat seekers. You have eight jets to work with—the number remaining is displayed on the screen, next to the score. If your jet is destroyed, all missiles reappear.

To pause the game, press SHIFT/LOCK.

The game includes an option for a two-player mode, but it's not competitive: The players take turns flying the plane, trying for the highest possible score. The game reads both joystick ports, so if you're using two joysticks, the inactive player should put down the joystick to avoid interfering.

There are three levels of difficulty: Novice, Intermediate, and Expert. The higher levels have faster action and tighter curves. A flight-time bonus of ten points is awarded every few seconds, just for staying in the air. ©

The Fast Assembler

Yves Han

Here's a truly amazing machine language assembler for the 64 and 128 (in 64 mode). "Fast Assembler" supports multiple statement lines, labels, and macro-like "include" files. It can assemble to memory or to disk. Written very compactly, it occupies only about 2600 bytes, leaving the rest of memory for your source code. It also adds to the BASIC editor several new features useful to both BASIC and machine language programmers.

Symbolic label-based assemblers are the most convenient way to write machine language (ML) programs. The instructions are entered as *source code* and later assembled into object code (the actual ML program—the numbers in memory). And rather than using memory locations, you can name routines with meaningful labels. It's as if you could enter GOSUB JOYSTICK in BASIC.

Saving Memory By Using The BASIC Editor

You write your ML programs for "The Fast Assembler" (FA) with the 64's BASIC editor. You save to tape or disk as you would a BASIC program, and listing it to a printer is exactly the same as listing BASIC.

The FA is an extension of the BASIC interpreter especially designed for writing programs in machine language. Writing it as a BASIC extension kept the program short (under 2600 bytes) because many subroutines of the BASIC interpreter could be used. Some modifications have been made to BASIC to make writing programs easier. To do this, the BASIC ROM had to be copied to its matching RAM.

Even if you don't write programs in machine language, you can still use the assembler because of the new features added to BASIC and the extra BASIC commands. The assembler will execute a BASIC program just like normal BASIC would.

To start up FA, first load it as if it were a BASIC program (don't use a secondary address of 1, just type LOAD "Fast Assembler",8. Then type RUN. The enabling SYS is built into the first line of the program. The screen will clear, and a message will appear at the top of the screen, indicating FA has been enabled. You can now start programming—in BASIC or machine language.

BASIC Modifications And Enhancements

The following changes have been made to the BASIC interpreter:

• **Structured listings.** Spaces between the line number and the first character on the line are not deleted. This makes it possible to indent lines and make listings easier to read.

• **List pause.** You can freeze a listing by holding down the SHIFT key or pressing SHIFT-LOCK. Listing can be continued by releasing the SHIFT key.

• **ASCII translations and hexadecimal/binary numbers.** In arithmetic expressions, you can use hexadecimal and binary numbers. Hexadecimal numbers should be preceded by "$" and binary numbers by "%". You can also use a character preceded by a single quote ('A is the same as ASC("A")). You can also use this to find the value of a BASIC token. For example, PRINT 'END will print the value 128, which is the BASIC code for END. If you put a space between the quote and the character, the ASCII value of the space will be taken instead of the character.

• **Variable and function names.** The rules for variable and function names have been changed a little bit. Instead of the first two, the first eight characters are recognized. FA recognizes NUMBER1 and NUMBER2 as separate variables, while ordinary BASIC would consider them the same variable (NU). Variables may contain but not be equal to BASIC/assembler commands or mnemonics: LAND is a legitimate variable name, even though it contains the keyword AND. But variable labels starting with TI or ST (reserved keywords) are not automatically set to zero the first time you use them. An exception to the eight character names is that only the first two characters of

array variables are significant.

• **Keywords.** Because variable and function names may contain keywords, FA has to be able to decide whether a keyword is a keyword or part of a variable or function name. So the assembler recognizes a keyword if it's followed by a space or nonalphabetic character. For example, in PRINT "OK" the keyword PRINT will be recognized as a PRINT command, but in A$="OK":PRINTA$, the keyword PRINT is recognized as part of the variable name PRINTA$. You would have to insert a space (PRINT A$) if you wanted to print the variable A$.

• **REM and DATA.** Capital letters in REM and DATA lines are listed as capital letters and not as tokenized BASIC keywords. For example, **10 rem AB** lists as it is entered and not as **10 rem atnpeek** as normal BASIC would do.

New BASIC Commands

AUTO ***step value***

This command turns automatic line numbering on and defines the step value between the line numbers. To enter AUTO mode, type AUTO followed by the step value and press RETURN. Then enter a line with a line number. The next line number prints automatically. To leave auto mode, move the cursor to an empty line and press RETURN. To turn automatic line numbering off altogether, enter AUTO only.

You can also use this command to delete part of a program. Turn automatic line numbering on with a step value of one. Type the number of the first line you want to delete and press RETURN. Keep pressing RETURN until you've reached the end of the section you want to delete. Instead of pressing RETURN again and again, you can enter **POKE 650,128** and hold RETURN down until you've reached the last line to be erased.

OLD

If you accidentally type NEW, you can restore your program with this command. It can also be used if you've installed a reset button. If you've assembled a program and are testing it, sometimes your computer locks up. Use the reset button and then enter SYS 4408 to restart the assembler and type OLD to restore the source program. If your program has not destroyed the assembler or the source program, everything will be there.

Semicolon (;)

This has the same function as the REM statement. It need not be separated with a colon from the preceding command. For example:

```
10 X=0:REM SET X TO ZERO
```

is the same as

```
10 X=0;SET X TO ZERO
```

The semicolon in the commands PRINT and INPUT is not treated as a REM statement but as a separator.

Using Labels As Variables And Addresses

Label names follow the same rules as variable names. They can be used in arithmetic expressions like normal variables. You can define a label in two ways:

You can place the label name just before the command to which you want to refer. If more commands are on the same line, you must separate the label from the commands with a colon.

Or you can label the current program counter: **LABELNAME=***. The asterisk (*) is a special variable which gives the value of the program counter. The counter is the address where the next instruction or datum will be placed. You can only read the variable *. You cannot assign a value to it with the statement *=expr.

Here's an example of using labels to mark routines in a program (don't type this in, it's only a fragment of a program):

```
50 JSR DISPLAY1; JUMP TO LABELED
   SUBROUTINE (LINE 90)
60 LDA $FF: BNE SKIPIT ; CONDI-
   TIONAL BRANCH AHEAD TO
   SKIPIT
70 TYA
80 SKIPIT: LDX #4: STA $8000,X: RTS;
   TARGET OF BRANCH IN 60
90 DISPLAY1=* ; THIS LABELS THE
   CURRENT PROGRAM COUNTER
100 ;
110 LDA #65: JSR $FFD2: RTS
```

Remember that in the lines above, the semicolon marks the beginning of a comment which, like a REM, is ignored by FA. The technique in line 90 is valuable if you think you may be adding some code at the beginning of the routine. As listed, the subroutine called DISPLAY1 starts with LDA #65, but later you could go in and add some lines between 90 and 110.

Three Passes To Assemble

Three passes are required to assemble source code (what you write) into object code (an executable ML program that the computer can follow). But FA doesn't do it by itself. You have to insert a loop that repeats three times with BASIC commands:

```
10 FOR PASS=1 TO 3
   .
   . (Insert source code)
   .
90 NEXT PASS:END
```

If you use an invalid addressing mode such as LSR (expr),y you'll see ILLEGAL ADDRESSING-MODE ERROR. Mnemonics can only be used in program mode—that is, in a program you execute with RUN. If you enter a mnemonic in direct mode, you'll see ILLEGAL DIRECT ERROR.

Also note that for Immediate Addressing, the argument can be an actual number or an arithmetic expression with a value in the range 0–255. Or you can substitute a string expression, in which case the assembler takes the ASCII value of the first character as the argument. If the string length is zero, the argument becomes zero.

Assembler Commands

Assembler commands which write data to the output device can only be used in program mode, otherwise you'll get ILLEGAL DIRECT ERROR. All assembler commands must be included in every pass.

ORG ***address,mode,device,name***

This command must be used at the start of each pass. It does several things. First, it sets the origin (ORG), the memory address for the beginning of the ML program. It assigns an initial value to the program counter. It also sets the assembler mode, which should be zero on the first two passes and one on the third and last. ORG also sets the output device and filename (if necessary).

Not all arguments are necessary. Also permitted are:

ORG
ORG address
ORG address,mode

Default values for the arguments are:

address = 49152 (=$C000)
mode = 0
device = 0 and no name

If you use a mnemonic or assembler command before you've used the command ORG, you'll see UNDEF'D LOCATION COUNTER ERROR.

The address assigns a value to the program counter. Usually, you use more than one pass to assemble the source program. Only during the last pass should the object code be written to memory or to the output device. Mode tells the assembler when the last pass is reached. Zero means it's not the last pass, so no object code should be produced, and there's no range checking for arguments and no checking for too large branches.

On the final pass, you should set the mode to one, which signals the last pass, when object code is written to the output device.

Finally, you set the device number of the output device and a string expression which contains the filename if the object code is not written to memory. Zero means the output device is memory. Be careful not to write to memory locations where the assembler is placed ($0801–$121B) or where the BASIC interpreter is placed ($A000–$BFFF).

A device number in the range 8–11 means the output device is a disk drive. If mode is equal to one, the assembler will open a PRG file with the name specified in the argument name. The logical file number will be eight.

BYTE ***expression,expression,...***
This command writes numbers or characters to memory or the selected output device. It can have one or more arithmetic or string expressions separated by commas. Arithmetic expressions must give a positive value less than 256. The value will be placed in one byte. Each character of a string expression will be placed in one byte.

WORD ***expression,expression,...***
This has the same function as BYTE except that values of arithmetic expressions must be positive and less than 65536. The value will be placed in two bytes in low/high format.

INCLUDE ***name,device***
This command assembles a file from disk and inserts the resulting object code into memory or the output device. The file must be a normal PRG file and may not contain BASIC commands which cause a branch to another line or stop the program. Also not permitted are the BASIC commands DEF, RETURN, CLR, NEW, and the assembler commands SEND and INCLUDE.

The file is opened with a logical file number of nine. The file is closed when the end of the file is reached. The name is the filename you're including, and the device number can be 8–11 (use 8 if you have a single drive). If you have only one disk drive and you assemble to disk, the file(s) for the command INCLUDE must be on the same disk to which you assemble.

All variables and labels are *global*, which means you can pass parameters to INCLUDE files so they can work like macro-instructions. Let's say you're writing a program that needs to access several different disk files, and there are several points in the program that use the Kernal routines SETLFS, SETNAM, and OPEN. You could write the source code that performs these Kernal calls and save it to disk under the program name "OPEN" to be used later. Then, in the main program, use INCLUDE "OPEN",8. When the source code is compiled, the series of commands from the OPEN file are automatically inserted in the proper place in the object code.

SEND ***stringexpr***
The command SEND may be used only if the object program is written to disk. It's used to link object code to a BASIC program. *Stringexpr* must contain a BASIC line with line number. If you forget the line number, you'll get MISSING LINE NUMBER ERROR. If you want to send more than one line, you must use SEND for each line, and you have to send the lines in the right order. You must send the lines before the actual object code is written to disk. The address in the ORG command must be the start of BASIC RAM (2049).

UNSEND
If you load a program which consists of both BASIC and ML, the interpreter has to know where the BASIC part ends. UNSEND places a mark which the computer recognizes as the end of the BASIC part.

Example Programs

```
100 FOR PASS=1 TO 3:PRINT
    "PASS"PASS
110 ORG $C000
120 IF PAS=3 THEN OFG $C000,1
130 START: LDX #0
140 LOOP: LDA TEXT,X:PRINT TEXT,
150 BEQ EXIT
160 JSR $FFD2
170 INX
180 BNE LOOP
190 EXIT: RTS
200 PRINT *
210 TEXT: BYTE "EXAMPLE 1",0
220 NEXT PASS:END
```

Lines 110 and 120 show how to use the command ORG. In every pass, line 110 sets mode 0. But in pass three, line 120 sets mode 1. The object code will start at 49152 (hexadecimal $C000). Line 200 prints the current value of the location counter (*).

You can assemble the program with the command RUN. The program will give the following output:

```
PASS 1    0        49165
PASS 2    49165    49166
PASS 3    49166    49166
```

The first column is the pass number. The second column is the value of the label TEXT in the instruction LDA TEXT,X in line 140. The third column is the value the label should have when the source code is assembled. You can see that only in pass three are these values equal to each other. This is because the assembler defaults to zero-page addressing. In pass one, TEXT has a value less than 256 so zero-page addressing is assumed. This means a two-byte instruction instead of three. The value assigned to TEXT will be too low, as you can see in pass one. In pass two, this value, which is too low, will be used in assembling line 140. The assembler decides not to use zero-page addressing, so TEXT is assigned the correct value. In pass three, the cor-

rect value replaces the previously incorrect values during assembly.

```
5 ; EXAMPLE PROGRAM 2
6 ;
10 PRINT CHR$(147)
11 DEF FN H(X)=INT(X/256)
12 DEF FN L(X)=X-256*FN H(X)
20 PRINT:PRINT" Loader maker"
30 PRINT:PRINT" Enter the name of the
   program that"
40 PRINT" has to be loaded by the
   loader."
50 INPUT" >";NAME$
60 PRINT:PRINT" Enter the name of the
   loader."
70 INPUT" >";N$
80 PRINT:PRINT" Enter the address to
   execute the"
90 PRINT" program."
100 INPUT" >";ADDRESS:ADDRESS=
    ADDRESS-1
105 ;
110 FOR PASS=1 TO 3
115 ;
120 ORG 2049
130 IF PASS=3 THEN ORG 2049,1,8,N$
135 ;
140 SEND "10 SYS+STR$(LOADER)
150 UNSEND
155 ;
160  LOADER:  LDA #8:TAX:LDY #1
170            JSR $FFBA
180            LDX #FN L(NAME)
190            LDY #FN H(NAME)
200            LDA #LEN(NAME$)
210            JSR $FFBD
220            LDA #FN H(ADDRESS):
               PHA
230            LDA #FN L(ADDRESS):
               PHA
240            LDA #0:JMP $55D5
250  NAME:     BYTE NAME$
255  ;
260  NEXT PASS:CLOSE 8:END
```

The above example program shows how to use the commands SEND and UNSEND to write a program that includes a SYS within a BASIC line.

The main routine at 160–250 illustrates how to load another program from an ML program. Note that the lines up to 100 are BASIC; they prepare the variables and defined functions for use in the source code. If you assemble the program with the command RUN, you'll get a program that can load another ML program from disk and execute it. The object code will be written to disk.

In line 140, the command SEND writes a BASIC line to the output device by which you can load and run the program as if it were a normal BASIC program. Line 150 marks the end of the BASIC part of the object code.

The INPUTs in lines 50, 70, and 100 permit you to enter the parameters for the object program when the source program is assembled. In this way you can make different object programs with one source program.

Another advantage of writing the assembler as a BASIC extension is that you can assemble a program to the top of memory. Use the following construction to do this:

```
100 POKE 56,PEEK(56)-4:CLR
110 TOPOFMEM=PEEK(55)+256*(PEEK
    (56)+4)
120 ADDRESS=0:MODE=0
130 FOR PASS=1 TO 3
140 ORG ADDRESS
150 IF PASS=3 THEN ORG ADDRESS,
    MODE
     .
     .
    . Source code
     .
     .
900 NEXT PASS
910 IF MODE=1 THEN END
920 ADDRESS=TOPOFMEM-*
930 MODE=1:GOTO 130
```

In this program, the source code goes through six passes. During the first three passes the location counter remains at zero. Mode 0 is used so the object program will not be written to the output device. The length of the program is calculated and subtracted from TOPOFMEM. This address is used in the second three passes to assemble to the top of memory. MODE is set to one so the assembler will write the object code to the output device during the sixth pass (actually pass three of the second time around). Line 100 is used to reserve 1K at the top of memory for the object program.

Large Programs

If your source program won't fit into memory, you can split your program and use the command INCLUDE. For example:

```
10 FOR PASS=1 TO 3
20 ORG ADDRESS
30 IF PASS=3 THEN ORG ADDRESS,1
   .
   . Part 1 of source code
   .
90 INCLUDE "PART 2",8
100 INCLUDE "PART 3",8
110 NEXT PASS:END
```

The labels and variables used in the INCLUDE files will be global variables, which means you can use them in arithmetic expressions everywhere in the program.

Another possibility is chaining the programs, but then you can't use a FOR-NEXT loop for the passes. You must use another way to define the passes. For example:

```
FIRSTPROGRAM
10 PASS=PASS+1:IF PASS=4 THEN
   END
20 ORG ADDRESS
30 IF PASS=3 THEN ORG ADDRESS,1
   .
   . Source code
   .
90 LOAD"SECONDPROGRAM",8
SECONDPROGRAM
   .
   . Source code
   .
90 LOAD"FIRSTPROGRAM",8
```

Note that these are just examples. You'd have to insert your own source code as indicated. To chain programs, you would load and execute the first program. It controls the number of passes and loads the next program. The next program loads the following program and so on until the last program, which must load the first again. ©

Disk Editor

Kevin Mykytyn

For more advanced readers, here's a utility that offers power and flexibility unavailable with BASIC. With a few commands, you can easily examine and alter any byte on a disk.

You probably use your disk drive mainly for saving and loading BASIC programs, occasionally dabbling in sequential and relative files. BASIC provides simple commands for using each type of file: SAVE, LOAD, PRINT#, etc. When you type SAVE, for example, you don't have to worry about what instructions to give the disk drive to make it write your program on the disk. BASIC works together with the Disk Operating System (DOS) to automatically perform all the necessary tasks.

The disadvantage of using these simple commands instead of accessing the disk directly is lack of control and power. For example, DOS has a scratch command, but not an unscratch command. So if you accidently scratch a file, there's no way to bring it back, even though all the information is still on the disk. And DOS has the ability to lock a file so that it can't be scratched, but no command is available for using this feature.

Accessing the disk directly gives you much more power and flexibility. DOS has several low-level commands for this purpose. The Block Read and Block Write commands enable you to change the contents of an individual block or sector. But these commands are difficult to use.

"Disk Editor" makes it easy to examine and alter any byte on the disk. To run it, type LOAD "DISK EDITOR",8,1 and NEW. Then insert the disk you want to edit and type SYS 12000. *Note: Before using one of your regular disks, test Disk Editor with a new disk containing copies of a few programs.*

How To Use It

To use Disk Editor effectively, it's important to understand how a disk is organized. A Commodore disk is divided into 35 tracks, each containing a number of sectors (see table 1). Each sector contains 256 bytes. A disk has a total of 683 sectors, but only 664 are available for data storage because the directory occupies all 19 sectors of track 18 (track $12 in hexadecimal).

Disk Editor begins by reading track $12, sector 0, and displaying the 256 bytes in that sector. (All numbers in Disk Editor are in hexadecimal for better screen formatting.) The editing cursor appears in the upper-left corner. Move it around the screen with the cursor keys. The sector currently being displayed is indicated at the top of the screen along with the track number and the current byte number within the sector. Notice how the byte number changes as you move the cursor. Press the HOME key to quickly position the cursor at the upper-left corner.

To make a change, move the cursor to the desired byte and type in the new value. This doesn't change the actual byte on the disk. When you've made all the changes you want on that sector, press CTRL-W to write the sector to the disk.

To display another sector, press + and type the new track number at the top of the screen. Press − to change the sector number. The INST/DEL key allows you to correct mistakes. Then press CTRL-R to read the sector and display it on the screen. If a track or sector number is outside the range shown in Table 1, Disk Editor prints ILLEGAL TRACK OR SECTOR and cancels the CTRL-R command.

Table 1: Number of sectors per track

Track	Number of sectors
1–17	21
18–24	19
25–30	18
31–35	17

Disk Editor checks for errors on all commands. If an error occurs, a message is printed and the command is aborted. Press any key to continue.

To quickly get to an adjacent sector, hold down SHIFT and press + or −. Pressing + with SHIFT displays the next highest sector, and pressing − with SHIFT displays the previous sector.

Another way to move around the disk is CTRL-J (Jump), which

uses the number under the cursor as the track and the number in the next byte as the sector. This is handy for jumping to chained sectors.

A program on disk is stored not in sequential sectors, but in sectors scattered about the disk. To link all these sectors together, the first two bytes of each sector contain the track and sector numbers of the next sector in the chain. The last sector is flagged by a zero in the first byte. The second byte contains the number of bytes used in the sector. To jump to a chained sector, press HOME to put the cursor on the link bytes, then press CTRL-J.

When examining sectors containing a BASIC program or a text file, it's helpful to display the bytes as screen code or ASCII characters. CTRL-A toggles between screen codes and hexadecimal format. When you're in screen code format, pressing the Commodore and SHIFT keys together toggles between ASCII and screen code format. *SpeedScript* and some other word processors store text as screen codes, while *EasyScript* saves documents as ASCII codes.

CTRL-$ displays the normal directory. Hold down SHIFT to pause the scrolling. To exit the program, press CTRL-X. It also closes all files, and initializes the disk.

Changing The BAM

Bytes $4–8F of sector 0 are the Block Allocation Map (BAM). The BAM is divided into 35 groups of four bytes each, one group for each track. The first byte of each group is the number of free sectors on that track. Each bit in the other three bytes represents a sector on the track. If a bit is on, it means the sector is free. If it's off, the sector is used.

Track $12, the directory, contains all the information about how the files are organized on the disk. By changing a few bytes in this track, you can perform some neat tricks. Sector 0 contains the Block Allocation Map (BAM), and the other sectors store the directory entries and are linked by the first two bytes in each sector.

Bytes $2–1F of sector 1 contain information about the first file in the directory. Byte 2 is the file type. A value of $82, for example, indicates a PRG file. By setting bit 6 of the file type, you can lock the file to keep it from being scratched. Simply change the 8 to a C and remember to write the sector to the disk with CTRL-W. If you display the directory with CTRL-$, you should see a < character next to the file type, indicating a locked file.

Bytes 3 and 4 contain the track and sector numbers of the first sector that the file is stored. You can view the contents of the file by positioning the cursor on byte 3 and pressing CTRL-J. Press CTRL-J again to jump to the next sector in the chain, and so on.

Recovering A Scratched File

If you should accidentally scratch a file, you can unscratch it with Disk Editor. Follow these steps:

1. Don't save anything to the disk with the scratched file; you may overwrite the sectors containing the program or data.

2. Load Disk Editor, insert the disk to be altered, and SYS 12000.

3. Display track $12, sector $00. Press CLR/HOME to go to the top and then CTRL-J (to get to the first track of directory entries).

4. Switch from hex to ASCII, to look at the filenames, which should be at locations $05, $25, $45, and so on. If the file to be unscratched is not in the sector, press CLR/HOME to go to the first two bytes and then CTRL-J, to jump to the next sector.

5. When you've located the correct directory entry, look three bytes to the left—$02, $22, $42, etc. A $00 in this location marks a scratched file. Change it to $82 to make it a program file, $81 for a sequential file.

6. Write the sector back to the disk with CTRL-W and exit with CTRL-X. To make sure the BAM is correctly updated, you should validate the disk with the line OPEN 15,8,15,"V0". This process may take some time, but when it's done, the file has been restored. ©

Table 2: Disk Editor Commands

CTRL-A	Alternate between hexadecimal and screen code.
CTRL-J	Jump to track and sector under cursor.
CTRL-R	Read track and sector at top of screen.
CTRL-W	Write to track and sector at top of screen.
CTRL-X	Exit program.
CTRL-$	Display directory (SHIFT to pause).
SHIFT-Commodore	Toggle between screen code and ASCII.
SHIFT-+	Display next sector.
SHIFT-–	Display previous sector.
+	Set track number.
–	Set sector number.
INST/DEL	Delete a character when entering track or sector number.
HOME	Position cursor at upper-left corner.
Cursor keys	Move cursor.

Fast File Copier

Ross Ouwinga

This handy disk utility does more than just back up files; it also scratches, renames, formats, and validates. It's especially helpful when you're reorganizing your disk library. For the 64 and 1541. Also works on a 128 (in 64 mode) and the 1571.

When you find yourself loading and listing 50 (or more) disk directories just to find one program, you know it's time to start putting your disks in some sort of order. But copying programs takes a lot of time, especially if you have to load and save programs one at a time to transfer them from one disk to another. There are much more interesting things to do with your computer than loading and saving. It gets worse if you have sequential files rather than program files. "Fast File Copier" makes it easier to make single or multiple backups, eliminating much of the burden of disk organization.

Fast File Copier displays all the files in the directory; you select any or all of them by merely moving the cursor around the screen. The program loads as many files as possible into memory, prompts you to insert a destination disk, and then saves the programs or files to the backup disk. If all files fit in memory the first time around, you're done. Otherwise, follow the prompts until the entire procedure is completed. There's no need to remember the names of the files or how to spell them, and you don't even have to know what type of file it is or where it's located. Fast File Copier handles everything for you. Fast File Copier does more than just copying. It can also delete selected files, rename files, and validate or format a disk. There are menus and prompts for every function so there is no need to memorize anything to use the program. Just load, run, and follow the menus.

Load and run Fast File Copier just as you would any BASIC program. The title screen will appear and prompt you to insert the source disk. Press any key when you're ready to get started. The next thing you should see is the directory, unless there are no files on the disk or the disk is unformatted. In these cases, the type of problem will be displayed, and you'll be allowed to quit, insert another disk, or format the disk.

The Directory

The directory screen makes this program easy to use. At the top of the screen is the menu of the options you can use. Below this menu you'll see the disk directory split into two columns. Up to 32 titles may be displayed on this screen at a time. If there are more than 32 files on the disk, press f7 and the next page of titles will be displayed. To return to the previous page, press f5. There may be several pages since the directory has room for up to 144 files on a disk.

If you look closely at the listing of titles, you may notice that some of the titles have a comma and the letter *S*, *U*, or *R* after the filename. This indicates that a file is a sequential, user, or relative type. Program files are unmarked. This feature is only for your reference; it does not affect how you use any of the functions in this program. Don't bother trying to copy relative files. They're considerably different from other types of files and cannot be copied with the techniques used by the Fast File Copier.

The only other unusual item that may appear in the directory is a title with a large dot at the end. This indicates that the file is unclosed (a "poison" file, also called a "splat" file) and you will not be allowed to delete or copy it. If an unclosed file appears on a disk, you should eliminate it by validating the disk. If you don't eliminate an unclosed file, serious problems may result in the directory. To validate the disk, press f2.

Copying Files

Before copying a file, you must mark it so the Fast File Copier knows which one to copy. Using the CRSR keys, move the X-shaped cursor in front of the file to be copied. When the cursor is positioned in front of the desired title, press the space bar and the title will be changed to reverse characters. If you change your mind, press the space bar again and the title will return to normal. To copy more than one program, move the cursor to the next desired title and press the space bar. There is no limit on how many titles you may select. You may even switch to other pages if necessary. All titles displayed with reverse characters will be copied. When you've made your selections, press f1 to start the copying procedure.

The copy routine begins with a prompt to make sure you are ready. If you change your mind, press f1 again and you will return to the

directory. Press any other key to begin copying. The program will begin with the first title in the directory and check whether you selected it. If it wasn't selected, it will skip to the next one until it finds one to copy. It then checks whether the file will fit into the available memory. If there is sufficient space, the file is loaded.

The program continues to check each of the selected files and loads them as long as there is room in memory. If a file is too big to be loaded, it is skipped until the next pass. After as many files as possible have been loaded into memory, you are prompted to insert the destination disk and press a key when you're ready. Again, if you change your mind, press f1 to abort the backup. If your sound is turned up, you'll hear a chime to let you know that the program is waiting for you to switch disks.

After all the files are saved, the program checks to see whether there were any files that were not loaded on the first pass. If no more files are to be copied, you will be asked if you want to copy the same files again to another disk or begin work on another disk or quit the program altogether.

A maximum of 49664 (about 48K) bytes of RAM is available for copying. This works out to 194 blocks. Files larger than this cannot be copied by this program. The amount of memory available varies slightly depending on how many titles are on the disk. All memory between 2047 ($07FF) and 53248 ($D000) is used for storage of loaded programs.

The disk drive status message is displayed each time after the disk drive has been accessed. This is handy information if there is a problem copying any of the files.

Fast File Copier is written so that fast load cartridges such as *Fast Load* from Epyx may be used while copying. This will speed up the loading of the files, but it will not speed up the saving of the files. (Sorry, "TurboDisk" will not work in conjunction with this program.)

Scratching Files

An important feature that is necessary after copying files is the ability to delete them from the source disk when they're no longer needed. With the Fast File Copier this is just as easy to do as copying files. Merely mark the files you want to delete and press f3.

There are two ways to proceed from this point. If you want all the marked files to be deleted automatically, press the A key. If you're not too sure of yourself, press the V key. This option will stop at each title and ask you if you are sure. If you want the file deleted, press the Y key. If you don't want it deleted, press the N key and it will skip to the next title. When all the selected files are deleted, you will be asked to press any key to continue. You may insert another disk at this time or keep the present disk in the drive. Press a key, and the directory of the disk will be loaded and displayed and you may continue again with any of the options shown on the directory screen.

Renaming Files

It is sometimes desirable to change the names of certain files after transferring them to another disk. This is done much the same way as copying or deleting files. First, mark all the titles you would like to rename and press f4. The old name of each file will be displayed, and you are asked to type in a new name. If you press RETURN without typing anything else, the file will not be renamed. The new name may be only 16 characters long. If more than 16 characters are typed in, only the first 16 will be used.

Formatting And Validating

Formatting and validating a disk is straightforward. You will be asked to type in a disk name and ID before formatting. Remember that formatting erases everything on the disk, so be careful with this function. Validating only requires you to type any key when you're sure you're ready. Both functions will allow you to abort by pressing f1.

When you're through using Fast File Copier, you may return to BASIC by pressing Q. Quitting this way is much the same as pressing RUN/STOP–RESTORE, except the screen is cleared and the colors are changed back to normal. ©

Omega Sort

Jonathan J. Holuta

Written entirely in machine language, this fast sorting routine for the Commodore 64 can be used by anyone and does not take away any space from BASIC memory.

If you write programs that handle data, sooner or later you will need a routine to sort items into alphabetical order. There are several sorting methods suitable for use in BASIC, including the bubble sort, shell sort, and quick sort. None of those methods, however, is very efficient for sorting large amounts of data.

"Omega Sort" is a speedy machine language routine which you can use in any BASIC program, even if you don't know a thing about machine language.

Omega Sort can sort 1000 randomly ordered strings in alphabetical order in less than six seconds. To see the routine at work, load and run the program "OMIGA.DEMO".

When you run OMIGA.DEMO, it loads the machine language routine (OMIGA SORT) from disk into memory. Then it prompts you to enter the number of strings you wish to sort. To create 1000 random strings, for instance, type 1000 and press RETURN. The program prints all of the strings on the screen in their original order, then it sorts them alphabetically. When the sorting is done, the program displays the strings in the new order, one screenful at a time. Press any key to view the next page of data, or press f1 to exit the program.

How To Use It

To use Omega Sort, your program must begin by loading the machine language routine into memory. The first line of OMIGA.DEMO demonstrates how this is done.

Some machine language sorting routines sort only one dimension of a multidimensional array, which is not always convenient. To demonstrate why, suppose that you have an address file program that stores a list of names and addresses in a two-dimensional array as shown here:

```
N$(1,1)=name 1
N$(1,2)=street 1
N$(1,3)=city 1
N$(1,4)=state 1
N$(1,5)=zip code 1
N$(1,6)=phone 1
```

Each full entry contains six separate items: the name, street, city, state, zip code, and phone number. In a real program, of course, you might have dozens or even hundreds of such entries. The name for entry 2 would be contained in N$(2,1), and so forth.

If you sort the first dimension of this array (name), then the names will be mismatched with the other data items. The name for entry number 1 might be matched with the street for entry 36, and so on.

Instead of sorting the strings themselves, Omega Sort sorts a numeric index array. Each element of the numeric array points to one data set in the string array. The advantage of this method is that all the items within each data set remain in their original order. In addition to great speed, this gives you more flexibility in using string arrays.

In OMIGA.DEMO, the string array is named A$, and the index array is named N%. Note that the index array must be an integer array (one whose name ends with %). Any legal Commodore variable names may be used, provided you follow this simple rule.

Calling The Machine Language

Like other machine language routines, Omega Sort is called with a SYS command. In addition to the command itself, which includes the starting address of the machine code, you must supply three items of information: the number of elements to sort, the name of the string array, and the name of the index array. Here is an example:

```
100 SYS 49152,N,N$(0),N%(0)
```

In this statement, the variable *N* indicates the number of elements to be sorted, and the variable *N$(0)* indicates the name of the array you wish to sort. If there are 40 elements in the N$ array, for instance, you would set N to 40 before executing line 100. Or, you could just replace N with the number 40. The variable *N%* is the index array.

Once the sorting is complete,

the index array contains the new order. To gain access to the sorted data, you must refer to elements of the string array through the index. Look at line 110 of OMIGA.DEMO. The expression A$(A%(X)) causes PRINT to display the elements of A$ in the order contained in the A% array. Remember, Omega Sort rearranges the order of the numeric index array, not the string array itself. Each element of the index array points to one element of the string array.

The SYS statement for a multi-dimensional array is the same, except that you must specify which dimension to sort. Here is an example:

```
100 SYS 49152,N,N$(0,3),N%(0)
```

For the address array mentioned above, the preceding statement would sort the addresses according to the array's third dimension (city). This statement would sort it according to the first dimension (name):

```
100 SYS 49152,N,N$(0,1),N%(0)
```

This statement would sort the address array by its fifth element (zip code):

```
100 SYS 49152,N,N$(0,5),N%(0)
```

Here is an example line that would print the elements of the address array in their new order:

```
110 FOR X=0 TO N:PRINT
X,N$(N%(X),5):NEXT
```

You can use this routine without knowing how it works, but, for those who are interested, here is a brief explanation. Omega Sort first stores important zero page pointers in the cassette buffer so it can use these locations for its own purposes. Then it determines where in memory the arrays reside. In the case of strings, the actual text is stored from the top of BASIC memory in a downward direction. The array storage space (located just above the end of BASIC program text) contains a series of pointers to the strings in high memory. Omega Sort checks the pointers and then changes the values of the integer array to match the alphabetical order of the strings themselves. When finished, it restores the contents of the zero page and returns to BASIC. The entire process works so quickly that it can sort a hundred strings in less than a second. ©

Automatic Syntax Checker

Philip I. Nelson

How many times have you typed PRNIT instead of PRINT, or LIT instead of LIST? When it comes to typing, none of us is perfect. This automatic utility for the Commodore 64 can save you a lot of time by catching such mistakes before they're added to your programs.

Unlike some other computers, the Commodore 64 doesn't check BASIC lines for errors as you type them in. We all know a line like 010 MONKEY(BIZ)*5-DOGA$# is nonsense, but in many ways the computer treats it as normal BASIC. You can type in that line, list it, renumber it, even save it as a program and load it back into memory without any protest from your 64.

When a line starts with a number, the computer simply stores it in BASIC memory—no questions asked. The 64, like all other Commodore computers, can't find mistakes until it's *running* a program. As a result, after spending hours writing a program, you may spend hours more watching it crash until you've corrected all the typing errors.

"Automatic Syntax Checker" eliminates that headache by adding automatic error checking to your Commodore 64's BASIC. Once the Syntax Checker is installed, the computer automatically checks every BASIC line you type in. If the line is free of typing errors, it's added to your program as usual. If not, the Syntax Checker prints an error message and lets you try again. Since the line isn't correct, the Syntax Checker doesn't add it to your program.

Getting Started

Although Automatic Syntax Checker is written entirely in machine language, you load and run it as you would any BASIC program. *Do not try to start this program with SYS.* Because it handles like BASIC, it's easy to make new copies of the Syntax Checker as well. Simply save it on a new disk or tape as you would a BASIC program.

When you run the Syntax Checker, it moves itself from BASIC program space to a safe memory location at 49152. After a brief preparation, it performs a NEW to let you type in your own programs. When you see the message CHECKER ON, you know the Checker is ready to go. Since this utility does NEW after installing itself, you should always run it *before* you load or type in any other program.

Although you can run BASIC programs while the Syntax Checker is active, some programs may POKE into the same memory space at 49152 or disrupt it in other ways. Thus, it's wise to turn the Syntax Checker off before you run any BASIC program. Type SYS 49152 and press RETURN: The message

CHECKER OFF tells you the system is back to normal. To turn the Checker back on, enter SYS 49152 again (it won't do a NEW this time). Once the Syntax Checker is installed, SYS 49152 turns it on or off safely, whenever you want, without disturbing the BASIC program in memory.

Automatic Error Checking

When the Syntax Checker is active, it analyzes every line you type into the computer. If you type a command in direct mode (without a line number), the Syntax Checker simply passes it along to the computer (Commodore BASIC can find direct mode errors by itself). Thus, you can use direct commands as usual to load and save programs, and so on.

However, when you put a number at the beginning of the line, the Syntax Checker scans everything in the line to make sure it's correct BASIC. If no error message appears, then you know the line is correct—that is, it's all BASIC that the computer can understand. To demonstrate, run the Syntax Checker, then type in the following line:

```
10 PRINT "HI"
```

Since this line has no errors, the 64 behaves normally. After you press RETURN, the blinking cursor goes to the next line on the screen. You can list this program to confirm that it's there, or run it to see if it works. Now let's force an error. Move the cursor up and type over line 10 so it looks like this:

```
10 PNIRT "HI"
```

Because PNIRT isn't a BASIC word, the Syntax Checker prints SYNTAX ERROR. List the program again to see whether the computer replaced the good line with a bad one. As you'll see, the first (correct) version of line 10 is still there. The Syntax Checker won't let you add *or change* any lines in a program unless they're free of typing errors. This feature makes it handy for editing existing programs as well as writing new ones.

Phantom BASIC

The Syntax Checker does far more than check BASIC keywords like PRINT for correct spelling. It passes judgment on everything else in the BASIC line as well. Using a modified version of BASIC, it performs a *phantom execution* of each new line, performing most of the same error checks BASIC would use if you ran the line in a program.

To do this, the Syntax Checker switches out the computer's BASIC and switches in a special, modified BASIC which it prepared when you started it up. The phantom BASIC "runs" the line to perform normal error checks, but stops short of actually completing the commands. This method can flush out a multitude of errors, including missing colons in multistatement lines, misplaced commas and parentheses, type mismatches (putting a string where a number belongs, and vice versa), missing parameters, illegal operations (like A$=B$−C$) and many illegal quantity errors as well.

Runtime Errors

But there are certain things the Syntax Checker cannot do. A program can contain innumerable flaws even when each of its lines is *syntactically* correct. To illustrate, type NEW and press RETURN, then type in the following line and enter RUN.

```
1000 NEXT
```

The NEXT statement in line 1000 is perfectly good BASIC—the 64 understands exactly what it means—but NEXT only makes sense when it's preceded with a matching FOR statement. This is a *runtime* error, so called because it can't possibly be detected without running the program. When you type in this line by itself, the Syntax Checker can tell that NEXT is a word from the BASIC language, but there's no way it can tell whether it's logical to put NEXT at that particular point in the program. There are many runtime errors (DEVICE NOT PRESENT, OUT OF DATA, etc.) which can't be found until a program is finished and running.

Thus, while the Syntax Checker looks after the fundamentals, catching obvious mistakes, it's up to you, the programmer, to make sure your creation makes sense as a whole. It can't read your mind to figure out what you really meant the program to do. Though it eases the burden of programming, the Syntax Checker can't perform magic: There's no way it can turn a badly structured, illogical program into something that works, and certain errors are simply beyond its scope.

Quirks And Compatibility

You should also keep in mind that ordinary BASIC lets you do some very odd things without signalling an error. LIST 10–30ABCDEF looks very strange, but doesn't cause an error. DIM A(5.001,26.3) is silly (you can't dimension a fractional number of array elements), but there's no "illegal fraction" error message to tell you about it.

In short, BASIC has some built-in limitations when it comes to error checking. Fortunately, most of its quirks are rarely encountered, and the ones shown above don't cause any real harm. But because the Syntax Checker uses existing BASIC routines, it's subject to the same quirks and anomalies. It doesn't sense errors that BASIC itself can't detect.

You may already have wondered whether the Syntax Checker can be used along with other programs like "MetaBASIC." The Syntax Checker works with MetaBASIC. However, MetaBASIC's QUIT command turns off the Syntax Checker, too. So unless you want to turn off both programs at once, you must restart the Checker with SYS 49152 whenever you QUIT MetaBASIC. As you may know already, the more utilities concurrently active, the more fragile the system is likely to become. No matter what other programs you're using at the time, it's always best to disable the Syntax Checker with SYS 49152 before you run a BASIC program.

Space doesn't permit a detailed explanation of how this program works, but here's a brief synopsis. When the Syntax Checker sets up, it copies BASIC and the computer's operating system (OS) from their normal places in ROM (Read-Only Memory) into underlying free memory, then modifies them extensively. When you enter a numbered line, the program turns off the computer's ROM and uses the modified BASIC and OS. After it scans the input line, the Syntax Checker either adds it to the program or signals an error. In both cases, ROM is turned back on before the blinking cursor reappears. ©

Programming Books

from COMPUTE!

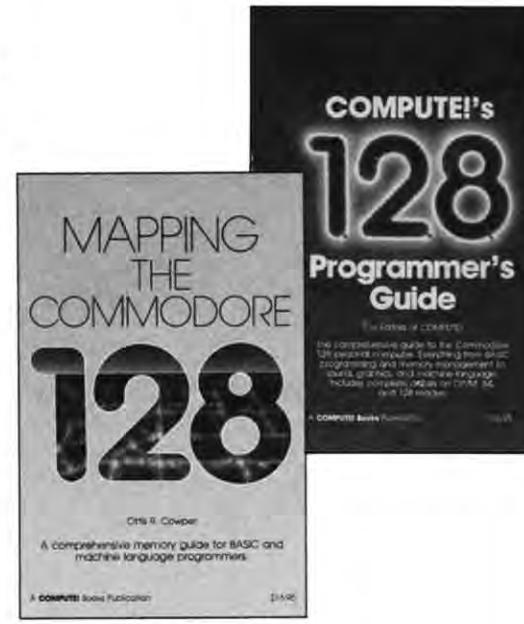
COMPUTE! Books offers a line of programming books for the intermediate to advanced Commodore 64 and 128 users. These reference books take you beyond BASIC and into machine language programming, helping you learn about memory maps, addresses, the new *GEOS*, and time-saving routines you can add to your own programs.

COMPUTE!'s 128 Programmer's Guide

Editors of COMPUTE!
ISBN 0-87455-031-9 444 pages
A complete guide to the Commodore 128, this book explores BASIC 7.0, shows you how to create graphics and sounds, explains how to program peripherals, and introduces you to machine language programming.
$17.95

Mapping the Commodore 128

Ottis R. Cowper
ISBN 0-87455-060-2 704 pages
The comprehensive memory map and programmer's guide that provides a detailed explanation of the inner workings of the Commodore 128 including memory management, BASIC 7.0, I/O chip register, the operating system, system RAM, and more.
$19.95



Machine Language Routines for the Commodore 128 and 64

Todd Heimarck and Patrick Parrish
ISBN 0-87455-085-8 592 pages
This collection of machine language routines is a must for every Commodore 128 and 64 machine language programmer. Scores of these routines can simply be inserted into your own programs. Included is the assembly language code with easy-to-understand documentation and instructions. *There is a companion disk available for $12.95 that includes all the programs in the book* (858BDSK).
$18.95

Programming the Commodore 64 Revised: The Definitive Guide

Raeto Collin West
ISBN 0-87455-081-5 642 pages
This bestselling, encyclopedic reference guide which covers the Commodore 64 in its entirety has been updated to include information on the new Commodore 64C and *GEOS*, from Berkeley Softworks. *There is also a disk available for $12.95 which includes the programs in the book* (507BDSK).
$24.95

Mapping the Commodore 64 and 64C

Sheldon Leemon
ISBN 0-87455-082-3 324 pages
An update of the bestselling memory map and programming guide that's a necessity for intermediate and advanced programmers. This definitive sourcebook has been expanded and now covers the new icon-based *GEOS* (Graphics Environment Operating System) with clear descriptions of how to make it work for you. For BASIC and machine language programmers of both the Commodore 64 and 64C.
$16.95

These books are now available at your local book or computer store.
You can also order directly from COMPUTE! by calling toll free 800-346-6767 (in NY call 212-887-8525) or by mailing your order to COMPUTE! Books, P.O. Box 5038, F.D.R. Station, New York, NY 10150.

Customer Service hours are 10:00 am–12:30 pm and 1:30 pm–3:00 pm EST, Monday–Friday
Please include $2.00 postage and handling per book or disk. NC residents add 5 percent sales tax, and NY residents add 8.25 percent sales tax. Please allow 4–6 weeks for delivery.

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

COMPUTE! books are available outside the United States from subsidiaries of McGraw-Hill International Book Company.

X BASIC

Kevin Martin

This powerful BASIC extension for the Commodore 64 focuses on two of the most complex areas to program: graphics and sound. With 33 new commands, it offers control of high-resolution screens, character and screen display functions, sprites, and sound.

Two of the most complex and tedious areas of programming on the 64 are graphics and sound. Even some of the most elementary graphics and sound routines require a lot of programming time and long strings of POKEs. "X BASIC" (eXtended BASIC) is a programmer's language that does much of the difficult programming for you, allowing you more time to design and experiment.

Written entirely in machine language, X BASIC adds 33 graphics and sound commands to BASIC: 7 commands for hi-res; 7 for screen and character functions; 6 for sprite creation and manipulation; 10 for sound; 2 for joystick control; and 1 for exiting the program.

To load X BASIC, type **LOAD "X BASIC",8,1**. To activate it, type **NEW** and press ENTER, then type **SYS 49152.**

All commands can be used in direct mode as well as program mode. Some commands need to be followed by one or more parameters consisting of numbers or strings. The parameters can be variables (X or N$) or actual values (5 or "HELLO"). Numeric parameters must be within a specified range. Attempting to use a number that's outside the range of legal values results in an ILLEGAL QUANTITY error. Also, if you try to use a number for a parameter which requires a string, or vice versa, you'll receive a TYPE MISMATCH ERROR. *(Note: All X BASIC commands are in boldface capital letters. Command parameters follow in boldface lowercase letters.)*

Hi-Res Graphics Commands

HIRES: Turns on the hi-res screen and sets it to multicolor mode. (All hi-res graphics in X BASIC are set up for multicolor mode.) The hi-res screen is located underneath the Kernal ROM starting at $E000. You can POKE directly to the screen, but you can't PEEK the screen unless you switch out the Kernal.

TEXT: Returns to normal text mode. *Always use TEXT to go from hi-res mode to normal display. Do not exit hi-res with RUN/STOP-RESTORE.*

CLEAR *color0,color1,color2, color3*: Clears the hi-res screen and lets you choose the colors for plotting. *Color0* corresponds to the 01 bit pair, *color1* the 10 bit pair, and *color2* the 11 bit pair. *Color3* is the background color. The values for the colors correspond to the POKE values in the range 0–15.

COLR *color#*: Selects the color for the PLOT and LINE commands on the hi-res screen. Its parameter allows you to select one of the four colors (0–3) defined by the CLEAR command.

PLOT *x,y*: Plots a point on the hi-res screen in the current color. If the color is 3 (the background color), it effectively erases a point. The x-coordinate is the distance from the left edge of the screen and ranges between 0–159. The y-coordinate is the distance from the top of the screen between 0–199.

LINE *x1,y1* TO *x2,y2*: Draws a line between *x1,y1* and *x2,y2* in the current color (or erases like PLOT). The x-coordinates range between 0–159, the y-coordinates between 0–199.

HPRNT *string*: Prints a string on the hi-res screen at the location set by LOCATE (see below). When the hi-res screen is in multicolor mode, the characters will appear distorted. Multicolor mode can be turned off with the MULTI 0 command. HPRNT defaults to the uppercase/graphics character set. If you'd prefer the upper/lowercase set, enter this line: **POKE 50819,216**. To return to uppercase/graphics, **POKE 50819,208**. You can mix characters from the two character sets on the same screen.

Screen And Character Commands

BRDR *color#*: Sets border color. *Color#* must be a number from 0–15.

SCREEN *color#*: Sets screen color. *Color#* must be a number from 0–15.

CENTER *string*: Centers a string of characters on the current line of text, then prints a RETURN. *The string must be 1–38 characters long.*

LOCATE *x,y*: Positions the current printing location on the text or hi-res screen to x,y. *X* must be between 0–39, and *Y* between 0–24.

MULTI 0 or **MULTI 1,*color1, color2*:** MULTI 0 turns off multicolor mode. MULTI 1 (with the two color parameters, between 0–15) turns on multicolor mode. *Color1* corresponds to the 01 bit pair, and *color2* the 10 bit pair.

EXTND 0 or **EXTND 1,*color1, color2,color3*:** EXTND 0 turns off extended background mode. EXTND 1 (with background color parameters, between 0–15) turns on extended background mode. Characters with a POKE value of 64–127 use *color1* for the back-

ground, characters 128–191 use *color2*, and characters 192–255 use *color3*. *Note: extended background color mode cannot be used when the hi-res screen is turned on.*

CSET *block*: Copies the uppercase character set from ROM to RAM at *block**1024. *Block* should be a number between 1–15. This command is handy when using custom characters. You may have to prevent BASIC and the character set from interfering with each other by moving the bottom (or top) of memory.

Here's a short program that demonstrates the use of some of the hi-res and screen and character commands above. After X BASIC is activated, type in this program and run it. You might try changing some of the X BASIC command parameters for various effects.

```
100 BRDR0
110 HIRES
120 CLEAR 1,2,3,4
130 LOCATE 10,10
140 HPRINT "HELLO"
150 FOR I=0TO3
160 COLRI
170 LINE 0,0TO159,159
180 LINE 0,199 TO 159,0
190 NEXT
200 GETA$:IFA$=""THEN150
210 TEXT
220 END
```

Sprite Commands

All of the sprite commands are for use in text mode only. You cannot use these commands for putting sprites on a hi-res screen.

SPRITE *sprite#,block,color*: Defines a sprite. *Sprite#* is the sprite number (0–7), *block* is the block number of the sprite definition (0–255), and *color* is the sprite color (0–15). The sprite shapes can be put in any free area of memory from 0–16383. You may have to move the bottom or top of memory to avoid memory conflicts.

SPRAT *sprite#,xexp,yexp,priority,multi*: Sets the sprite attributes (characteristics). To turn an attribute on, use 1; to turn it off, use 0. *Xexp* expands in the x-direction, *yexp* in the y-direction. The *priority* parameter determines whether the sprite has priority over screen characters. For example, if you want a sprite to pass over a screen character, the sprite has priority, thus a parameter of 1. If the sprite is to pass under a screen character, the parameter should be 0. *The multi* parameter makes the sprite multicolored. The colors are specified with SPRMULT.

SPRMULT *color1,color2*: *SPRMULT* sets the two additional sprite colors used in multicolor mode. The parameters must be within 0–15.

ASPRITE *sprite#*: Activates a sprite. *Sprite#* must be 0–7.

DSPRITE *sprite#*: Deactivates a sprite. *Sprite#* must be 0–7.

MOVE *sprite#,x,y*: Moves a sprite to the coordinates (x,y), based on the upper lefthand corner of the sprite. *Sprite#* must be 0–7. *X* must be in the range 0–511, *y* in the range 0–255. Note that the coordinates do not correspond to the screen coordinates; some locations may cause sprites to be partially or completely off the visible screen area.

This short program is a basic example of how to create and animate a sprite:

```
100 BRDR 0:SCREEN 11
110 SPRITE 1,123,7
120 SPRMULT 1,3
130 ASPRITE 1
140 FOR J=0 TO 1:SPRAT 1,J,J,J,J
150 FOR I=0TO350
160 MOVE 1,I,100
170 NEXTI,J
```

Sound Commands

SID: Clears the SID (sound) chip.

VOL *volume#*: Sets the volume register. *Volume#* must be 0–15.

ENVELOPE *voice#,ad,sr(,pulse width)*: Sets the attack, decay, sustain, release, and optionally (parentheses indicate an optional parameter) the pulse width. *Voice#* must be 1–3. *Ad* (attack and decay) and *sr* (sustain and release) each must be in the range 0–255. Pulse width must be in the range 0–4095.

WAVE *voice#,waveform(sync)(ring)*: Sets the waveform. You must specify the voice number followed by T, S, P, or N for triangle, sawtooth, pulse, and noise, respectively. Optionally, you can add *s* for synchronization, and *r* for ring modulation. Here are some examples: WAVE 1,TSR (triangle waveform with synchronization and ring modulation); WAVE 3,PR (pulse waveform with ring modulation); WAVE 2,SS (sawtooth waveform with synchronization).

Quick Reference Chart For X BASIC Commands

Hi-Res Graphics Commands

HIRES
TEXT
CLEAR *color0,color1,color2,color3*
COLR *color#*
PLOT *x,y*
LINE *x1,y1* **TO** *x2,y2*
HPRNT *string*

Screen And Character Commands

BRDR *color#*
SCREEN *color#*
CENTER *string*
LOCATE *x,y*
MULTI 0 or **MULTI 1,***color1,color2*
EXTND 0 or **EXTND 1,***color1,color2,color3*
CSET *block*

Sprite Commands

SPRITE *sprite#,block,color*
SPRAT *sprite#,xexp,yexp,priority,multi*
SPRMULT *color1,color2*
ASPRITE *sprite#*
DSPRITE *sprite#*
MOVE *sprite#,x,y*

Sound Commands

SID
VOL *volume#*
ENVELOPE *voice#,ad,sr(,pulse width)*
WAVE *voice#,waveform(sync)(ring)*
FRQ *voice#,frequency*
GATE *voice#,on/off*
FCUT *cutoff*
FRSN *resonance*
FMODE *type*
FILTER *voice#,on/off*

Joystick Commands

STICK *joystick#*
BTN *joystick#*
(joystick#'s **must be in parentheses)**

Reset Command

QUIT

FRQ *voice#,frequency*: Sets the frequency in the range 0–65535. *Voice#* must be 1–3.

GATE *voice#,on/off*: Gates a voice on or off. *Voice#* must be 1–3 followed by 0 to turn the gate bit off or 1 to turn it on.

The following sound commands deal only with filters. Some programmers may not be interested in these, but if you have a serious interest in 64 sound, you'll find these handy.

FCUT *cutoff*: Sets the cutoff frequency. *Cutoff* must be in the range 0–2047.

FRSN *resonance*: Sets the filter resonance. *Resonance* must be in the range 0–15.

FMODE *type*: Selects the filter

type. *Type* can be H for high band filters, L for low band, B for bandpass, or a combination of these. For example, FMODE HBL for all types; FMODE HL for high and low.

FILTER *voice#,on/off*: Turns the filter on or off. *Voice#* must be in the range 1–3 followed by 0 to turn off the filter or 1 to turn it on.

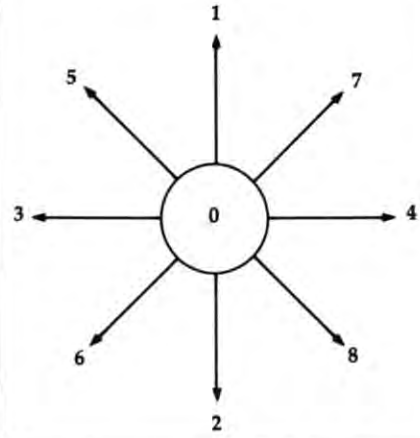
To demonstrate the sound commands, type in this program. By playing with the parameters, you can produce some most unusual effects.

```
100 SID
110 VOL 15
120 FOR I=1TO3
130 ENVELOPE I,15,240,2000
140 FILTER I,1
150 NEXT
160 FRSN 15
170 FMODE H
180 WAVE 1,TSR
190 WAVE 2,SS
200 WAVE 3,P
210 GATE 1,1
220 GATE 2,1
230 GATE 3,1
240 FRQ 1,4000
250 FRQ 2,6000
260 FORI=100TO2000
270 FCUT I:FRQ 3,20000-I*10
280 NEXT
290 SID
```

Joystick Commands

The parameters for these commands require parentheses ( ). Note the examples in the descriptions below.

STICK *joystick#*: Reads the joystick direction. *Joystick#* must be 1 to read port 1, or 2 for port 2. This command returns a value in the range 0–8, so it must be used like a function (X=STICK(1), for example). Values 1–8 correspond to the eight possible directions as shown in the figure. The value is 0 if the joystick is centered.



BTN *joystick#*: Returns a value of 1 if the firebutton is pressed, or 0 if it's not. *Joystick#* must be 1 or 2. As with STICK, this command is a function. A good way to use it is within an IF-THEN statement (IF BTN(1)=1 THEN ...).

Reset Command

QUIT: Disengages X BASIC by resetting all vectors to normal.

Space doesn't permit a lengthy discussion of *how* to program graphics and sound on the 64. If you've had experience in these areas, you'll find this utility an efficient tool. If you're a beginning programmer, you may wish to consult the *Programmer's Reference Guide* for specific information on programming graphics and sound. ©

128 Editing Functions For Commodore 64

Jim Allen

This powerful programming aid makes all of the important Commodore 128 screen-editing functions available on the Commodore 64.

If you own a Commodore 64, you may wish that you had the extra screen-editing functions available on the Commodore 128. With just two quick keystrokes, you can turn auto-insert mode on or off, clear selected portions of the screen, set and manipulate screen windows, move the cursor instantly to any location, and more. In the 128, these functions are called ESC (*escape*) functions because they are activated with the ESC key. "128 Editoring" adds 14 ESC functions to the 64, plus a NO SCROLL key to prevent screen scrolling, and three new functions that aren't even available on the 128. It also allows all keys to repeat.

This program loads with the command LOAD "128 EDITING",8,1. After the program has loaded into memory, type these commands in direct mode (without line numbers):

```
NEW
SYS 49152
```

Don't forget to press RETURN at the end of each line. After you've entered the second command, the program sets the screen border to the same color as the background as a signal that it is active.

New ESCape Functions

Since the 64 lacks an ESC key, this program uses the back-arrow key (←) in its place. In the remainder of this article, the term ESC refers to the back-arrow key at the upper left corner of the keyboard. ESC functions require two keypresses: First you press and release the ESC key; then you press a second key. For instance, to delete a line, you press ESC and then D.

If you change your mind after pressing ESC and decide *not* to perform an ESC function, simply press a key that has no special ESC function (a CRSR key, for instance). If you need to type the back-arrow character itself, press ESC twice in succession. A description of the ESC functions follows.

Erase/Delete Functions

ESC @ Clears the screen from the current cursor position to the bottom.

ESC D Deletes the line the cursor is on, scrolls the screen up to fill that line, and places the cursor on the left screen margin.

ESC P Erases from the start of the current line up to and including the current cursor position.

ESC Q Erases from the current cursor position to the end of the current line.

f1 Erases from the line the cursor is on, including the current line, to the top of the screen or window.

f3 Deletes the character under the cursor and moves the remainder of the line one space to the left.

Insert Functions

ESC A Turns on autoinsert mode, which allows you to insert printable characters without using the INST key. The cursor and other editing keys work as usual. RETURN cancels autoinsert mode (this is not true on the 128).

ESC C Cancels autoinsert mode. This works the same as ESC C on the 128. (Note, however, that some of Commodore's documentation for the 128 confuses ESC C with the ESC O function.)

ESC O Cancels insert and quote modes, allowing you to use cursor keys and other editing keys after pressing INST or typing a quotation mark.

ESC I Inserts a blank line at the current cursor position, scrolling the remainder of the screen down and placing the cursor on the left edge of the display.

Cursor Movement Functions

ESC J Moves the cursor to the beginning of the line.

ESC K Moves the cursor to the end of the text on the line.

f7 Moves the cursor to the lower left corner of the screen.

Miscellaneous Functions

ESC T Sets the top of the window. Blocks the top portion of the screen from being erased or scrolled. The cursor position determines the top row of the new window. To reset the window

to the full screen size, press the HOME key twice.

ESC V Scrolls the contents of the screen or window up one line. A new blank line will be scrolled in at the bottom, and the previous contents of the top line will be lost.

ESC W Scrolls the contents of the screen or window down one line. A new blank line will be scrolled in at the top, and the previous contents of the bottom line will be lost.

SHIFT Enables the NO SCROLL feature. The NO SCROLL key on the 128 lets you pause printing to the screen display (for instance, when you are listing a program). To pause a scrolling display, press SHIFT or SHIFT LOCK. When you release the key, printing resumes.

ESC Z Disables all 128 Screen Editor functions. You can reenable the ESC functions at any time with SYS 49152. (On the 128, ESC Z clears all TAB stops, a function not available in this program.)

You should be aware that these functions affect logical lines, not physical screen lines. On the Commodore 64, a physical line is always 40 characters long, but a logical line can overlap two physical screen lines. Thus, for example, the ESC V function may scroll the screen upward two lines if the topmost logical line in the current window is more than 40 characters long.

The window function (ESC T) does not prevent you from moving the cursor above the window boundary with the cursor keys. If you venture above the boundary, strange results will occur. (If this happens accidentally, perform a warm start by pressing RUN/STOP–RESTORE; then restart the program with SYS 49152.)

The delete-line function (ESC D) is intended for deleting a line which lies between two other lines. If you simply want to erase a line (particularly a line on the bottom screen line), use the ESC Q function.

This program works by copying BASIC and the Kernal ROM into underlying RAM and modifying them. It also modifies two important vectors: IQPLOP at $306 and IBASIN at $324. Locations $02, $B6, and $334–$338 are also used for various purposes.

Quick Reference

Function	Keys
Erase from cursor to end of screen	ESC @
Delete current line	ESC D
Erase from start of line to cursor	ESC P
Erase from cursor to end of line	ESC Q
Erase from cursor to top of window	† f1
Delete to right of cursor	† f3
Enable auto-insert mode	ESC A
Cancel auto-insert mode	ESC C
Cancel insert and quote modes	ESC O
Insert a line	ESC I
Move cursor to start of line	ESC J
Move cursor to end of line	ESC K
Move cursor to lower left corner	† f7
Set top of window	ESC T
Scroll screen/window up	ESC V
Scroll screen/window down	ESC W
Pause scrolling	† SHIFT
Disable Editor	† ESC Z
Enable Editor	† SYS 49152

† different from Commodore 128

Archive:
Two-Drive Backup For Commodore 64

Philip I. Nelson

Now you can copy entire disks at machine language speed with this convenient backup program for the Commodore 64 with two 1541 disk drives. It also works on the new Commodore 128 in 64 mode.

Sooner or later it's bound to happen. You'll make an unconscious error, or lightning may strike while you're resaving a program, or the family dog will chew a few disks for dinner—and an important disk will be utterly destroyed. If you have a backup copy, of course, such accidents aren't fatal. You take a moment to pat yourself on the back, pull out the archive disk, and go back to work.

If you don't have a backup, it's like watching a gold ring slip off your finger and go clanking down the drain. In the long hours spent reconstructing what you've lost, you have plenty of time to reflect on the wisdom of archiving your work on a regular basis.

Archiving is one of those grim tasks that's easy to postpone. BASIC programs (like COPY/ALL on your 1541 Test/Demo disk) are slow, and may not copy machine language (ML) programs or sequential files. Even good single-drive backup programs keep you tied to the computer, tapping your fingers until it's time for the next disk swap.

Fast And Convenient

"Archive" offers a better way: It links two drives together to take the misery out of backing up important disks. To speed things up, it's written entirely in machine language and copies only those disk sectors which actually contain data. But because it loads and runs just like a BASIC program, it's easy for anyone to use, even beginners.

You may find this program valuable even if you don't own two disk drives. Put your drive together with a friend's and swap several disks during one session. Or take it to a user group meeting to speed up the duplication of public domain library disks. Since 1541-format disks work with other Commodore computers, Archive running on a Commodore 64 can also copy disks that will be used with the Commodore 128, VIC-20, Plus/4, 16, and 4040-format PET/CBM. (Of course, a program written for one of these machines may not work on another. Also, Archive cannot copy Commodore 128 CP/M disks.)

Incidentally, Archive cannot duplicate commercially protected software. Protected disks invariably contain deliberate errors (which shut down the program) or data hidden in unused sectors (which Archive does not copy).

Getting Started

To use Archive, activate it like a BASIC program by typing LOAD "ARCHIVE",8 followed by RUN. (For this program, *do not* use ,8,1 after the LOAD; just use ,8.) If you're already comfortable using two drives, you needn't read any further, since Archive prompts you at each step. Just pop a disk in each drive as instructed, press the f7 special function key, and relax while Archive does its work. (If you've never used two drives before, see "Setting Up Your System" below.)

Archive displays your source disk's Block Availability Map (BAM) graphically on the screen, updating the display as copying proceeds. Thus, you can tell at a glance how much of the disk is used and how much has been copied. The number at the lower right of the screen shows the sector being copied; the graphic display shows which sectors have already been copied.

If you want to abort the copy for any reason, press the f1 special function key to return to BASIC. (When you abort the copy process, the archive disk is incomplete and may be garbled. You can reuse it immediately with Archive, but do not use it for anything else without reformatting it as explained below.) Once the copy is done, press the f3 function key to copy another disk, or press f1 to quit. Whenever you exit Archive, it clears the screen and reports the status of each drive.

Quick Formatting

Since Archive always makes a complete disk copy, it *formats* the archive disk with a NEW command. Formatting renames the disk and erases everything it contained before. You'll notice that the archive disk is formatted in only a few seconds rather than the usual couple of minutes, and without the usual knocking sound. To save time and minimize wear on the drive, Archive

We Won't Be UNDERSOLD

1st In Price, 1st In Support, 1st In Warranty

15 Day Free Trial • 90 Day Immediate Replacement Policy • Free Catalogs

Call before you order, our prices may be lower • All sale prices expire 11-30-87

80 Column Printer• 8½" Letter Size

Big Blue Printer

Dot Matrix
Heat Transfer
Upper Case
Lower Case
Underline
Enlarged
Affordable
Plus More

No One Sells This Printer For Less!

Sale $39.95

List $199

(Add $7.50 Shipping*)

This printer was made by **Canon**® for IBM. The Big Blue printer comes ready to hook up to the serial port of the **IBM**® PC jr, XT, AT & Compatibles. Plus with our low cost adapter cables you can connect this printer to the **Apple**® II, IIe, IIc and Compatibles, **Atari**® computers, **Commodore**® 64, 128, SX-64, Vic 20, Plus 4, and more.

RS-232 Adapter - Adapter for IBM® PC, AT, XT & Apple® II series RS-232 port. (Specify male or female). . . List $49.95 **Sale $19.95**
Apple® IIc & Laser 128 Interface - With printer driver program for graphics and text. . . . . List $49.95 **Sale $19.95**
Intelligent Commodore® Interface - Prints graphics and text. Use Print Shop and more. . . . . List $49.95 **Sale $19.95**
Intelligent Atari® Interface - (Excludes Atari 1200) Prints graphics and text. Use Print Shop and more. . . . . List $49.95 **Sale $19.95**
Paper (2 Rolls) . . . . . List $19.95 **Sale $ 5.95** **Single Sheet Paper** (Qty. 500) . . . . . List $29.95 **Sale $12.95**

IBM, Apple, Canon, Commodore, Atari & Laser are registered trademarks of International Business Machines, Apple Computer, Canon Inc, Commodore Business Machines, Atari Inc. & Video Technologies Respectively.

5¼" Floppy Disk Sale

Double-Sided • Double-Density • 100% Certified

24¢ each*

*1 Box of 100 — $24.00 (24¢ each)

Paper Economy Sleeves (100) — $5.00

Hi-Speed (160-180 CPS) Printer

NLQ-180 Printer

SALE

SALE

Near Letter Quality

Lifetime Warranty *

No One Sells This Printer For Less!

Sale $179.95

List $499

(Add $10.00 Shipping*)

• Near Letter Quality Selectable From Front Panel Controls • High Speed Dot Matrix • Letter Quality Modes • 8K Buffer frees up computer 4-times faster • Super Graphics • Pica, Elite, Italics, Condensed • Business or Personal • Tractor/Friction • 15 Day Free Trial • Lifetime Warranty on Print Head* • 6 Month Immediate Replacement Policy •

COMPUTER DIRECT (A Division of **PROTECTO**)
22292 N. Pepper Rd., Barrington, IL. 60010
312/382-5050 or 312/382-5244
We Love Our Customers

* Illinois residents add 6½% sales tax. All orders must be in U.S. Dollars. We ship to all points in the U.S., CANADA, PUERTO RICO & APO-FPO. Please call for charges outside continental U.S. or C.O.D. **MAIL ORDERS** enclose cashier check, money order or personal check. Allow 14 days delivery, 2 to 7 for phone orders and 1 day express mail. Prices and availability subject to change without notice. (Monitors only shipped in continental U.S.)

VISA — MASTERCARD — C.O.D.

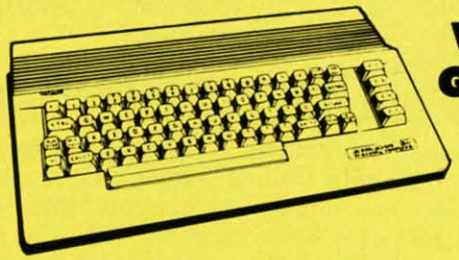
We Won't Be UNDERSOLD

15 Day Free Trial • 90 Day Immediate Replacement Policy • Free Catalogs

Call before you order, our prices may be lower • Sale prices expire 11-30-87

C64c Computer

Included with each computer is the GEOS Program: *word processor and a versatile drawing program.*



With GEOS!

Sale $149.95

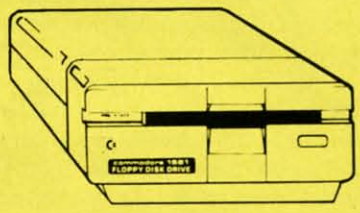
(Add $10.00 shipping.*)

List $249

1581 3½" Disk Drive

Over 800K bytes of formatted storage on double-sided 3.5" Microdiskettes.

NEW



Sale $179.95

(Add $10.00 shipping.*)

List $249

C128D Computer

Includes a full 128 detached keyboard and 1571 drive, with 128K of user memory expandable to 640K.

NEW

Sale $479.95

(Add $10.00 shipping.*)

List $599

Complete C64c System

INCLUDES

- **Computer**
- **Disk Drive**
- **Monitor**
- **Printer**
- **Software**

$395.95 System Includes:

- **Commodore 64c Computer**
- **Excellerator Plus Disk Drive**
- **Hi-Res 12" Monochrome Monitor** with connection cable
- **Big Blue 8½" Printer** with interface and 2 rolls of paper
- **GEOS Program:** Word processor and drawing prorgram

Sale $395.95

(Add $35.00 shipping.*)

List $1049

Call

(312) 382-5050
For IBM & Apple

(312) 382-5244
For Atari & Commodore

Mail

COMPUTER DIRECT
22292 N. Pepper Road
Barrington, IL. 60010
We Love Our Customers

uses a shortened NEW command: the equivalent of OPEN 15,8,15, "N0:*filename*" without a disk ID.

The abbreviated NEW command works only on a disk that has been previously formatted. To use a brand new disk, you must prepare it first with a full NEW command: OPEN 15,8,15,"N0:*filename,ID*".The filename can be up to 16 characters long. The ID can be any two letters or numbers and should be unique for each disk. The *1541 User's Manual* contains more information about formatting disks.

Setting Up Your System

Although the Commodore 64 has only one serial port connector, the 1541 disk drive has two, letting you hook up more than one drive at a time. Since the drives are chained together in a series, this arrangement is often called *daisy-chaining*. Connect the first drive as usual, then plug the serial cable from the second drive into the extra serial port connector on the first drive.

When more than one disk drive is active, each drive must be given a different *device number* so the computer can tell them apart. The 1541 is factory-set as device 8, but it can also have device numbers 9–15. Archive uses device numbers 8 and 9, reading from drive 8 and copying to drive 9. *You must always put the source disk (the original) in the drive that's device 8 and the archive disk (the copy) in the drive that's device 9.*

If both of your drives are device 8, don't despair. You can easily change one of them to device 9. The change is temporary; the drive reverts to device 8 when you turn off the power. Here's the procedure:

1. Turn on the drive that you want to change to device 9. *Make sure the other drive is turned off.* Now you can change the device number either by running the DISK ADDR CHANGE utility program on your 1541 Test/Demo disk, or by typing in direct statements.

To use DISK ADDR CHANGE, load the program from the 1541 Test/Demo disk and enter RUN. Follow the program's instructions, then skip to Step 2 below.

You can also change the device number by entering the following statements in direct mode (with no line numbers). Press RETURN after you type each line:

```
OPEN 15,8,15
PRINT#15,"M-W"CHR$(119)CHR$(0)
  CHR$(2)CHR$(32+9)CHR$(64+9)
CLOSE15
```

2. It's a good idea to verify the device number change. Put a disk in the drive, then type LOAD"$",9 and press RETURN to load its directory. After the blinking cursor returns, type LIST and press RETURN. If you see the directory, the change worked and you may proceed to step 3. If you get an error (probably ?DEVICE NOT PRESENT), turn off the drive and repeat step 1.

3. Turn on the other drive. This drive will remain device 8 (the source drive). Now load and run Archive, inserting the disks as explained in the instructions. The source (original) disk goes in device 8, and the archive (copy) disk goes in device 9. As an additional precaution, you may want to write-protect the source disk by taping over the notch in the sleeve.

In theory you can daisy-chain several drives to a 64, but in fact the 1541 doesn't enjoy sharing the serial bus. The drives should always be turned on one at a time, not simultaneously (as would happen with a power strip). Printer interfaces that draw power from the 64's cassette port are notorious for causing disk errors, and other peripherals can affect system voltage levels even if they're not turned on. Depending on your system, you may need to unplug other peripherals before using Archive.

Initialization

Before it starts copying, Archive *initializes* each disk to test whether devices 8 and 9 are active and if each contains a formatted disk. The initialization command transfers information (disk name, ID, etc.) from the disk into the drive's memory to prepare it for handling the disk.

If this step fails, it means one of the drives is not ready to go. Archive displays the status of both drives and returns you to BASIC. (If you forget to change one of the drives to device 9, Archive can't read its status; press RUN/STOP–RESTORE and proceed as explained below). Enter these lines in direct mode (without a line number) to retry the initialization:

```
OPEN 15,8,15,"I0":CLOSE 15
OPEN 15,9,15,"I0":CLOSE 15
```

When you enter each line, the drive motor should run and its red light should glow. After one or two seconds the red light should go off and stay off, and you should be able to run Archive. If one or both of the red lights blink continuously, turn off both drives and repeat the setup process. The drive makes a knocking sound if you forget to insert a disk or try to use an unformatted disk for the archive.

Is Your Drive Healthy?

In ordinary use your drive works intermittently. It may spend 30 seconds loading a game for you, then sit idle for 30 minutes while you play. Copying a full disk with Archive is far more demanding work, requiring several minutes of continuous running. If one of your drives is misaligned, tends to overheat, or has other mechanical problems, don't be surprised if you experience occasional errors. When an error occurs during the copy process, Archive stops copying, reports the status of both drives, and returns to BASIC.

Such errors are especially likely to crop up when the source disk is nearly full. For mechanical reasons it's harder for the drive to access the disk's outer area than the area near the middle. To make things easy on itself, the drive always starts storing programs in the middle of the disk, leaving the outer tracks empty until there's no room left elsewhere.

Archive's BAM display lets you observe this storage scheme. When the source disk contains only a few programs, they'll all be stored in middle tracks (near track 18). The outermost tracks (1 and 35) are usually the last to be used. If your drive consistently has trouble accessing outer tracks, it's probably misaligned. The same problem can result if the disk was formatted on a badly misaligned drive.

1541 ML Programming

To shorten and speed up the program, all of Archive's 21 variables and pointers are located in the zero page (lowest 256 bytes) of memory.

Zero page machine language instructions run faster and use less memory than instructions that reference higher memory addresses. The computer can find what it needs by checking only one byte, rather than wading through a two-byte address in search of the same information. In time-critical programs like Archive, which execute certain routines many thousands of times a minute, the microseconds you save can add up to a significant difference in running time.

Many programmers have trouble learning to handle disk files in machine language. For those who are interested, here's an outline of Archive's main routines.

$0852-0863	Initialize device 8
$0864-0878	Initialize device 9
$0879-0918	Error—report status
$0919-0981	Read BAM from source disk
$0982-0A32	Display BAM and disk name
$0A33-0A7E	Short NEW destination disk
$0A7F-0A8F	OPEN 3,8,15 command channel
$0A90-0AA0	OPEN 5,9,15 command channel
$0AA1-0AB4	OPEN 4,8,4,"#" buffer channel
$0AB5-0AC8	OPEN 6,9,6,"#" buffer channel
$0B94-0C42	Subroutine—copy a block
$0BCE-0C04	Read block from source disk
$0C05-0C42	Write block to archive disk
$0C43-0C5C	Subroutine—initialize disk
$0C94-0CA4	Subroutine—check error channel
$0CC0-0CCB	String—BAM Block Read (U1)
$0CCC-0CDF	String buffer—short NEW
$0CE0	String—"#" for buffer channel
$0CE1-0CE2	String—"10" to initialize
$0F04-0F0F	String buffer—Block Read (U1)
$0F10-0F1B	String—Block Write (U2)
$0FF3	256-byte data buffer starts here

First the program maps the source disk's BAM on the screen to record which sectors contain data. Then it copies each used sector in turn, reading it from the source disk and writing it to the archive disk. Note that to read a disk sector, you should always use the U1 direct access command rather than B-R (Block Read). Likewise, the U2 command must be used in place of B-W (Block Write). Despite what your user's guide says, the B-R and B-W commands are defective and should never be used. ©

Fontier 128

Tapan Desai

This thoughtfully designed program provides all the features you could ask for in an 80-column character editor for the Commodore 128. The custom characters you create with this program can be used in CP/M and Commodore 64 mode as well as Commodore 128 mode. An 80-column monitor and disk drive are required.

The 80-column video display of the Commodore 128 is a significant improvement over the chunky 40-column display of its predecessor, the Commodore 64. In fact the 128's 80-column resolution compares very favorably with those of machines costing much more. One thing the 80-column screen lacks, however, is a character set of its own: It borrows the familiar 40-column character definitions. These characters do not do justice to the superb resolution of the 80-column screen, since they were designed to overcome the limitations of a coarser screen format.

"Fontier 128" helps you create new 80-column fonts. It gives you complete control over character design and manipulation and makes extensive use of the 128's windowing abilities. The program is entirely menu-driven: All you need to do is choose options from the onscreen menus and follow the program's prompts. Best of all, the fonts you create with Fontier 128 can be installed and used independently with other programs—in CP/M and Commodore 64 mode as well as 128 mode.

Before you run the program, be sure that your 80-column monitor is properly connected and that the 80-column screen is the active display. Also, check that the 40/80 DISPLAY key is depressed to the 80-column position so that the 80-column display will remain active after you press RUN/STOP–RESTORE. Since the program runs in FAST mode, it does not work at all with a 40-column monitor.

Four Windows

When you run Fontier 128, it spends a few seconds initializing, then it displays a screen containing four windows. Here's an explanation of what the windows contain.

The *pattern window* is located at the upper left. All pixel-level work is done here. The window displays an enlarged view of the current character (the one you are editing), the character set number (0 or 1), and the current character number (0–255). The blinking element in the pattern window is the *pixel cursor.*

The *character set window* is centrally placed. It shows all the characters of a set in their true size.

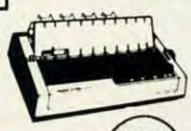
Panasonic Industrial Company

KX-PI080*i* — 120 C.P.S. — $159.95*

KX-P1091*i* — 160 C.P.S. — $179.95*

W/2 RIBBON PURCHASE*

KX-P1092*i* — 240 C.P.S. — $279.95*



KX-P1592 — 180 C.P.S. — $379.95*

1080 i-II
1091 i-II
$Call

SEIKOSHA

* w/2 ribbon purchase

SP180VC — $129.95*

SP1000VC — $139.95*

ALPS ALQ200
24PIN/COLOR

240 C.P.S. DRAFT
1YR. WARRANTY
80 C.P.S. L.Q.
EPSOM/IBM COMPATIBLE

Avatex

1200 Baud . . .	$79.95*
1200 Internal . . .	$99.95
1200 H.C.	$99.95*
2400 Baud . . .	$199.95*

*w/cable purchase

DISKS & ACCESS.

3½ DS/DD (30 lot)	$1.30
3½ SS/DD (30 lot)	$1.00
MOUSE PAD	$6
CASIO CZ101	$250

COMMODORE
5¼ Diskette . . . . . .25¢ (100)
Includes Tyvek Sleeves & label kit

3½ DISK CASE (30)	$8
5¼ DISK CASE (75)	$8
DISK HEAD CLEANERS	$8
1750 RAM	$CALL

SPECIALS

Commodore Logo	$25
Accounts Rec.	$20
Silent Butler	$10
Super Graphix Gold	$CALL
Codewriter	$20
Joy Sticks	from $5
Outlet Surg. Strip	$20

FSD-2

- Quieter
- Cooler
- 1 yr Warr

$149.95*
*Package Pricing

commodore Educator 64

$149.95*
Completely Commodore Compatible

commodore SFD 1001 (1 Meg)

$179.95
4040 . . . $ CALL
8250 . . $ CALL*

commodore Hard Drives

9060
$450*

XETEC/I.C.T. CALL

commodore 160 C.P.S. Printer with PC Interface

$150 Limited Qty.

*Completely Refurbished

AMIGA NEW RELEASES

X-rated Graphics Library	$ 15	Gizmoz 2.0	$ 42	Prownite	$ 75	Earl Weaver Baseball	$CALL
GEN-LOCK	$108	Sculpt 3D	$ 60	Word Perfect	$240	Perfect sound	$ 48
Digipaint 2.0	$ 36	Viza Write	$ 90	Laser Script	$ 27	Acquisition	$180
Faery Tale Adv.	$ 30	Calligrapher	$ 60	Surgeon	$ 36	Videoscape 3D	$120

ALL SOFTWARE AT LEAST 40% OFF

Bard's Tale	$24.00
Chessmaster 2000	$24.00
World Tour Golf	18.00
Marble Madness	18.00
Pegasus	18.00
Moebius	24.00
Murder Party	21.00
Ogre	24.00
Ultima IV	36.00
Ultimate Wizard	24.00
Bard's Tale II	24.00
Auto Duel	30.00
Scrabble	21.00
America's Cup	21.00
Artic Fox	24.00
Many Titles	9.00

MINDSCAPE

Defender of the Crown	NEW $21.00
Bop and Wrestle	18.00
Fist	18.00
Super Star Hockey	NEW 21.00
High Roller	18.00
Indoor Sports	18.00
Infiltrator	18.00
Halley Project	18.00
Parallax	18.00
Uridium	18.00
Luscher Profile	18.00

ACCOLADE

Hard Ball	Special $17.00
Fight Night	18.00
Apollo	$CALL
Law of the West	1800
Test Drive	$CALL
Ace of Aces	18.00
Comics	18.00

PRECISION

Superbase 128	$50.00
Superbase 64	40.00
Superscript 128	40.00
Superscript 64	40.00

50% OFF LIST

Berkeley Softworks

GEOS	Special $34.00
GEOS Fontpak 1	18.00
GEOS Desk Pack	21.00
Writer Work Shop	30.00
Geodex	24.00
GeoCalc/GeoFile	NEW 30.00
GeoCable	NEW 24.00
GEOS 128	$CALL
Geopublish	36.00
Geoprogrammer	$CALL

Abacus

GEOS In and Out (Disk)	9.00
Basic 128	$36.00
Basic 64	24.00
Cadpak 128	36.00
Cadpak 64	24.00
Chartpak 128	24.00
Chartpak 64	24.00
Cobol 64	24.00
Cobol 128	36.00
Super C Compiler 64/128	36.00
Super Pascal	36.00

Digital Solutions Inc.

Super Pack 64	Special $36.00
Super Pack 128	45.00
Pocket Writer 2/Filer/Planner	EACH 36.00
Super Pack 2 (128)	60.00
Pocket Writer 128/Filer/Planner	EACH 30.00
Pocket Writer 64/Filer/Planner	EACH 24.00

ACTIVISION

Portal	$24.00
Titanic	18.00
Labrynth	21.00
Aliens	21.00
Last Ninja	$CALL
Hacker II	21.00
Too Fuel Elim	$CALL
Game Maker	24.00
Music Studio	18.00
Maniac Mansion	$CALL

SPRINGBOARD

Certificate Maker	Special $28.00
Newsroom	$30.00
Clip Art 1	18.00
Clip Art 2	24.00

FIREBIRD

Knight Ork	NEW $24.00
Colossus IV	21.00
Golden Path	$CALL
Frankie Goes Holly	21.00
Pawn	24.00
Talking Teacher	24.00
Guild of Thieves	NEW 24.00

UNISON WORLD

Art Gallery	$16.00
Art Gallery 2	16.00
Print Master	21.00

HI TECH

Card ware	$6.00
Heart ware	6.00
Party ware	6.00
Ware with all	9.00

Broderbund Software

Printshop	Special $25.00
Graphics Lib. EA	15.00
Carmen Sandiego	21.00
Karateka	18.00
Print shop comp.	21.00

TIMEWORKS

Data Mgr. 128	$36.00
Swiftcalc 128	36.00
Wordwriter 3	30.00
All business titles	36.00
Partner 64	36.00
Partner 128	42.00
Sylvia Porters	36.00
Desktop Publisher	$CALL

MicroProse

Gunship	Special $20.00
F-15	21.00
Project Stealth Fighter	$CALL
Airborne Ranger	$CALL
Silent Service	24.00
Pirates	NEW 24.00
Top Gunner	18.00

Xetec, Inc

Font Master II 64	$30.00
Font Master 128	36.00

Micro L. Baseball

Micro L. Baseball	$24.00
General Mgr.	24.00
Stat Disk	15.00
'86 Team Disk	12.00

SSI

Gemstone Healer	$18.00
Gettysburg	36.00
Kampfgruppe	36.00
Phantasie 2	24.00
Ring of Zelfin	24.00
Road War 2000	24.00
Shard of Spring	24.00
Wizard's Crown	24.00

EDYX

FastLoad	Special $22.50
Bad Warrior	$CALL
Street Sport	24.00
World Karate Champ	18.00
Summer Games II	24.00
Super Cycle	24.00
World Games	24.00
Football	24.00
Multi-Plan	24.00
Winter Games	24.00
Sub Battle	NEW 24.00

subLOGIC

Flight Sim II	Special $28.50
Scenery Disk	12.00
Jet	30.00
Football	24.00
Baseball	30.00

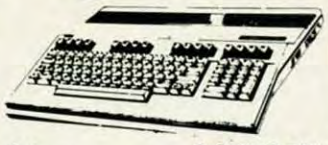
ACCESS Software Incorporated

MACH-128	$28.50
Leaderboard	24.00
Echelon	27.00
World Class leaderboard	Special 22.50
10th Frame	24.00
Exec Tournament	12.00
Tournament 1	12.00
Triple Pak	12.00
Famous Course	12.00

NEW SOFTWARE/BOOKS ARRIVING DAILY — $CALL

PRICES MAY BE LOWER! . . . . $CALL

commodore



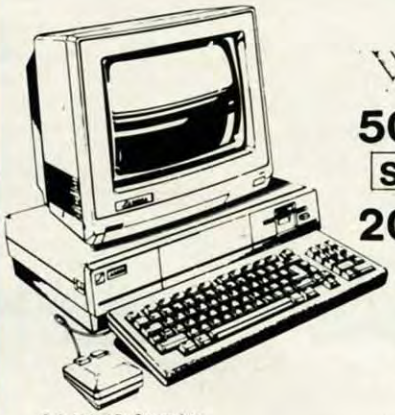
64c $129.95*
128 . . . . . $189.95*
128D . . . . . $Call
C64 . . (REFURBISHED) . $99.95
*package pricing

EXCEL-71 . . . . . $Call
1581 . . . . . $199.95
1541c $169.95
1571 . . . . . $219.95

2002A . . . . . $CALL
1901 (monochrome) $99.95
1802c $189.95
1902A $249.95

OKIMATE 20 . . . $120.00
1351 . . . . . $39.95
Plug & Plot . . . . . $CALL
Roland Plotters . . . . . $Call
1764 RAM . . . $139.95

SIDECAR NOW AVAILABLE . . . $699.95



AMIGA

500 . . . . . $Call
SPECIAL PROMO
2000 . . . $Call
LAST CHANCE
A1000 . . . $650
A1000R . . . $500

*Refurbished By Commodore

GENLOCK . . . $220.00	DIGI-VIEW $120.00	1010 DRIVE . . . $200.00 40 MEG $900.00
1680 MODEM . . $120.00	FUTURE SOUND $120.00	2 MEG $349.95 20 MEG $649.95
256K $80.00	MARAUDER II $24	1 MEG WITH CLOCK $280.00

G064 EMULATOR $99.95

star micronics PRINTERS

star micronics NP-10 *129.95
star micronics NX-10 *149.95

NB2415 • NR15 • LASERS • $CALL

star micronics NL-10 *229.95
star micronics NX-15 *279.95

***WITH PURCHASE OF TWO RIBBONS**

commodore IBM COMPATIBLE

- 512K
- 360K Dual Sided Drive
- CGA Board
- One year warranty
- Parallel Port
- Serial Port
- GW BASIC
- DOS 3.2

FREE SIDEKICK BY BORLAND

PC10-1 ~~$499.95~~ $550.00 Alone

NEW LOWER PRICES $CALL

DUAL DRIVE 640K

*With purchase of 1901 or 1902 A

PC10-2 ~~$574.95~~ $650 Alone

20 MEG . . . $299.95 Seagate w/controller 30 MEG . . . $349.95

ORDERS ONLY **800-433-7756**
CUSTOMER SERVICE 313-427-0267 M-F 10-6
IN MICH. 313-427-7713
Monday thru Friday – 10:00 A.M. to 8:00 P.M.
Saturday – 10:00 A.M. to 6:00 P.M. (E.S.T.)

SCHOOL P.O.'s ACCEPTED CALL FOR TERMS

M.C.S.

MICROCOMPUTER SERVICES

DEALER INQUIREES INVITED

No Surcharge for MC/VISA/DISCOVER. Sorry no walk in traffic.
All returns must have RA #. Merchandise found defective will be repaired or replaced. All items are new. We do not offer refunds for defective products or for products that do not perform satisfactorily. We make no guarantees for product performance. Any money back guarantees must be handled directly with the manufacturer. Call for shipping & handling info.
12864 FARMINGTON ROAD, LIVONIA, MI 48150 We cannot guarantee compatibility.

The blinking character in this window marks the current character and is referred to as the *character cursor*.

The *dialog window* on the right displays prompts and receives input from you. When you begin the program, this window shows a menu of special keys.

The *menu window* is a static portion of the screen which displays the various options available to you.

One-Key Commands

Every command in Fontier 128 is invoked with a single keystroke. Here's a description of what each command key does.

General Commands

Key	Description
cursor keys	Move the pixel cursor in the pattern window.
f1	Move character cursor up.
f3	Move character cursor down.
f5	Move character cursor left.
f7	Move character cursor right.
STOP	Restore original (40-column) character set and stop program.
A	Alternate between character sets 0 (uppercase/graphics) and 1 (lowercase/uppercase).

Character Manipulation Commands

Key	Description
+	Turn pixel on and move cursor.
−	Turn pixel off and move cursor.
R	Reverse character(s) and move character cursor.
I	Invert character.
@	Rotate character clockwise.
SHIFT–CLR/HOME	Clear current character.
f2	Slide character pattern up.
f4	Slide character pattern down.
f6	Slide character pattern left.
f8	Slide character pattern right.
M	Memorize character(s) starting from the current character.
C	Copy memorized character(s) from the current character onward. (An M command should always precede this command.)

Disk Commands

Key	Description
S	Save Font file. Files saved by this option are program files. They may be loaded and RUN like any BASIC program in 128 mode. Press RETURN to abort.
L	Load Font file (don't try to load a file created by any other program). Press RETURN to abort.
$	Display disk directory.

What You See Is What You Get

When you change a character definition in the pattern window, the change appears immediately in the character-set window, too. The advantage of this system is that you can see immediately how a character will look in its normal size and in relation to other characters in the set. The menu and dialog windows always use the character set not being edited at that time. If you're editing uppercase/graphics characters (set 0), all text in the dialog and menu windows will use the lowercase/uppercase set (set 1), and vice versa. It's best to edit only one character set at a time. For example, if you change all the letters in the uppercase/graphics set to new shapes and then press A to edit the lowercase/uppercase set, the text in the menu and dialog windows will use those new shapes and will be unreadable. The solution is to complete all your editing of one character set, save your changes, and then restore the original character set before switching to edit the other set.

Once you've created and saved a custom character set, how do you use it? With Fontier, it's easy. When you save a character set, the program automatically adds a routine to the character set data that will install the new character set for you. You don't need Fontier to load the new character definitions; simply load and run the font file as you would a BASIC program. For example, if you use Fontier to design an italic character set and save it using the name ITALIC.FNT, you can enable your custom character set at any time by typing RUN "ITALIC.FNT". You must not have a hi-res graphics area allocated when you run the font file. For example, Fontier allocates the hi-res area (see the GRAPHIC 1 statement in line 100) to create an area of reserved memory. You must reclaim this space before running a font file. Use the statement GRAPHIC CLR to deallocate any existing graphics area. The built-in font loader program will delete itself after the new font is loaded. (You should note, however, that loading and running the font file will overwrite any existing BASIC program.)

Once a new character set is in place, it behaves exactly like the original character set. It is not affected by RUN/STOP–RESTORE, and will be preserved intact if you switch to CP/M or Commodore 64 mode. For instance, you can run an 80-column CP/M word processor or telecommunications program with your own, personal character set. The same is true of any program written for a 128 in Commodore 64 mode that takes advantage of the 80-column display. Of course, any 80-column program that installs its own custom character set will overwrite the Fontier 128 character set. When using a Fontier 128 font in CP/M mode, keep in mind that CP/M always uses character set 1. You will not see any custom characters from set 0 in CP/M mode unless you invoke the *alternate character set* escape sequence (ESC G1).

Fontier 128 opens exciting possibilities in font design. For instance, 80-column word processors may now include foreign-language character sets, mathematical or scientific symbols, italics, or subscript and superscript characters. Special characters can be used to form background textures in charts or graphs, and even to build shapes in arcade-type games. ©

Expandable Graphics Dump

For The Commodore 1526 And MPS-802 Printers

Fred Solmer

Here's a quick and easy way to get standard- or double-sized high-resolution printouts. The program works with Print Shop, Doodle, Koala Pad, *and other hi-res screens. Versions for the 64 and 128.*

The April 1985 issue of COMPUTE!'s GAZETTE contained a useful program titled "1526 Hi-Res Screen Dump," which allowed 64 owners to print out high-resolution screens on Commodore's 1526 or MPS-802 printers—models that aren't normally capable of high-resolution graphics printing. "Expandable Graphics Dump," for the Commodore 1526 and MPS-802 printers *only*, goes a few steps beyond. It provides for column placement of the normal screen dump, allowing it to be printed out anywhere from column 0–40. Also, a blowup option is included. With a single keypress, you can get a printout twice the size (horizontally and vertically) of the normal one. Two hi-res screens could be combined into an 8½" × 11" "poster."

Expandable Graphics Dump works with almost any hi-res screen. The 64 version prints graphics screens created with the *Print Shop* ("Screen Magic" pictures only), *Doodle*, and the Koala Pad. In fact, it's compatible with almost any software designed to make hi-res screens.

A Simple Procedure

Before getting started, be sure your printer is turned on. If you own a 64, first load your hi-res screen as if it were a machine language program: LOAD"*screenname*",8,1. You probably won't be able to see it, but it has been loaded into memory. Then type NEW. Next, load Expandable Graphics Dump with the command LOAD "EXPANDABLE 64",8 and type RUN. The screen prompts make it very easy to use. In the 64 version, you're first asked if the hi-res screen was created by *Print Shop*. After you press the appropriate key, you're asked to choose a normal size or enlarged, double-size printout. After making this selection, you're asked for a "space-over" value, 0–40. A computer screen is 40 columns wide, but the printer has 80 columns. The space-over number controls the number of spaces between the left margin and the printed picture. For example, if you wish your hi-res screen to be printed at the left margin, choose 0. To center it on the page, enter a value of 20.

On the 128, you may load Expandable Graphics Dump first with the command LOAD "EXPANDABLE 128",8. Then use the built-in graphics commands to create a picture in GRAPHIC 1 mode. When you're satisfied with the result, load Expandable Graphics Dump (if you haven't already done so) and run it. In this version, you're asked to choose a "1:1" (normal size) or blow-up printout, and then a space-over value, as described above. On the 128, you can draw a picture using the built-in graphics commands, exit to the text screen (with GRAPHIC 0), and run the program. It's not necessary to save the picture to disk and load it back into memory.

Note that the 64 version of Expandable Graphics Dump is written for hi-res screens which can be saved as disk files. This restriction means it can't be used to print the greeting cards from *Print Shop*, although it works well with the "Screen Magic" portion of the program. Also, Expandable Graphics Dump handles monochrome hi-res screens somewhat better than multicolor screens. ©

Hi-Res Screen Dump

Gregg Peele

Have you ever created a hi-res picture or graph and then tried to reproduce it on your printer? This program allows you to do just that. The prgram is compatible with the Commodore 1525 or MPS-801 printers (but not the 1526).

The Commodore 64 allows you to create high-resolution graphics images on the video screen. With the VIC or 64 Super Expander cartridge or another hi-res program, it's easy to produce detailed artistic creations. However, most of these programs don't provide a method of printing out these artistic endeavors once you've finished them. Unless you leave your computer turned on indefinitely, your creation is short-lived.

"Hi-Res Screen Dump" works with a Commodore 1525 or compatible printer. (Note that the new 1526 printer from Commodore is *not* compatible with the 1525, and will not work with this program.)

Bit Transfer

Hi-Res Screen Dump is designed to transfer the bit information from screen memory to the printer. Since the 1525 printer can only accept seven bits of data at a time in graphics mode (the high bit must always be set), the eight-bit bytes in screen memory must be split into odd units before they are sent to the printer. Transferring the information from screen to printer is further complicated since the location of screen memory bytes must also be calculated, and hi-res screens for the 64 can be moved to several different areas of memory.

This program reads data from the screen one bit at a time starting from the lower leftmost corner of the screen. After seven bits, the program moves to the leftmost bit of the next row up and prints seven more bits, continuing up the screen. After the leftmost seven-bit column has been printed, the program starts at the eighth bit over from the bottom left corner and continues cycling from bottom to top until the entire screen has been read. Each seven bits are combined to form the byte to be sent out to the printer. Since the program reads from the left bottom side of the screen to the right top side, the printout is a 90-degree-turned reproduction of the screen image.

Hi-Res Screen Dump is written in machine language. A BASIC loader (the first several lines of the program) puts the machine language (in the form of DATA statements) into the appropriate location in memory. The BASIC loader also prompts you for the width of the printout. To operate the program correctly, *you must load and run Hi-Res Screen Dump before you load the program which creates the hi-res image.*

Selecting A Width And Making A Printout

You can select either a single-width or double-width printout by POKEing a 1 (for single width) or a 2 (for double width) into location 2 (i.e., POKE 2,1 or POKE 2,2). This location is changed by your selection of width when you are prompted in the BASIC program, but can be changed at any time. A SYS to location 52224 will initiate a printout of the hi-res screen. You can issue this SYS in direct mode if you have a design on the screen, or add it to a hi-res drawing program if you make sure the machine language is loaded into memory before the SYS is encountered. Also, be sure that the printer is turned on before giving the SYS.

The machine language for the program resides at the top of the 64's free block of RAM above location 49152 ($C000). This makes it compatible with the *Super Expander 64*, but also means that it cannot be used with the 64 DOS wedge program, as both occupy the same area of memory. The program is designed to print the hi-res screen that is currently visible. If you want a screen dump when you are not in hi-res mode, POKE location 900 with the high byte of the starting address of the hi-res screen and SYS to location 52224+32. This alternate SYS bypasses the routine which determines the location of the hi-res screen. For example, if your hi-res screen starts at location 57344 ($E000)—as in "Screen-80"—you would initiate the screen dump with:

```
POKE 900,(57344/256): SYS 52256
```

Sprite Magic:
An All-Machine-Language Sprite Editor

Charles Brannon

Sprites make animation on the 64 fun and easy to program. But actually drawing and creating sprites with graph paper can be tedious. "Sprite Magic" simplifies their creation, and lets you concentrate on the artistic aspects of sprite design.

Most of the what you've read about sprites covers how to program them: setting them up, protecting memory, moving and animating them, and using them in games. But sprite design is usually left up to you.

A sprite is defined by 63 binary numbers. The one bits (on) represent solid pixels. Zeros (off) represent blank areas in which the screen background is visible. Normally, you sketch a sprite on a grid 24 squares across and 21 squares high. This is three bytes per row (8 bits*3 bytes=24 bits) and 21 rows of bytes (3*21=63 bytes). But after you've drawn the sprite, you have to convert the squares into binary, and then into decimal so that you can put the numbers in DATA statements.

There are utility programs that will do the conversion for you, even editors that let you clear and set squares with a joystick. Since you're using a computer, other functions can be supported to let you clear, invert, reflect, reverse, shift, and test out your sprite. The more work the computer does, the less you have to think in terms of binary numbers.

Sprite Magic offers the best features of most sprite editors, including true multicolor mode, and pulls it off with the speed and power of an all-machine language program. Sprite Magic's style (and even some of the coding) is similar to "Ultrafont +." Many of the commands are the same, so you can get up to speed quickly. If you've learned how to use Ultrafont +, it won't take much to become comfortable with Sprite Magic.

Here's how you get it up and running:

```
LOAD "SPRITE MAGIC,8,1
```

Be sure to add the ,1 to the end. After the computer comes back with the READY message, type NEW and press RETURN. This resets some important memory locations, but leaves Sprite Magic in its protected cubbyhole at $C000.

Doodle

Activate Sprite Magic with SYS 49152. Instantly, the main screen should appear, with a large 24 × 21 grid. The grid is a blow-up of the sprite you are editing. The actual sprite will be seen to the right of the grid. The flashing square within the large grid is your cursor. Move the cursor with either the cursor keys or with a joystick plugged into port 2. To light up a blank spot (in other words, to turn that pixel on), press either the space bar or the joystick fire button. If the square is already lit, it will turn dark. This signifies that the pixel has been turned off. The button or space bar thus toggles points on or off. You can draw your sprite quite easily in this manner. One fine point: With the joystick, you can hold down the fire button and move the cursor. If the first point you change was set, then the fire button will continue to set points as you move the joystick, regardless of the other points' original state. If the first point you change was empty, then you can hold down the fire button and move about, clearing anything the cursor passes over. Notice how any changes are immediately visible in the actual sprite.

If you've just entered Sprite Magic, the grid is probably full of garbage pixels. To clear out the grid for a new picture, press SHIFT-CLR/HOME. You now have an empty area (a fresh canvas, so to speak) to draw upon. You can press CLR/HOME without holding down SHIFT to home the cursor to the upper-left corner of the grid.

Does the cursor move too slow or too fast? To change the velocity (speed) of the cursor, press V. Answer the prompt with a number key from 0 (slow) to 9 (very fast).

Shift, Expansion, And Symmetry

Sometimes when you're drawing, it's necessary to reposition the shape within the grid. The first two function keys let you shift the sprite shape around within the grid. If you shift something out of the grid, it wraps around to the opposite side. The f1 key shifts right, f3 shifts down. Use the SHIFT key along with the function key to move in the opposite direction: f2 moves the

Lyco Computer

Marketing & Consultants

Since 1981

* Special Purchase

star micronics inc

NP 10

- 100 cps draft
- 25 NLQ
- EZ Front Panel Selection
- Auto Single Sheet Feed
- Adjustable Tractor Feed

(Limited Quantity) **$129.95**

* The Real 24-Pin Printer

SEIKOSHA

SL-80Ai **$299.95**

- Letter quality 54 cps
- Quiet (52 dBA)
- Automatic paper loading
- 16 K buffer

PRINTERS

STAR MICRONICS

NP-10	$129
NX-10	**$159**
NL-10	$195
SD-10	$229
SD-15	$349
ND-10	$265
ND-15	$389
SR-10	$379
SR-15	$439
NB24-10	$399
NB24-15	$569
NX-15	$295
NB-15	$769
NR-15	$459

BROTHER

M1109	$195
HR10 Daisy	$195
M1409	$329
HR20	$339
M1509	$365
M1709	$475
HR40	**$579**
2024 letter quality	$665
Twinwriter Dot & Daisy	$$849

SILVER REED

EXP 420P	$209
EXP 600P	$539
EXP 800P	**$649**

CITIZEN

120D	$165
MSP-10	$269
MSP-20	**$299**
MSP-15	$325
MSP-25	$389
Premiere 35	$469
Tribute 224	$625

OKIDATA

Okimate 20	$119
Plug N Print	$CALL
120 NLQ	**$189**
180	$219
182	$229
192+	$315
193+	$455
292 w/interface	$455
293 w/interface	$595

EPSON

LX800	$169
FX86e	$299
EX800	**$365**
FX286e	$419
LQ800	$429
EX1000	$489
LQ1000	$595

SEIKOSHA

SP180Ai	$129
SP180vc	$145
SP1000vc	$149
SP1000i	$159
SP1200vc	$NEW
SP1200Ai	$165
SP1200AS RS232	$165
SL80Ai	**$299**
MP1300Ai	$299
MP5300Ai	$375
BP5420Ai	$889

Versatile and IBM Compatible

Panasonic

1091i

- 160 cps Draft Mode
- 32 cps NLQ
- Word Process
- Friction Feed

(2-year warranty)

$169.95

PANASONIC

1091i	$169
1080i	**$139**
3131	$259
1092i	$295
3151	$379
1592	$379
1595	$419
1080i Model 2	$154.95
1091i Model 2	$174.95

DIABLO

D25	$499
635	**$779**
D-80IF	$1299

TO ORDER
CALL TOLL FREE
1-800-233-8760
IN PA
(717) 494-1030
OR SEND TO
LYCO COMPUTER
P.O. BOX 5088
Jersey Shore, PA 17740

Risk Free Policy

In-stock items shipped within 24 hrs. of order. NO deposit on C.O.D. orders. Free shipping on prepaid cash orders within the Continental U.S. Volume discounts available. PA residents add sales tax. APO, FPO and international orders add $5.00 plus 3% for priority mail. Advertised prices show 4% discount for cash, add 4% for MasterCard and VISA. Personal checks require 4 weeks clearance before shipping. We cannot guarantee compatibility. We only ship factory fresh merchandise. Ask about USP Blue and Red Label shipping. All merchandise carried under manufacturers' warranty. Return restriction applicable. Return authorization required. We check for stolen credit cards. We accept purchase orders from educational institutions. All items subject to change without notice.

1-800-233-8760

PERIPHERALS

IBM

MONITORS

Thompson:

230 Amber TT1/12"....$85
450 Amber TT1/15"....$149
980 White TT1/12"....$695
4120 CGA.....$229
4160 CGA.....$289
4460 EGA.....$359
4375 Ultra Scan.....$525

Panasonic:

1320.....$369
1361.....$459

Zenith:

ZVM 1220.....$89
ZVM 1230.....$89
ZVM 1240.....$119
ZVM 1330.....$435

THOMSON

A sight for sore eyes.™

4120 Monitor

- 14" RGB/Video Composite Analog
- Compatible with IBM
- RGB Data Cable Included

$229.95

MODEMS

Avatex:

1200i.....$99
1200hc.....$99.95
2400.....$219
1400i.....$CALL

Hayes:

Smartmodem 300....$125
Smartmodem 1200...$369
Smartmodem 2400...$559

SOFTWARE

Activision:

Pebble Beach Golf $24.95
Champ. Baseball $24.95
Champ. Basketball $24.95
Zork Trilogy $39.95
Leather Goddesses $24.95
Moonmist $24.95

Microprose:

Conflict in Vietnam $24.95
Crusade in Europe $24.95
Decision in Desert $24.95
F-15 Strike Eagle $20.95
Silent Service $20.95
Gunship $28.95

Microleague:

Microleag. Baseball $23.95
General Manager $18.95
Stat Disk $15.95
'86 Team Disk $11.95

Epyx:

Apshai Trilogy $14.95
Create A Calendar $17.95
Destroyer $24.95
Karate Champ $19.95
Movie Monster $14.95
Rogue $19.95
St. Sports Basketball ... $24.95
Sub Battle Simulator ... $23.95
Winter Games $24.95
World Games $23.95

Strategic Simulations:

Road War 2000 ..... $24.95
Kampfgruppe $35.95
Shard of Spring $24.95
Battle of Antetiem $35.95
Computer Baseball $14.95
Gettysburg $35.95
Wizards Crown $24.95
Chickamauga $35.95

Sublogic:

Jet Simulator $31.95
Scenery Japan $17.95
Scenery San Fran. $17.95
Scenery #1-#6 ..... $69.95

Broderbund:

Ancient Art of War $28.95
Print Shop $32.95

Unison World:

Art Gallery 2 ..... $18.95
News Master $55.95
Print Master $36.95

Firebird:

Starglider $26.95
Guild of Thieves $26.95

Access:

Wld. Cl. Leader Board . $CALL

Learning Company:

Think Quick $29.95
Reader Rabbit $24.95
Math Rabbit $24.95
Writer Rabbit $24.95
Magic Spells $22.95

COMMODORE PERSONAL COMPUTERS!

PC 10-1

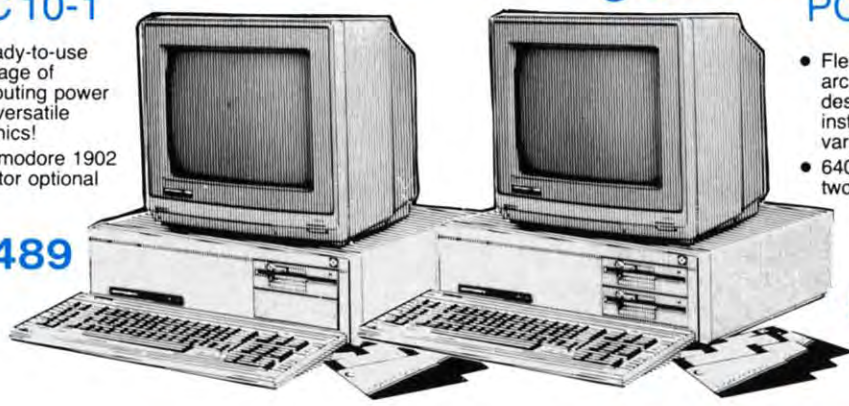
- A ready-to-use package of computing power and versatile graphics!
- Commodore 1902 monitor optional

$489

PC 10-2

- Flexible "open architecture" design for easy installation of variety of options
- 640 K RAM and two 5¼" drives

$589



BLUE CHIP

PERSONAL COMPUTER

- 100% IBM PC/XT compatible!

$499

(Green, amber & color monitors available)

PC COMPATIBLE HARDWARE

PC 5¼ Drive	$119
MS DOS + Basic Soft	$59
BCM 12G Gr. Monitor	$89
BCM 12A Am. Monitor	$89
BCM 14C Color RGB Monitor	$269
BCC CG Color Card	$94.99
BCPC PC/XT Comp	$559
Avatex Graphics Card	$169
Zuckerboard	$CALL
Iomega Bernoulli	$CALL
Hercules	$CALL
AST	$CALL

ATARI® PC

~~499~~

Will Not Be Undersold!

Seagate HARD DRIVES

Hard Drives for PC Compatibles

20 MEG $275.95
30 MEG $315.95
40 MEG $489.95

(DRIVES INCLUDE CONTROLLER)

Attention Educational Institutions:

If you are not currently using our educational service program, please call our representatives for details.

MasterCard

VISA

PRICE GUARANTEE

Since 1981, we have led the industry by continuing to offer the lowest national prices while providing quality service. Many companies have come and gone trying to imitate our quality and service. If by some oversight we do not have the lowest prices advertised on the products you desire, then we would appreciate the opportunity to rectify this oversight.

sprite shape left, f3 up.

After you've drawn something, press F. Instantly, the sprite is flipped upside-down. Press it again to flip it back over. Remember F as the command for Flip. Now try M, for Mirror. The shape you've drawn is mirrored left to right. Of course, if you've drawn something symmetrical, you may not see any change.

Now try CTRL-R or CTRL-9. The sprite will become reversed. Every square that was on is now turned off, and vice versa.

A sprite can also be expanded or contracted either horizontally or vertically, or both horizontally and vertically. The X and Y keys on the keyboard let you do this. Press X to switch from wide to narrow, or vice versa. Press Y to switch from tall to short, or vice versa. Regardless of your choices, the main grid will not change size or proportion.

An unusual command is Symmetry. I added this command after some suggestions that many shapes are symmetrical from left to right, as if a mirror were put in the middle of the grid. To enter the Symmetry mode, press the back-arrow (←) key (found in the upper-left corner of the keyboard). Now, every square drawn on one side will be instantly mirrored to the left. Blank squares are not copied over, though, so you cannot erase in this mode. This command is not only quite useful, but is also a great deal of fun to play with. To return to normal editing, press the back-arrow key again.

Notice the number in the upper-right corner of the screen. This is the sprite page number, which can range from 0 to 255. You start out at the top of the sprite memory. The + and − keys are used to go forward or backward through sprite shapes. Press the minus key and see how you now have a new shape in the grid.

There is a limit to how far back you can go. If you have no BASIC program in memory, you can step back to sprite page number 32. However, character information resides in sprite pages below 128. You can still clear the page and draw a sprite shape on pages below 128, but it won't really register. To be safe, use only the sprite pages from 128 on up. If you have a program in memory, Sprite Magic will not let you step back past its end. This protects your program from being accidentally overwritten by a sprite shape. If you want maximum space available for sprite shapes, be sure to NEW out any BASIC program before you SYS 49152. You'll sometimes want to keep a program in memory, however. We'll show you why a bit later.

Programming note: The sprite page number, when multiplied by 64, gives you the starting memory location for the 63 numbers representing the sprite.

Quick Reference Chart

B: Cycle through background colors
F: Flip sprite upside-down
J: Move sprite with joystick. Press button when done.
L: Load sprite from tape or disk
M: Mirror sprite from left to right
S: Save sprite(s) to tape or disk
V: Set cursor velocity
X: Toggle X expansion on/off
Y: Toggle Y expansion on/off

CTRL-D: Create DATA statements
CTRL-R or CTRL-9: Reverse sprite
CTRL-X: Exit to BASIC

+: Next sprite page
−: Previous sprite page
CLR/HOME: Home sprite editing cursor
SHIFT-CLR/HOME: Erase grid
Space bar or fire button: Set/clear points
CRSR keys or joystick in port 2: Moves cursor
Back arrow: Symmetry mode
Keys 1-4: Select drawing color for multicolor mode
SHIFT 1-4: Change a drawing color

f1: Shift right
f2: Shift left
f3: Shift down
f4: Shift up
f5: Multicolor mode
f6: Normal mode
f7: Store sprite to buffer
f8: Recall sprite from buffer

Put It In The Buffer

You might use Flip to design two views of a shape, such as a spaceship pointing in two directions. Draw one freehand, then do the other with Flip. Mirror can be used to design separate left and right views as well. But what you first need is a way to copy the original shape to another sprite area. One way to do this is to copy the sprite shape to an area of memory (a buffer). You can use + or − to step to another sprite page, then copy the buffer to the sprite. This, you may remember, is the way you copy characters with Ultrafont +. The same keys are used in Sprite Magic. Press f7 to copy the sprite to the buffer. The grid flashes to affirm this. Then go to the sprite page where you want to put the copy and press f8 (SHIFT-f7). The shape in the buffer replaces any shape already in the sprite grid. You can also use the buffer as a fail-safe device. Before modifying an existing sprite, press f7 to save it in the buffer. Then, if you mangle the sprite, or accidentally erase it, you can recall the previous shape from the buffer.

Computer Disney?

The buffer is also useful for animation. Since you can change sprite pages so easily, you can also use Sprite Magic as an animation design tool. Cartoons make only minor changes between frames. Too much change makes the animation jerky. So put the first frame into the buffer, copy it to the next area, then make a change. Put the new image into the buffer, copy it again to a new area, then make another small change. Continue in this fashion as you build up a whole series of frames. Put different but similar shapes on adjacent pages, then hold down plus or minus to step

through the shapes. As with cartoon animation, you will get the illusion of motion. Use a cursor velocity of 9 for maximum speed. So even if you don't care to program sprites, Sprite Magic is a fun tool for making moving cartoons.

A Bit Of Color

The normal drawing mode lets you set or clear points, but in only one color. If you're willing to give up half as many horizontal points, you can have four colors to work with. Multicolor mode lets any square be one of four colors, but gives you only 12 pixels across instead of 24. This is because two dots are grouped together to give four combinations. The colors come from four memory locations:

Pattern	Color location	
00	53281	Background color register
01	53285	Sprite multicolor register 0
10	53287-53294	Sprite color registers
11	53286	Sprite multicolor register 1

There are two multicolor sprite registers, which are shared among all sprites (in programming, but not in Sprite Magic, you can have eight sprites on the screen at the same time). The bit pattern marked 10 is unique to each sprite, and comes from that sprite's own color register. Pattern 00 is blank, and whatever is underneath the sprite shape will show through.

The reason for this sojourn into bits and addresses is that only the 10 bit pattern has a unique color for that sprite. If you're designing several sprites for a game, remember that anything drawn in that color can be changed individually for each sprite. Squares drawn with bit pattern 01 or 11 will be colored from two locations shared by all sprites.

Many sprite editors let you see how the sprite would look in multicolor, but you still have to pair up the pixels yourself, and keep track of binary bit pairs. No fun! Instead, Sprite Magic offers a multicolor mode. When you press f5, the screen instantly changes. Each square in the grid is now rectangular, two squares wide. The cursor has also been enlarged, and can be moved about as before in the new grid. But the way you set and clear points has been changed, since you are now working with four colors.

Multicolor Palette

The fire button or the space bar always sets a point, but you have to tell Sprite Magic which color you are currently drawing in. The number keys 1 to 4 select the drawing color. The number you press is one number higher than the binary value of the bit pairs in the table above. The 1 key, for instance, chooses the 00 bit pair, which represents the background color. In practice, you are choosing from a palette of four colors. The 1 key can be used when you want to erase, although the fire button can still be used to toggle points on and off.

When you press a number key from 1 to 4, the border color changes to remind you which color you're drawing with. If you want to change one of the four colors, hold down SHIFT while you type the number. The prompt ENTER COLOR KEY appears. Now you have to enter another key combination. Press CTRL and one of the number keys from 1 to 8, or hold down the Commodore key and one of the number keys from 1 to 8. These are the same key combinations you use to change the text color in BASIC. You can also change the screen background color by pressing the letter B on the keyboard until the color you want appears.

Some Sprite Magic commands act strangely in multicolor mode. For example, a shift left or shift right (done with the f1 and f2 keys respectively) moves the sprite over by only one bit, which changes the color assignments. In general, you must press f1 or f2 twice to preserve the same colors. Pressing the M key (for Mirror) reverses the bit pairs, so that every 01 becomes a 10. The effect is that colors 2 and 3 are exchanged. The R key (Reverse) also inverts the bits, so that 01 becomes 10, 10 becomes 01, 00 becomes 11, and 11 becomes 00. Colors 2 and 3 are switched, as well as colors 1 and 4.

If you want to go back to normal (non-multicolor) mode, press the f6 key (SHIFT-f5). There's nothing to prevent you from designing both normal and multicolor sprites on different pages.

If you changed colors in the multicolor mode, some of the colors in the normal mode may have been changed. You can alter these colors as in multicolor mode. Press SHIFT-1 to change the color of the empty pixels, and SHIFT-2 to change the color of the on pixels. (You'll be prompted to press a color number key after each SHIFT-1 or SHIFT-2 combination. Remember to press either CTRL or Commodore simultaneously with the color key.)

Mobilizing Your Sprite

If you want to try out your sprite in action, press J (for Joystick). You can now move the actual sprite around with the joystick. The speed of movement depends on the current cursor velocity. When you've finished putting your sprite through its paces, press the fire button to return to Sprite Magic. Also, if you want to test the animation while you are moving about, hold down the SHIFT key to step forward through the pages of your defined sprites, or the Commodore key to step backward. You can lock the SHIFT key to keep the animation happening while you move around.

Saving Your Sprites

After all your work, you surely want to save your creations on tape or disk for future use. You can save an individual shape, or all the sprites. Press S (for Save), then either D (Disk) or T (Tape). Next, enter the filename. You'll be asked if you want to "Save all from here?" If you press N, for No, then only the current sprite you are working on is saved. If you press Y for Yes, then every sprite from the current sprite to sprite 255 will be saved. Thus, if you want to save a range of sprites, be sure to use the minus key to step back to the first sprite you want saved.

To recall your sprites, press L. The Load command loads everything that was saved. If you're loading in more than one sprite, be sure you step backward far enough with the minus key so that all the sprites will fit between the current sprite and sprite 255. The sprites load

starting at the current sprite page number. After you press L, enter T or D for Tape or Disk.

Making Sprite DATA

If you're a programmer, you're probably more interested in DATA statements. That way, you can use BASIC to READ and POKE the numbers into memory. If you have some kind of "DATA maker," you can run it on the memory used by the sprite in Sprite Magic (again, the memory location is the sprite number times 64). But Sprite Magic has a special DATA maker of its own. It's similar to the Create DATA option in Ultrafont +, but it's been enhanced.

Press CTRL-D to create a series of DATA statements from the current sprite in memory. Just tap the key, or you'll get hundreds of DATA statements as the key repeats. Sprite Magic will create eight DATA statements, with eight bytes per line. The last byte is not strictly used. Sprite shapes are made from 63 bytes, but the sprite areas are padded out so they will conveniently fall in 64-byte ranges. To create DATA statements for another sprite, use the + or − key to move to the correct sprite page, then press CTRL-D again.

If you have a program already in memory, the DATA statements are appended to the end of the program, starting with the next available line number. To add DATA statements to an existing program, then, first load Sprite Magic. Type NEW. Load your BASIC program, and SYS 49152 to enter Sprite Magic. You can then load in sprite shapes and use CTRL-D to add those DATA statements to the end of the BASIC program in memory.

You can check to see that these DATA statements were added by exiting Sprite Magic (press CTRL-X) and typing LIST. Your program should have eight new DATA lines for each sprite pattern. If there was no program in memory, the DATA statements form a program all their own, starting with line 1. If you want, you can save just the DATA statements to tape or disk, using the normal SAVE command.

To exit Sprite Magic and return to BASIC, press CTRL-X. You can also use RUN/STOP-RESTORE. ©

Fast Hi-Res Screen Dump

Robert F. Mills

This machine language program prints out screen dumps at high speed when used with the Epson, Gemini, and compatible printers. For the Commodore 128 and 64.

The "Hi-Res Dump" program in the July 1986 GAZETTE prints out excellent copies of whatever is on the hi-res (high-resolution) screen. Unfortunately, the program is very slow when used with a non-Commodore printer like my Gemini 10X with its Cardco +G interface. The interface emulates the Commodore graphics commands well—but the process is slow and it gives the printhead quite a workout. So, I dug out my printer manual to learn more about its built-in graphics capabilities. "Fast Hi-Res Dump" is the result.

Although I wrote this program for my Gemini printer, it also works with most other Star Micronics models, and with most Epson printers as well. In fact, the program will work with most any printer that uses the ESC K *n1 n2* sequence for printing graphics. *The program will not work with any Commodore printer.* Refer to the owner's manual that came with your printer for details of its built-in graphics capabilities. Regardless of the printer used, you must also have an interface which allows a transparent mode of operation. Transparent mode makes the interface pass all characters to the printer without modification. This gives you control over the printer's special built-in functions. This program uses the Cardco convention whereby a secondary address of 4 in the OPEN statement specifies transparent mode. For other interfaces, it may be necessary to set a switch on the interface to select transparent mode. This program does not work with most third-party printers that have built-in Commodore interfaces, such as the Star SG-10C or Gemini II.

Printing A Screen

Fast Hi-Res Dump is relocatable, so you can put this program nearly anywhere in RAM. The program will place the machine language at any address you request. On the 128, either of the RS-232 buffers (starting addresses 3072 or 3328) provide excellent areas if you won't be using an RS-232 device. The free memory starting at 4864 can also be used. For the Commodore 64, the free memory beginning at address 49152 is a good place to put this routine if your screen or another program isn't located there. The program is 221 bytes long, so it won't fit in the cassette tape buffer.

When you run the program, you'll be asked to specify a starting address for the dump routine. The

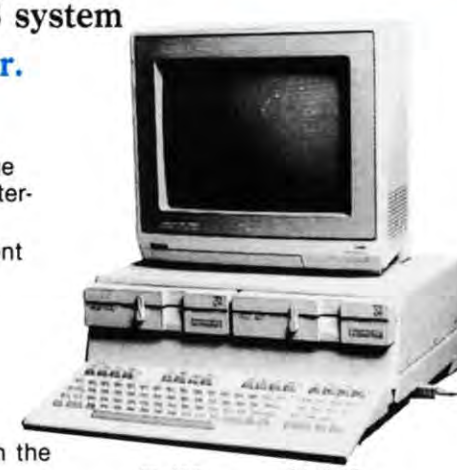
Get your workspace back again.

Consolidate your 64, 64C or 128 system with the Command Center.

☐ 64..... $119.95

Just look at all it includes:

- **Built-in AC Power Strip** with power surge and voltage spike protection, line noise filtering and power outlets.
- **Built-in Drive/CPU Cooling Fan** to prevent overheating.
- **Modular Telephone Plug** with its own on-line/off-line telecommunications switch. (Optional on 64 and 64C).
- **Master AC Switch** for easy system power up.
- **Single or Dual Drive Configurations** with the standard drive insert.



☐ 128..... $149.95
☐ 64C..... $129.95

The Command Center will untangle your wires, unclutter your desk and put peripherals at your fingertips.

COMMAND CENTER ACCESSORIES

(Specify 128, 64, or 64C when ordering)

Dust Cover $19.95 — covers entire system
Keyboard Cover $19.95 — smoked acrylic
Drive Reset $14.95 — one switch per drive
Short Serial Cable $9.95 — for chaining drives
Modem Switch $9.95 — 64/64C option
Tilt/Swivel Monitor Stand $14.95 — see below

(Call for details)

DATA DIRECTOR

NEW

$39.95

- Share two Commodore® 6-pin serial devices with one computer.
- Share two computers with one serial device.
- Works with Commodore® compatible disk drives, printers, and all interfaces.
- LED indicates which device is in use.
- Convenient button resets device.
- Saves wear on disk drive serial ports.
- Eliminates cable swapping.

TILT/SWIVEL MONITOR STAND

NEW

$14.95

- Swivel base adjusts to desired viewing angle.
- Accommodates most popular monitors.
- Tension adjustment holds stand firmly in correct position.
- Padded feet to protect surface.
- Cushioned pads hold monitor securely in place.
- Places monitor at ideal viewing level and position.

SHIPPING AND HANDLING

Continental U.S.
$ 4.50 for each Command Center
$ 2.00 for one accessory item
$ 3.00 for two or more accessories

APO/FPO/CANADA/Puerto Rico/Hawaii/Alaska
$13.00 for each Command Center
$ 4.00 for one accessory item
$ 6.00 for two or more accessories

$2.00 charge for C.O.D. orders
Iowa residents add 4% sales tax

®Commodore is a registered trademark of Commodore Electronics, Ltd.

Free 30-day trial offer
and one-year warranty.

For faster service, call
1-800-626-4582 toll-free
1-319-338-7123 (Iowa Residents)

DEALER INQUIRIES INVITED

KETEK P.O. Box 203
Oakdale, IA 52319

Name _____
Address _____
City _____ State _____ Zip _____
Phone Number _____

loader then POKEs the values into that area and checks for errors. Anytime you want to dump the image on the hi-res screen to the printer, just SYS to the address you specified for the dump routine. For example, if you placed the Fast Hi-Res Dump routine at 49152, use SYS 49152 to dump your graphics screen. This can be done either in direct mode or within a program. The dump takes just over a minute. On the 128, you can BSAVE (binary save) the routine and then BLOAD it wherever you wish without having to use the loader program again. For example, if you told the loader to place the routine at address 3072 ($0C00), the RS-232 input buffer, you could save a copy of the routine with:

```
BSAVE "FASTDUMP",P3072 TO P3293
```

Then when you want to use the routine, just BLOAD "FASTDUMP" and SYS 3072. Or, if you have another ML program residing at 3072, you can use BLOAD "FASTDUMP",P *address* and SYS *address*, where *address* is some other free ML area. For the 128, make sure that the routine is loaded into a bank configuration where Kernal ROM is visible (bank 15, for example).

Behind The Dump

Fast Hi-Res Dump is written entirely in machine language. First, the program opens a channel for talking to the printer (equivalent to OPEN 4,4,4 in BASIC). The secondary address of 4 puts the Cardco +G interface in transparent mode with line feeds. If you don't have a Cardco interface, you may need to change the DATA in line 70. Replace the 4 in that line with the secondary address value that puts your interface into transparent mode. (You can ignore this change if your printer interface instead requires a switch setting to select transparent mode.) If you do change line 70, remember to change the total in line 40 to reflect the new value. Otherwise, you'll get a false error message when you run the program.

Next, the routine sends the character codes to set up the printer for 8/72-inch (versus the normal 9/72) line feed length to account for the fact that the Gemini print head has nine vertical firing pins. (It's more convenient when working with the Commodore graphics screen to deal with only eight.) Changing the line feed length removes the gaps between the lines.

At the beginning of each column to be printed, the sequence ESC K *n1 n2* is sent to the printer. The expression $n1 + n2 \times 256$ gives the number of characters to follow. The program sends ESC K 200 0 for each column because there are 25 rows in the hi-res screen, and each row has eight lines ($25 \times 8 = 200$).

The program prints the screen contents sideways on the paper, with each printed dot corresponding to a lit pixel on the screen. The program scans from the top right corner of the screen down to the bottom right corner, through 25 rows of 8 lines. The address of a particular byte in the screen is $(320 \times row) + (8 \times column) + line + base$, with *column* ranging from 0 to 39, *row* from 0 to 24, and *line* from 0 to 7. The *base* address is the starting address of the screen bitmap. As written, this is assumed to be 8192 ($2000). If you want to dump a hi-res screen located elsewhere, just change the 32 at the end of line 240 to the high byte of the starting address of your screen. For example, if your screen is at 16384 ($4000), then change the 32 to 64. (As before, you'll also have to adjust the total in line 40 to reflect the new value.)

Note that there is a one-to-one relationship between the printed dots and pixels; this makes the final printed copy about 4 × 5 inches.

Like most hi-res dump programs, this program can't be used to dump text screens or sprites, but its speed, small size, and portability make it a very useful and powerful utility. ©

Ultrafont+

Charles Brannon

This powerful custom character editor for the Commodore 64, which originally appeared in the July 1984 GAZETTE, has been one of our most popular programs. Now it's been enhanced, and a few minor bugs have been removed. We think you'll be surprised at its lightning speed, ease of use, and unique features. It's designed to let you concentrate on your artwork instead of programming details.

Anyone who has used graph paper to plot out characters, then tediously converted the rows of dots into numbers can appreciate a character editor. Instead of drawing and erasing on paper, you can draw your characters freehand with a joystick. "Ultrafont+" offers almost every conceivable aid to help you design whole character sets.

Load Ultrafont+ with the command LOAD "ULTRAFONT",8,1. After it's loaded, enter NEW and press RETURN to reset important memory pointers, then start the program by typing SYS 49152 and pressing RETURN.

The Display

At the bottom of the screen you'll see eight lines of characters. These are the 256 characters you can customize, arranged in eight 32-character rows. Note the normal character set in the top half, with reversed images in the bottom half. A flashing square rests on the *at* (@) symbol, the home position of the character set. Above the eight rows is the main grid, a blown-up view of ten characters. The bottom row of the screen is reserved for messages. The first time you SYS to Ultrafont+, you'll be asked whether you want to edit the uppercase/graphics character set or the lowercase set.

About The Grid

The grid is like a large window on the character set. You see the first five characters and the five beneath them. Each character is made up of 64 *pixels* (an abbreviation for "picture elements"), arranged in an 8 × 8 matrix. A large red box shows you which character you're currently editing, and a smaller flashing square is the cursor you use to set and clear pixels while you're drawing a character.

You can use the cursor keys (up, down, left, right) to move the large red cursor to any character you want to edit. If you move to a character not on the large grid (out of the window), the window automatically scrolls to make the character appear. You can also look at the bottom of the screen to move the larger cursor, since the flashing square on the character set moves with the main grid.

The HOME key moves the small cursor to the upper-left corner of the grid. If you press it twice, it takes you back to the top of the character set—to @.

Use a joystick plugged into port 2 to control the small cursor within the grid. If you move the cursor out of the current character, the red cursor jumps to the next character in whatever direction you are moving. The display at the bottom adjusts, and the grid scrolls as necessary. This means that you can ignore the traditional boundaries between characters and draw shapes as big as the entire character set (256 × 64 pixels). There is no wraparound for the cursor in the bottom section of the screen. When it hits an edge, it will go no further in that direction.

The joystick's fire button is used to set and clear points. If you press it when the cursor is resting on a solid square, the pixel is turned off. If the square is currently off, it's turned on. Holding down the button while you move the joystick keeps you in the same drawing mode. If you set a point, you'll continue to draw as you move. If you clear a point, you can move around and erase points all over the screen.

If the drawing cursor is too fast or too slow, just press V to set the cursor velocity. Answer the prompt with a speed from 0 (slow) to 9 (very fast—but too fast for practical use).

Manipulations

There are several functions that affect the current character (where the red box is). You can rotate, shift, mirror, reverse, erase, replace, and copy characters. The best way to learn is to play with the functions. It's really a lot of fun. The following keys control each function.

Function Keys

f1 Scrolls character right. All pixels move right. The rightmost column of pixels wraps around to the left.

f2 Scrolls character left. Wraparound is like f1.

f3 Scrolls character down. All pixels move down. The last row of pixels wraps around to the top.

f4 Scrolls character up. Wraparound is like f3.

R Rotate. Rotates the character 90 degress. Press twice to flip the character upside down.

M Mirror. Creates a mirror image of the character left to right.

CLR (SHIFT-CLR/HOME) Erases the current character.

CTRL-R or **CTRL-9** Reverses the character. All set dots are clear, and all empty dots are set. The bottom half of the character set is the reversed image of the top half.

CTRL-back arrow (←) Copies upper half of the character set, reverses it, and places it in the lower half. This way, you have to redraw only the normal characters, then use CTRL-back arrow to create the reverse character set.

F Fix. Use this if you want to restore the normal pattern for the character. If you've redefined the letter *A* and press F while the red cursor is on the character, the Commodore pattern for *A* will be copied back from ROM.

T Type. This lets you try out your character set. The screen clears, with a copy of the character set provided for reference. You can type and move the cursor around, just as in BASIC. This is handy for envisioning sample screens and fitting together multiple-character shapes. Press the RUN/STOP key to exit from Type and return to Ultrafont+.

Saving And Loading Character Sets

To save your creation to tape or disk, press S, then either T for tape or D for disk. When requested, enter the filename, up to 16 characters. Don't use the 0: prefix if you're using a disk drive; it's automatically added for you. The screen clears, displays the appropriate messages, and then returns to the editing screen if there are no errors. If there *are* errors, such as the disk being full, Ultrafont+ will read the disk error message and display it at the bottom of the screen. (Tape users should remember that the 64 is never able to detect an error during a tape save.)

Press a key after you've read the message and try to correct the cause of the error before you save again.

To load a character set previously saved, press L and answer the TAPE OR DISK prompt. Enter the filename. If you're using tape, be sure the tape is rewound and ready. After the load, you'll be returned to the editing screen; a glance is all it takes to see that the set is loaded. If an error is detected on a tape load, you'll see the message ERROR ON SAVE/LOAD. Once again, if you're using disk, the error message will be displayed. Press a key to return to editing so that you can try again.

Copying And Moving Characters

You can copy one character to another with function keys 7 and 8. When you press f7, the current character flashes briefly, then is copied into a buffer. Ultrafont+ remembers that character pattern. You can position the cursor where you want to copy the character before pressing f8. The memorized character replaces the character the cursor is resting on. You can also use the buffer as a fail-safe device. Before you begin to edit a character you've already worked on, press f7 to store it safely away. That way, if you accidentally wipe it out or otherwise garble the character, you can press f8 to bring back your earlier version.

Creating DATA Statements

A very useful command, CTRL-D, allows you to create DATA statements for whatever characters you've defined. Ultrafont+ doesn't make DATA statements for all the characters, just the ones you've changed. After you press CTRL-D, Ultrafont+ adds the DATA statements to the end of whatever program you have in BASIC memory. If there is no program, the DATA statements exist alone.

If the screen turns to garbage when you press CTRL-D, then you probably didn't type NEW after loading Ultrafont+. The NEW command resets some important pointers after you've loaded a machine language program from immediate mode.

You can load Ultrafont+, enter NEW to reset the BASIC pointers, load a program you're working on, then SYS 49152 to Ultrafont+ to add DATA to the end of the program. The DATA statements always start at line 63000, so you may want to renumber them. If you press CTRL-D twice, another set of DATA statements will be appended, also numbered from line numbers 63000 and up. Since the keys repeat if held down, just tap CTRL-D. If you hold it down, you may notice that hundreds of DATA statements have been created. See the notes at the end of this article for more details on using DATA statements in your own programs.

Exiting Ultrafont+

After you create the DATA, you'll still be in Ultrafont+. If you want to exit to see the DATA statements or go on to other things, press CTRL-X. The screen will reset to the normal colors and you'll see the READY prompt. If you've made DATA, LIST reveals it. It's best to enter the command CLR to make sure BASIC is initialized properly after creating DATA statements.

One thing to watch out for: Don't use RUN/STOP-RESTORE to exit Ultrafont+. The program moves screen memory from the default area at address 1024, and the RUN/STOP-RESTORE combination does not reset the operating system pointers to screen memory. If you do press it, you won't be able to see what you're typing. To fix it, blindly type POKE 648,4 or SYS 49152 to reenter Ultrafont+ so you can exit properly, via CTRL-X.

Reentering Ultrafont+

To restart Ultrafont+ within the program, press SHIFT-RUN/STOP. After you've exited to BASIC, you can rerun Ultrafont+ with SYS 49152. You'll see the character set you were working on previously, along with the message USE ROM SET? (Y/N). Usually, Ultrafont+ will copy the ROM character patterns into RAM where you can change them. If you press N, however, the set you were previously working on is untouched. Press any other key, like RETURN, to reset the characters to the ROM standard. You can copy either the uppercase/graphics set from ROM, or the lowercase set.

A Whole New World Of Multicolor

You're not finished yet. There's yet another mode of operation within Ultrafont+, the multicolor mode. In multicolor mode, any character

can contain up to four colors simultaneously. One has to be used for the background, which leaves three for the character itself. Multicolor changes the way the computer interprets character patterns. Instead of a one bit representing a solid pixel and a zero representing a blank, the eight bits are organized as four *pairs* of bits. Each pair can represent four possibilities: 00, 01, 10, and 11. Each pair is also a number in decimal from 0 to 3, and represents one of the four colors.

Ultrafont+ makes multicolor easy. You don't have to keep track of bit pairs any more than you have to convert binary to decimal. Just press the f5 key. Presto—the whole screen changes. The normal characters are rather unrecognizable, and the drawing cursor is twice as wide (since eight bits have been reduced to four pixel-pairs, making each dot twice as wide). You have only four dots horizontally per character, but you can easily combine several characters to form larger shapes.

Multicolor redefines the way the joystick and fire button work. The fire button always lays down a colored rectangle in the color you're currently working with. That color is shown in the center of the drawing cursor. Press the number keys 1, 2, 3, or 4 to choose different colors to draw with. The number of the key is one more than the bit pattern, so color 1 is bit pattern 00, and color 4 is bit pattern 11. When you first SYS to Ultrafont+, the four colors show up distinctly on a color TV or monitor.

You can easily change the colors. Just hold down SHIFT and press the appropriate number key to change that number's color. You will see the message PRESS COLOR KEY. Now press one of the color keys from CTRL-1 to CTRL-8, or from Commodore-1 to Commodore-8. Hold down the CTRL or Commodore key as you do this. Instantly, that color, and everything previously drawn in that color, is changed.

Three of the colors (including 1, the background color) can be any of the 16 colors. But because of the way multicolor works, color 4 (represented by bit pattern 11, or 3 in decimal) can only be one of the 8 CTRL colors. Assigning it one of the Commodore logo colors just picks the color shown on the face of the color key. Incidentally, it's the color of bit pattern 3 (color 4) that changes according to the character color as set in color memory. The other colors are programmed in multicolor registers 1 and 2 (locations 53282 and 53283), so all characters share these two colors. When you want to vary a certain color without affecting the rest of the characters, you'll want to draw it in color 4.

Some of the commands in the multicolor mode aren't as useful as others. You have to press f1 and f2 twice to shift a character, since they only shift one bit, which may cause all the colors to change. You can use CTRL-R or CTRL-9 (Reverse) to reverse all the colors (color 1 becomes color 4, color 2 becomes color 3, color 3 becomes color 2, and color 4 becomes color 1). R (Rotate) changes all the colors and is rather useless unless you press it twice to just turn the characters upside down. M (Mirror) works as it did before except that colors 2 and 3 are switched. And you can still copy characters using f7 and f8 (see above).

Returning To Normal

You can switch instantly back to the normal character mode by pressing f6 (SHIFT-f5). If you were drawing in multicolor, you can see the bit patterns that make up each color. Multicolor characters look just as strange in normal mode as normal characters look in multicolor.

If you changed colors in the multicolor mode, some of the colors in the normal mode may have been altered. You can change these colors just as you did in multicolor mode. Press SHIFT-1 to change the color of the empty pixels, and SHIFT-2 to change the color of the *on* pixels. Use SHIFT-4 to change the color of the eight rows of characters.

How To Use The DATA Statements

The DATA statements are created from lines 63000 and up, as many as necessary. Each line of data has nine numbers. The first number is the internal code of the character (the code you use when POKEing to the screen). It represents an offset into the table of character patterns. The eight bytes that follow are the decimal numbers for the eight bytes needed to define any character. Here's a sample program to read them and display them:

```
10 POKE 56,48: CLR
50 READ A: IF A=-1 THEN 70
60 FOR I=0 TO 7: READ B
63 POKE 12288 + A * 8 + I, B
66 NEXT: GOTO 50
70 PRINT CHR$(147);"{10 DOWN}":
   REM TEN CURSOR DOWNS
80 FOR I = 0 TO 7: FOR J = 0 TO 31
83 POKE 1028 + J + I * 40, I * 32 + J
86 POKE 55300 + J + I * 40, 1: NEXT:
```

Quick Reference: *Ultrafont+* Commands

Cursor keys	Move to next character
HOME (CLR/HOME)	Moves the cursor to upper-left corner. Press twice to go back to start
V	Cursor velocity; answer from 0 (slow) to 9 (fast)
f1	Scrolls right with wraparound
f2(SHIFT-f1)	Scrolls left
f3	Scrolls down
f4(SHIFT-f3)	Scrolls up
R	Rotates 90 degrees; press twice to invert
M	Mirror image
CLR (SHIFT-CLR/HOME)	Erases current character
CTRL-R, CTRL-9	Reverse pixels
CTRL-back arrow (←)	Copies upper half of character set, reversed, to lower half
CTRL-F	Copies first four rows of characters, reversed, to bottom four
F	Fix characters from ROM pattern
L	Load. Tape or Disk, Filename
S	Save. Tape or Disk, Filename
T	Typing mode: RUN/STOP to exit
f7	Memorizes character (keep)
f8 (SHIFT-f7)	Recalls character (put)
f5	Switches to multicolor character mode
f6 (SHIFT-f5)	Returns to normal character mode
CTRL-D	Makes DATA statements
SHIFT-RUN/STOP	Restarts Ultrafont+
CTRL-X	Exits Ultrafont+ to BASIC

```
NEXT
90 POKE 53272, (PEEK (53272) AND 240)
   OR 12
95 END
```

You'll also need to add the following line to the end of your DATA statements:

```
63999 DATA -1
```

If you want to have your cake and eat it, too—that is, also have the normal ROM patterns—copy them from ROM down to RAM by adding:

```
20 POKE 56334, PEEK (56334) AND 254:
   POKE 1, PEEK (1) AND 251
30 FOR I = 0 TO 2047: POKE 12288 + I,
   PEEK (53248 + I): NEXT
40 POKE 1, PEEK (1) OR 4: POKE 56334,
   PEEK (56334) OR 1
```

To turn on multicolor mode, POKE 53270, PEEK(53270) OR 16. Three color registers are at 53281–53283. The fourth is the individual character color, which is changed by holding down CTRL or Commodore and pressing a numbered color key on the top row of the keyboard.

Custom Characters On The 128

Commodore 128 Users can use Ultrafont+ to create characters for the 128's 40-column screen.

First, go into 64 mode and load Ultrafont+. You'll be using the characters in 128 mode, but you have to create them in 64 mode. When you're satisfied with the new character set, save it to disk, following the instructions above. (It's easier to use a disk file on the 128, so don't use the DATA option.) To load the character set into the 128, add these lines at the beginning of your own program:

```
10 GRAPHIC1: GRAPHIC0
20 BLOAD "characterset", B0, P14336
30 POKE 2604,30: POKE 217,4
```

That's all there is to it. Line 10 allocates memory for a hi-res screen by moving up the start of BASIC. This is necessary to protect the character set from interference from your program. Note that while the custom characters are in this memory, you should not use the hi-res screen. Also, if you use the SPRDEF command to create sprites, the hi-res area is cleared and your character set will be erased. After exiting the sprite editor, press RUN/STOP– RESTORE and BLOAD the characters again.

Line 20 loads the character set you created into locations 14336–16383. Substitute the appropriate filename for *"characterset"* in this line. Line 30 then sets a pointer to 14336. To move it to another location in memory, the character set must start at an even multiple of 2048; divide the starting address by 1024 and add 16 (this is the number to POKE into 2604). The POKE to 217 tells the 128 to look for the character in RAM instead of ROM.

For multicolor characters, a couple of additional POKEs are needed:

```
40 POKE 218,255: BANK 15: POKE
   53270,PEEK(53270)OR16
```

The POKE to 218 turns off the split-screen option, which is accessed by the BASIC commands GRAPHIC2 and GRAPHIC4. Split screens use a raster interrupt to switch back and forth; this interrupt interferes with multicolor mode. Once the interrupt is disabled, it's safe to go into multicolor mode with the POKE to 53270. To change the colors in multicolor mode, use the COLOR command. ©

80 Columns For The 64

Gregg Peele And Kevin Martin

Did you ever wish for an 80-column screen? "Screen-80" transforms your 64 into an 80-column machine without affecting the normal screen editing keys. We've also included "Custom-80," which allows you to create your own 80-column character set with a joystick.

"Screen-80" offers a full 80-column screen and provides you the ability to use your 64 to write, edit, and even run BASIC programs (including some commercial software). You can also use all the cursor controls of the normal screen editor. It runs concurrently with the normal system, allowing a quick switch between 40- and 80-column modes.

Best of all, little memory is used by Screen-80. The program consists of approximately 3K of machine language which goes into RAM "underneath" ROM. There are also 43 bytes which are placed in a little used area of RAM (locations 710–753). Since the bulk of Screen-80 uses the same memory locations as the operating system, and the location of the 43 bytes used from RAM are normally unused anyway, Screen-80 works without any apparent loss of programming space.

Using 80 Columns

Once you enter 80-column mode, the first thing you're likely to notice is the smaller size characters. Since increasing the size of the screen is impossible, adding 40 more columns to the 64 makes it necessary to halve the size of each character. Some televisions may not produce a clear enough picture to make these smaller characters readable, so you may find it difficult to read text in 80 columns. We recommend using a video monitor with the color turned off. You may want to change the character set to suit your personal taste or needs. "Custom-80" (discussed later) is designed to let you do just that.

Screen-80 provides a different cursor than does the normal 40-column mode. Rather than a blinking block, it uses an underline character; but like the normal cursor, it can move anywhere on the screen. In fact, you can use all the cursor control keys, just as you would normally, to insert or delete, home the cursor, clear the screen, or create BASIC program lines.

Both uppercase/graphics and lower/uppercase modes are supported in Screen-80, but you cannot toggle between these modes with the SHIFT-Commodore key combination. Instead, you can put the screen editor in lower-/uppercase mode by pressing the CTRL and N keys simultaneously, or by printing CHR$(14). To return to uppercase/graphics mode, simply print CHR$(142) to the screen. These methods affect only characters printed after these commands. Thus, you may have both sets (for example, graphics and lowercase) on the screen at the same time for increased programming flexibility.

You can change the color of the background, text, or border by simply POKEing the appropriate color number into location 53281 (for the background), location 646 (for text), or location 53280 (for the border). Changing text color changes the color of all text on the screen. If you want to change the background or text color during program mode, print a CHR$(13) after POKEing the appropriate location. Since color memory is fixed on the 64, it's impossible to have true 80-column color. Therefore, Screen-80 does not recognize color codes in PRINT statements as being any different from other graphics characters. All printing to the screen uses the color specified in location 646.

Graphics And Sound Routines

Screen-80 can be used with sprites, high-resolution graphics, and sound—just like the normal 64 screen. Since this program actually uses a hi-res screen, you can also use it for other graphics displays. You can even have text and hi-res graphics on the screen at the same time. (Check your *Programmer's Reference Guide* for more detailed information on how to plot points on the hi-res screen.)

To plot points (or do anything else) to the hi-res screen, it is important to know how to POKE and PEEK to the screen. The hi-res screen for Screen-80 is located at 57344 ($E000). Since this screen memory shares addresses with ROM, you may POKE graphics safely to the screen, but attempting to PEEK from the screen will give you values from the ROMs. To PEEK these screen locations, you must first disable interrupts, then use location 1 to reconfigure the 64. Location 1 allows you to selectively replace ROM locations with RAM—making it possible to have a full 64K of RAM on the 64. Once you reconfigure the 64, you can PEEK locations on the screen. Since the operating system must access the ROMs to function properly, you must immediately restore the 64 to its normal configuration and reenable interrupts. Here's an example of how to do it in BASIC:

```
10 POKE 56334,PEEK(56334)AND254:
   REM DISABLE INTERRUPTS
20 POKE 1,53: REM RECONFIGURE 64
30 PRINT PEEK(57344): REM PEEK THE
   FIRST BYTE OF SCREEN RAM
40 POKE 1,55: REM RETURN 64 TO
   NORMAL
50 POKE 56334,PEEK(56334)OR1: REM
   REENABLE INTERRUPTS
```

Using sprites in Screen-80 requires all sprite data to be kept within the same 16K block as the hi-res screen. Locations 49152 ($C000) to 53247 ($CFFF) are perfect places to put sprite data. The sprite pointers for Screen-80 are located at 53248+2040 to 53248+2047. To cause sprite 0 to get its data from 49152 ($C000), put a zero into location 53248+2040. Since POKEs to this area of memory are normally intercepted by the I/O chip, we must disable interrupts and I/O to put a value into these locations. Here's a program to put a sprite onto the screen:

```
10 V = 53248
20 POKE V,100:POKE V+1,100
30 POKE V+39,2
40 POKE 56334,PEEK(56334)AND254
50 POKE 1,PEEK(1)AND251
60 POKE 53248+2040,0
70 POKE 1,PEEK(1)OR4
80 POKE 56334,PEEK(56334)OR1
90 POKE V+21,1
```

Creating sound from within Screen-80 is done exactly the same way as from the normal screen. In fact, since you want to POKE the information to the SID chip (in the I/O area), you don't have to disable interrupts or do any bank switching, as was necessary for hi-res graphics or sprites. The normal POKEs will do.

Using Other Programs With Screen-80

This program is designed to intercept any calls to the normal Kernal PRINT routine ($FFD2). Software which bypasses this routine or POKEs directly to the screen will not work correctly with Screen-80. An example of a program which bypasses the PRINT vector is the DOS wedge program (on the TEST/DEMO disk which comes with 1541 disk drives). Fortunately, this problem can easily be fixed by changing all PRINTs to pass through the standard vector. This program, when used in place of the normal DOS boot program ("C-64 Wedge"), changes these references.

```
10 IF A=0 THEN A=1:LOAD"DOS
   5.1",8,1
20 FOR I=1 TO 7:READ A:POKE A,210:
   POKE A+1,255:NEXT
30 DATA 52644,52650,52712,52726,
   52752,52765,53075
40 SYS 52224
```

With these changes, the DOS support program will work with Screen-80.

One of the best applications of Screen-80 is with terminal software. We tested Screen-80 with the terminal software that comes with the VICmodem and the 1650 Automodem by Commodore. Since both programs use the standard PRINT vector, they work fine. Programs which depend on sprites should be avoided, as should programs which move screen memory or otherwise change the normal configuration of the 64.

SpeedScript does not use the PRINT vector at $FFD2 to update the screen, which makes it incompatible with Screen-80.

How To Use Screen-80

To use Screen-80, LOAD the program from disk using the normal format for LOADing BASIC programs:

```
LOAD "filename",8
```

Notice that we LOADed our program without the ,1 that generally accompanies ML programs. If the program loaded correctly, you can LIST it. You should see one line with a SYS command. Simply RUN the program and you'll have Screen-80. Pressing RUN/STOP–RESTORE will put you into 40-column mode, and typing SYS710 and pressing RETURN will bring back 80 columns. You can make a backup copy by simply saving it as you would a BASIC program:

```
SAVE "filename",8
```

At this point, type NEW to begin programming in 80columns.

Custom-80: Creating Your Own Character Set

"Custom-80" allows you to create your own character set for use with Screen-80. It's easy to use and requires a joystick.

Custom-80 "borrows" the character set from Screen-80 and then moves it to a safe location in memory for editing. After editing, you can return the custom characters to the Screen-80 program, or save your new character set to disk or tape. To load Custom-80, type:

```
LOAD"CUSTOM-80",8,1
```

After loading it into memory, type NEW to reset the BASIC pointers. Next, LOAD Screen-80 into memory and SYS 49152. This puts you in Custom-80 and, at the same time, accesses the character set included with Screen-80.

The Screen-80 character set is displayed in the lower half of the screen, while the current character being edited is framed by a yellow cursor. In the upper-left corner of the screen the character is enlarged for editing, and brief instructions are provided to the right.

Customizing Characters

You can choose which character you want to edit by moving the cursor around the screen using the joystick or the cursor keys. The flashing blue square indicates the current pixel in the character you are editing. To set the pixel, press the fire button on the joystick. To reset the pixel, press the fire button again.

Press SHIFT and CLR/HOME to clear all the pixels in the character you're editing. (This will not affect the characters previously edited.) To home the cursor to the first character, press CLR/HOME without pressing SHIFT. To copy a character from one position to another, press f1 to store the current character into the buffer. Then move the cursor to the new position of the character and press f7 to retrieve it from the buffer.

Pressing S saves the character set to tape or disk as a short program file. It can then be loaded back into memory with the L command. When loading or saving, you will first be asked for the name of the file, then asked to press T for tape or D for Disk. If an error occurs during a disk operation, the program will return with the error message.

If you wish to make the new character set a permanent part of Screen-80, press X. This puts the redefined character set back into Screen-80 and exits to BASIC. You can then save the new version of Screen-80 to disk with the redefined characters already in the pro-

gram by entering:

```
SAVE"SCREEN-80",8
```

The next time you run Screen-80, you'll have your new character set in the program.

If you wish to use various character sets with Screen-80, you should save the character sets to tape or disk using the method just discussed and load the individual character sets by using the program "Custom Loader" while in Screen-80. This program LOADs the new character set into Screen-80 *after* it is activated. When the program prompts you for the name of the character set you want to load, enter the filename, comma, and the number of the device you want to LOAD the character set from. Use 8 for disk, 1 for tape.

One important note: You cannot SYS to Custom-80 from Screen-80. You must press RUN/STOP–RESTORE to leave Screen-80 before typing SYS 49152 to run Custom-80.

How It Works

First, Custom-80 performs a block memory move of the character set data from Screen-80 to 12288 ($3000). This is done to make it easier to display the character set at the bottom of the screen.

Next, a raster interrupt splits the screen to show both the redefined character set and the normal character set. The instructions and the enlarged character are printed on the top half of the screen. The enlarged character is a 4 × 8 matrix of reverse SHIFT-Os. Before entering the main loop, all variables are initialized.

The main loop has two major routines. The first routine checks the joystick and keyboard. If a key is pressed, the appropriate flag is set. Pressing X sends the program to the routine that moves the character set back into Screen-80. The S key saves a character set, while the L key loads a character set.

The second routine prints the enlarged character on the screen. If any flags were set, this routine handles them. It takes care of the save-to-buffer routine, the get-from-buffer routine, the clear-character routine, and the routine that handles the flashing of the blue cursor in the enlarged character. ©

Skyscape

Robert M. Simons

This unique program, written by a planetarium director, presents the sky as it can be viewed at any date and time from the year 1977 onward—including zodiac constellations and all the visible planets. It also calculates planet tables, positions of the sun, and phases of the moon for any date and time from 1977 into the future. "Skyscape" is both educational and entertaining.

For thousands of years the sun, moon, and planets in our solar system have excited human imagination. In ancient times they were regarded as gods whose distant motions influenced the course of earthly events. Though we now understand more about the true nature of celestial objects, many facts remain unknown, and a brilliant nighttime sky still presents an inspiring spectacle.

Whether you're seriously interested in the sky or just casually curious, "Skyscape" is a convenient tool for extending your knowledge. It opens a movable window on the heavens, displaying the position of our sun, moon, and neighboring planets from almost any location on Earth, at any point in time from 1977 into the distant future. Since it performs all the necessary calculations, you can enjoy and learn from this program even if you're not an expert in astronomy. In addition to providing data about the position of celestial objects, it draws a sky map on the screen, showing each object as it would appear to you at the chosen location and time.

Past, Present, Or Future

Skyscape begins by asking you to answer several questions. Enter the year, choosing any year from 1977 forward. In some ways this is the most important input of all, since objects in our solar system move significantly from one year to the next. After you choose the year, Skyscape allows you to enter the month and day.

Next you must enter the latitude (north/south position on Earth) from which you wish to view the sky. Latitude 0 places you, the observer, at the equator. Latitudes 1–90 place you in the northern hemisphere (north of the equator). To choose a southern latitude (south of the equator), enter a negative number from −1 to −90. Skyscape generally represents southerly locations with negative values.

Whenever Skyscape asks for information, it checks your entry to make sure it's in the acceptable range. If you enter an illegal value, the program displays an error message and gives you another chance.

The Sun And Moon

Though very different in size and composition, the sun and moon are alike in being the largest celestial objects visible from Earth. After you enter the date and latitude,

Skyscape displays a table of data for the sun and moon. In addition to the date, day of the year, and latitude north or south, you'll see the following information:

- Sun's geocentric angle. This figure represents the sun's position as a number of degrees relative to the vernal equinox. The vernal equinox is where the sun is located when spring begins in the northern hemisphere (the same time that autumn begins in the southern hemisphere).
- Sun's declination. The number of degrees north or south of the equator. Negative values indicate a southerly location.
- Sun's altitude at noon. The location of the sun in degrees from the northern or southern horizon at noon.
- Sun's right ascension. Just as longitude and latitude indicate locations on the Earth, *right ascension* and *declination* are used to pinpoint locations in the sky. For this purpose the sky is visualized as a gigantic sphere surrounding the Earth. Declination locates a point vertically in the celestial sphere and right ascension locates it horizontally. Right ascension values are given in *hours* and *minutes* in the range 0:00–23:59. Right ascension 0:00 is exactly at the vernal equinox. Larger right ascension values lie to the east of smaller ones.
- Right ascension at 9 p.m. The right ascension which would be on the meridian at 9 p.m. This coordinate system would be found on star charts. By comparing this number with those charts, you can tell what stars and constellations would be visible at that time.
- Moon's age. The number of days since the last new moon.
- Moon's elongation. The location of the moon in degrees east or west of the sun.
- Moon's phase. The phase of the moon on this particular day.

The Planet Table

After viewing the sun and moon display, press P to continue to the next display screen, which contains the planet table. (Press D if you wish to enter a new date.) The planet table shows vital information about the visible planets (through Uranus, which is at the limit of our visibility). The table shows the position of each planet in right ascension and degrees east or west of the sun. It also shows the distance of each planet from Earth in millions of miles.

If you'd rather see the distance in kilometers, modify the line 220 program to change the value of ES=93 to ES=149.6.

Some planets have an asterisk to the left of the right ascension figure. This signifies that they are visible at 9 o'clock this evening. For reference, the planet table also includes the sun's present right ascension and its right ascension at 9 p.m. Press D to input a new date or S to view a graphics display of the sky at any time in the current day.

The Visible Skyscape

After selecting the sky display, you must enter the hour when you wish to view the sky. The hour value should be a whole number from 0–23 (enter 22 for 10 p.m., etc.). You'll also need to enter the minutes (0–59). Skyscape then displays the time and offers you a chance to enter different values. Press RETURN or Enter when you're satisfied with the time.

Skyscape now displays the sky as it would appear at the chosen latitude, date, and time. Since the sky looks very different from different places on Earth, the latitude affects the display considerably. If your latitude is in the range 24–90 degrees north or south, the sky shows a dashed line representing the position of the celestial equator, along with symbols representing the sun, moon, and planets visible at that time. If your latitude is in the tropical region—from 23½ degrees north to 23½ degrees south—the dashed line indicates a position directly overhead.

If you're viewing in the northern hemisphere, north is above the dashed line and south is below it. In the southern hemisphere these directions are reversed. Below the sky display is a key that interprets the symbols used to represent celestial objects. If more than one object is positioned at the same spot, the symbols are displayed above each other.

At the bottom of the sky you may see two-letter abbreviations. These represent zodiac constellations that would be visible from your chosen vantage point. Skyscape uses the abbreviations AR (Aries), PI (Pisces), AQ (Aquarius), CP (Capricorn), SA (Sagittarius), SC (Scorpio), LI (Libra), VI (Virgo), LE (Leo), CA (Cancer), GE (Gemini) and TA (Taurus). Each constellation is located above the spot where its abbreviation appears. In northern latitudes, the border of each constellation's zone begins at its abbreviation and extends left. In southern latitudes, the constellation extends right from the position of its abbreviation.

Daytime skies are shown in blue and nighttime skies in black. Skyscape does not calculate the actual rising or setting time of the sun. Average rising and setting times of 6 a.m. and 6 p.m. are used in every case. You may obtain exact rising and setting times from local newspapers. However, keep in mind that there is usually about an hour of twilight before sunrise and after sunset.

Halley's Comet

In addition to permanent objects, Skyscape's graphics display includes Halley's Comet, visible during late 1985 and early 1986. If you choose a date from November 1, 1985 to May 29, 1986, Skyscape calculates the position of Halley's Comet and includes it in the graphics display (if it would be visible at the place and time you select). The comet's position is based on the best predictions available at the time of this writing (summer 1985). These positions may differ slightly from the comet's actual position when it finally makes its appearance.

While Skyscape is generally accurate, it bases most position calculations on circular orbits. This introduces a certain element of error, since no object in our solar system has a perfectly circular orbit. The position error is most pronounced for Mercury and Mars (whose orbits are quite elliptical), but does not significantly affect other objects. I've found Skyscape accurate enough for my own purposes, which include planning astronomy classes and planetarium displays. ©

Number Construction Kit

Gerald W. Rightmer

You've got the tools, materials, and a goal in this construction project. All you have to do is think hard and work fast. For children, an excellent program for practicing math skills; for adults, a real brain teaser.

Your bid has just been accepted for a new construction project—to build a three-digit number between 100 and 999. Fortunately you have your number construction kit, which includes everything you need for the task. Your tools are the four basic arithmetic operators: +, −, *, and /. Your building materials are single digit integers, three, four, or five of them, depending on the level of play you choose. And, as with all construction projects, there is a deadline: one, two, or three minutes, depending on your level of expertise.

Getting Started

To play, load the program and type RUN. First, you're asked for the number of players. Press 1 or 2. Then you're asked to choose one of three difficulty levels: 1. Beginner, 2. Intermediate, or 3. Advanced. Press the number key which corresponds to your choice. At the beginner level, you're given five numbers for materials and three minutes to complete the construction project. At the intermediate level, there are four numbers to build with and two minutes to finish. And, if you're up to it, the advanced level offers only one minute and three numbers to work with.

After selecting the number of players and a difficulty level, you're ready to begin. The project total (your goal) and materials (the numbers you're given to work with) are presented and you're prompted to press any key to begin. As soon as the game begins, the timer starts counting down. First select one of the numbers from the materials listed. Then choose one of the operators: the plus or minus sign (on the top row of the keyboard), the multiplication sign (on the second row beneath the minus sign), or the division sign (at the bottom row next to the right SHIFT key).

Begin building toward your project goal without hesitation. If you reach the project goal before time runs out, press RETURN. Each game consists of five rounds. In the two-player game, one round consists of one turn for each player. After five rounds, the game ends, and you're asked if you'd like to play again. At this point, you can change the number of players and the difficulty level.

Staying In Bounds

Like any construction project, there are constraints. First, there are only 15 spaces with which to work. This means that you can use only eight numbers and seven operators to complete the project. You're allowed to move across the computation line using the cursor keys, and change numbers or operators simply by typing over the existing characters. A running total is displayed to the right of the equation, allowing you to instantly monitor your progress. Since the target number is positive and greater than 100, the program does not allow numbers less than 0 in the target number. Also, numbers greater than 9,999 cannot be created during the construction process. Error messages will appear, and the game will pause if the running total is greater than 999 or less than 0.

Scoring is based on how close you are to the target, the difficulty level, and how quickly you respond. If your total at the point when time expires is greater than 200 from the proposed project total, then no score is awarded for the round. A round is terminated either by the clock or by pressing the RETURN key.

The program randomly generates numbers for the project goal and materials, excluding duplicates and zeros. The only keys that work are the designated materials (numbers) keys, the four operators, left and right cursor keys, DELete, and RETURN. The DELete key deletes only from the end of the line and removes only the character under the cursor. This means that no gaps can be created in the middle of a computation line (this would result in a meaningless arithmetic expression).

Number Construction Kit is a challenge to any age, and it can be an entertaining way to reinforce basic arithmetic skills in children. Since immediate feedback is given throughout the game, a child can learn much by trial and error.

For those simply looking to win the game, a few strategy tips might be helpful. The first step is to study the project number and the materials before pressing a key to start construction. For larger, more difficult numbers, it's usually easier to get within range of the project as fast as possible and to fine tune your answer with the remaining time and space. Remember, the materials may be used as often as necessary—you have an unlimited supply. ©

Home Financial Calculator

Patrick Parrish, Programming Supervisor

Many home budget programs have been published in magazines, but rarely has there been a program integrating as wide a variety of loan and investment calculations as "Home Financial Calculator." It is versatile, easy to use, and flexible. Rapid recalculation features make it an ideal tool for "what if" projections. A calculator mode with memory lets you solve problems not directly supported by the program, and you can pass values generated by one calculation to another.

Investment and loan calculations are readily computerized. In fact, many programs have been written which perform these tasks individually. "Home Financial Calculator" goes a step further by integrating several common financial calculations in a menu-driven package. It also features a calculator mode or scratch pad area where program variables can be manipulated using common mathematical operations.

When you run the program, a main menu offers you a choice of Investment or Loan calculations. Type I or L to reach the appropriate submenu.

Common Variables

Before looking at any calculations, let's consider some basics of the program. Home Financial Calculator uses some parameters or variables repeatedly in the calculations. These variables are *Total* (also referred to as Future Value, Total Owed, etc., depending on the calculation); *Present Value* (principal); *Interest Rate; Years; Months; Number of Periods* (of either compounding, deposits, withdrawals, or payments, depending on the application); *Deposits;* and *Withdrawals.* When in the calculator mode (explained below), you'll reference these eight variables with the single letters T, P, I, Y, M, N, D, and W.

As you work with Home Financial Calculator, the values of the eight variables are preserved until you change them. Whenever the program asks you for an input (for example, Interest), the current value of that variable is displayed (zero if no value has been entered yet). If you want to keep the current value, just press RETURN. Otherwise, enter the new value and press RETURN.

With this feature, Home Financial Calculator makes it easy for you to generate "what if" projections. Simply run the same calculation repeatedly, each time changing a previously entered value. Press RETURN to keep a value, and change only one or two values to see the effect on the final result.

You can also store the current value into the calculator mode's Memory Register or recall a value from the Memory Register. To see how all this works, let's take a look at some calculations possible with Home Financial Calculator.

Investment Calculations

Here is the Investment submenu that appears when you type I from the main menu:

1) Future Value with Periodic Interest
2) Future Value with Interest Compounded Continuously
3) Future Value with Regular Deposits
4) Future Value with Cash Flows
5) Withdrawal of Funds
6) Net Present Value
7) Calculator Mode
8) Return to Main Menu.

Determine which option you want and press the appropriate key.

Each option displays screen prompts which ask you to input several values. These values are stored in the eight variables mentioned above: *T* for Total (Future Value), *P* for Present Value (principal), *I* for Interest Rate, *Y* for Years, *M* for Months, *N* for Number of Periods, *D* for Deposits, and *W* for Withdrawals. Of course, not all calculations require you to enter all these values, while others may ask for additional information.

Most calculations can be solved for any *one* of the variables. To solve for a variable, enter an uppercase X at the corresponding input prompt. For example, you could enter values for everything

except the Interest Rate, typing X at the Interest Rate prompt. Home Financial Calculator then solves for the Interest Rate.

Remember, however, that the program can solve for only *one* variable during each calculation. If you enter an X at more than one prompt, the program does not have enough information to calculate an answer. Keep this in mind, because the program does not check for potential conflicts.

Future Value With Periodic Interest

Home Financial Calculator's options are fairly self-explanatory when you run the program, but let's try an example. We'll calculate the future value of an investment drawing periodic interest. This kind of investment could be a savings account, interest-bearing checking account, bonds, or a money market account. Choose this option by entering 1 at the Investment submenu.

After the screen clears, the program asks for the first input—Future Value, which appears with an asterisk (*). Below this is a zero (the current value of this variable in memory; all variables start out with a value of zero). Following this is an input prompt.

The asterisk preceding Future Value means that this is one of the variables you can solve for. (A variable *not* preceded by an asterisk means that variable *cannot* be solved for in that particular calculation, so X would be an illegal response.) If you'd like to calculate the Future Value, enter an X here, and answer all the other prompts with the appropriate values.

Let's calculate the future value of a $1,000 investment drawing 8 percent interest for two years and three months, with four compounding periods each year. Enter an X for *Future Value*, since we'll be solving for this total. Answer *Present Value* with 1000 (the principal you're investing); *Annual Int Rate (%)* with 8 (enter the percentage, not a fraction); *For # Of Years* with 2; *For # Of Months* with 3; and *# Of Periods (Compounding)* with 4. After you enter the last value, Home Financial Calculator figures the *Total Future Value* and displays the answer—$1195.09.

Now suppose you wish to know the future value of the same $1,000 investment if you make 9 percent interest. Choose option 1 on the Investment submenu again and rerun the calculation. Notice how Home Financial Calculator automatically prints the current value of each variable at each prompt. The *Future Value* prompt shows a current value of 1195.09 from the previous calculation. Type an X at this prompt, 9 for Interest Rate, and RETURN at all other prompts to preserve their values. The result should be $1221.71.

The versatility of Home Financial Calculator becomes apparent when you realize how many different ways you can run this calculation. Using this same menu option, you can calculate the initial investment (or present value) necessary to accrue a certain future value with periodic interest; the interest rate necessary to accrue a future value from a present value; or the time (in years and months) it would take to accumulate a future amount from an initial investment with periodic interest payments. Just enter an X for the unknown value you're seeking, and fill in all the other prompts.

Future Value With Interest Compounded Continuously

Option 2, a variation of option 1, handles investments paying a continuous interest rate. Like option 1, option 2 can handle a number of calculations—just place an X in the slot you'd like to solve for.

Here, after entering all other parameters, you can calculate the future value of an investment; the initial investment required to reach a certain future value; the interest required to reach a desired future value; or the time required to reach a certain future value at a specified interest rate.

Notice that any variables used in option 1 will be displayed with their current values when running option 2. As mentioned above, the eight major variables in Home Financial Calculator retain their values throughout the program until you change them. This feature is convenient when going from one option to another on the Investment or Loan submenus.

In addition, the values are preserved for use in the calculator mode. For instance, you could compare the effect of continuously compounded interest to periodic interest (option 1) without having to retype the input.

Future Value With Regular Deposits

If you're interested in setting up an annuity, you'd choose option 3 on the Investment submenu. You can determine the future value of an account (such as a savings account, Individual Retirement Account, college or vacation fund, etc.) with regular deposits where interest is compounded with each deposit.

Option 3 can also tell you the amount of each deposit necessary to accrue a future value; the interest rate needed to provide some future value with regular deposits; or the time it would take to amass a future value with regular deposits.

Future Value With Cash Flows

Option 4 does a single calculation—it always solves for *Future Value*, so don't enter an X anywhere. It calculates the future value of an investment with yearly cash flows (either positive or negative). The *Annual Interest Rate* you input here is the growth rate on the money you've invested.

As an example, suppose you wish to determine the value of a vacation fund collected over four years. You're asked for the number of years, then for the deposit or withdrawal each year. You deposit $500 in the fund the first year and $200 the second. The third year you are forced to withdraw $300 (entered as −300), and the fourth year, you put in $400. The fund has a growth rate of 12 percent. Its value after four years will be $1,017.34.

A future value determination can also tell you whether an investment is worthwhile. If the future value of all cash flows is positive or zero, the investment is profitable. A negative future value, on the other hand, represents a losing investment.

Withdrawal Of Funds

If you intend to open an account from which you can regularly withdraw funds, choose option 5. With

this option, you can determine the initial deposit required in the account to cover your withdrawals; the amount you can withdraw regularly from this account; the rate of interest you must make on funds in the account; or the period of time over which you can make withdrawals.

Net Present Value

Option 6 lets you determine the feasibility of a prospective investment by calculating its net present value. Net present value is the current value of all future yearly cash flows to an investment along with any initial cash requirement. The interest rate you input here is the rate of return you require on your investment. A positive net present value indicates a profitable investment, while a negative result signifies a losing investment.

As an example, suppose you have the opportunity to make a $2,000 investment which would return $1,500 the first year, cost you $750 the second year, and return $1,900 the third year. You hope to make 13 percent on your money. With option 6, you determine a net present value of $56.87, representing a profitable investment.

The Calculator Mode

Option 7 puts you in the calculator mode (also available from the Loan submenu). Calculator mode works very much like a handheld calculator with a single memory. You can type in a value or recall one from a variable by entering its symbol—T(otal), P(resent Value), I(nterest Rate), Y(ears), M(onths), N(umber of Periods), D(eposits), and W(ithdrawals). You can perform simple math on values stored in the Memory Register using reverse Polish notation. And you can use the results in future calculations.

When you enter calculator mode, the calculator command line appears on the screen:

V S H R M+ M− M* M/ MR MC MEM=0

Here are the commands:

V	(View the values of the eight primary variables)
S	(Store Memory Register into a variable)
H	(Help—prints the command line)
R	(Return to main menu, exit calculator mode)
M+	(Add the last input to the Memory Register)
M−	(Subtract the last input from the value in the Memory Register, and store the result in the Register)
M*	(Multiply the last input times the value in the Memory Register, and store the result in the Register)
M/	(Divide the last input into the value in the Memory Register, and store the result in the Register)
MR	(Memory Recall)
MC	(Memory Clear to zero)
MEM=	(Memory Register's current value)

If you've run through a sample investment calculation, you now have some variables in memory. Enter V in the calculator mode to see them. The screen displays the eight values currently in memory for the eight variables.

To work with one of these variables, enter one of their letters (T, P, I, Y, M, N, D, or W) and press RETURN. Then type M+ to add it to the Memory Register (all variables must be stored in the Register before you can perform any operations on them). Suppose you put the current value for T into the Register and now wish to add $229 to this value. Enter 229, press RETURN, then type M+ and press RETURN. The addition is performed and the result displayed. To store this value back into the T variable, enter S for Store. A prompt appears, requesting the variable in which you intend to store the value. Type T to store the value into the variable T.

You can also use the Memory Register to hold a value not represented by any of the eight variables. To do this, determine a value using the calculator mode and store it into the Memory Register with M+. Then, when you're running a calculation elsewhere in the program, you can substitute this value for any of the eight primary variables by typing MR (Memory Recall) at the appropriate prompt. MR can be used both in the calculator mode and at any prompt where the previous value is displayed.

Finally, option 8 on the Investment submenu returns you to the main menu. Once there, you can perform some loan calculations by typing L.

Loan Calculations

Here is the Loan calculations submenu:

1) Regular Loan Payments
2) Remaining Loan Liability
3) Final Loan Payment
4) Single Payment Loan
5) Loan Amortization Schedule
6) Calculator Mode
7) Return to Main Menu

Regular Loan Payments

Option 1 handles a number of calculations for equal payment loans. You can figure the principal of a loan; the amount of each regular payment necessary to repay a loan; the annual interest rate on a loan with regular payments; or the term of the loan.

Remaining Loan Liability

With option 2, you can determine the remaining balance on a loan with regular payments after a number of payments have been made. Enter the principal on the loan, the amount of each payment, the annual interest rate, the number of payments yearly, and the last payment number.

Final Loan Payment

Option 3 calculates the amount of the final payment on a loan. In many cases, the last payment of a loan will vary from the amount of the regular payment. This option handles situations where the final payment is greater than ("balloon payments") or less than the regular payment.

Single Payment Loan

Option 4 calculates the amount owed on a loan that is paid off with a single payment. You must input the principal on the loan, its annual interest rate, its term in years and months, and the number of times a year the interest on the principal is compounded.

Loan Amortization Schedule

Option 5 displays a loan amortization schedule. Enter the principal on the loan, the amount of each payment, the annual interest rate, the term of the loan, and the number of payments yearly. Then enter the period of the year in which the loan began (for instance, 10 for October) and the range in years of the

amortization schedule you'd like to examine.

Because of the complexity of these calculations, there may be a delay before the output appears on the screen, especially if you have chosen to look at the latter years in a long-term loan repayment schedule (such as a home mortgage). When the amortization table appears, it displays the payment number, the beginning balance for the period, the amount paid toward the loan principal, the amount paid in interest, and the ending balance. To keep the information from scrolling off the screen, the program shows only a few payment periods at a time. Press RETURN to view another screenful. When the end of a year is reached, the program gives the total amounts paid on the principal and in interest for the year. In addition, when the last period of the loan is reached, the program displays the final payment for the loan.

The last two options on the Loan submenu are the same as those on the Investment submenu.

Modifying The Program

Home Financial Calculator is written in a modular format for easy modification. For many routines, it uses common input labels (lines 4710–5080) and some output labels (lines 5090–5170). If you want to add an investment or loan calculation routine, choose the labels from these lines that fit your application.

Also, you may wish to add a printer option to the loan amortization schedule. Examine lines 3230–3940. Here, variable D5 (defined in line 150) determines the number of loan payments considered on each screen. Variables S1, S2, S3, and S4 (defined in lines 160–190) format the output horizontally on the screen. ©

Budget Planner

Gregory L. Smith

Managing your budget doesn't have to be a chore with this program. It totals your income and bills week by week, allowing you to plan your expenses in advance and make adjustments for special occasions like vacations or birthdays.

Have you ever wondered if you can afford additional monthly expenses—like a mortgage or additional credit cards? Or how much "discretionary income" (money left after paying bills) you might have around Christmastime? I used to manage all the details in my head, but usually had little or no idea as to how much money there actually was (or where it all went).

"Budget Planner" is a mini-spreadsheet written in BASIC. It's designed specifically to handle weekly and monthly income/expenses for an entire year. Budget Planner will sum a column (a week of income and expenses) and present a total. The total is then carried over into the next week as a previous balance.

Commodore 128 owners should modify the program by changing line 90 as follows.

```
90 FOR I=1 TO 8: READ A: KEYI,
   CHR$(A): NEXT: DATA 133,137,134,
   138,135,139,136,140
```

When you first run it, Budget Planner asks for a filename of the information to be loaded. Once you've set up your yearly finances, you can save the data to disk or tape. Since you're just getting started, simply press RETURN and you'll be prompted for the date. This is the date you want to start budgeting. Enter the day as a number, the month as a three-character abbreviation, and the year as a four-digit value. You're not required to use the beginning of the year. You may start in the middle or at the end if you like. The program is even smart enough to handle leap years.

The screen then displays four columns of information. The first column holds the labels for each row. The only two rows available when you first start are the "PREVBAL" (previous balance) and "TOTAL."

The other three columns are the budget data. The week number is shown on the left side. The date of each week is above the corresponding column. When you use the left/right cursor keys to scroll around the budget, the week number will change.

You enter income and expenses by typing the amount (which appears on line 2 of the screen) and pressing RETURN. Only numbers and the "+", "−", and "." keys will work here. Use the delete key to remove the last character typed.

The arrow keys are used to move the reverse-video cursor around the screen. You cannot alter the TOTAL row since that is controlled by the computer. You can see the next week by scrolling off

the edge of the screen. This is allowed in both directions. The up/down arrow keys work after more expenses have been added. However, you cannot scroll vertically as only 18 rows (one screenful) are allowed.

The Function Keys

The function keys trigger the various options. Pressing f1 causes the help menu to appear. It draws a window in the top left corner of the screen with a reminder as to which key does what.

To go to a specific week, press f2. You may jump to any week you like by entering its week number (week 1 is the start of the budget and week 52 is the end.) You can also do a "relative jump" by using a plus or minus sign (−3 means go back 3 weeks and +3 means go forward 3 weeks). This is faster than pressing the right-arrow key many times. Also, SHIFT-E will jump to the end of the next month. This is handy since most of us pay bills at the end of the month.

F3 is the LOAD key. You may load a new budget (data from the disk) at any time. You'll be asked to supply a filename. If you pressed f3 accidentally, just press RETURN in response to the filename prompt. You'll be returned to your original budget. Likewise, f4 (SHIFT-f3) is the SAVE key. You might want to use the year of the budget as the filename. Again, if you decide not to save the budget, simply press RETURN.

To add a new row, use f5 (labeled +ROW in the help menu). It inserts a row below the cursor for new expenses (or new income). You'll be prompted for a name, which can be up to ten characters long. As with the other options, pressing RETURN cancels the operation. Once a name has been assigned, you must specify whether it is a source of income (I) or an expense (E). Pressing any key other than "I" defaults to an expense. This is important: Every value you enter in this field will be negative if it is an expense and positive if it is income. (You don't need to place a minus sign in front of an expense). Likewise, f6 deletes the row the cursor is on. You must verify by entering YES (or the letter Y) when asked. If you enter anything else, you'll be returned to the current budget.

The f7 key causes a new balance to be computed. Balances are *not* recalculated after each new amount is entered. Since the program is written in BASIC, it takes several seconds to recompute the array and display the results. This delay would be annoying if the program made all the computations every time you added another number.

Pressing f8 makes the results print out. (Be sure your printer is turned on.) You may specify a range of months to print. Pressing RETURN in response to the FROM? prompt returns you to the budget, without printing. Enter a valid 3-character abbreviation (JAN or AUG, for example). When prompted to enter TO?, you may press RETURN to print only one month. To print several months, use the appropriate three-character abbreviation.

Special Keys

There are five *SHIFT* functions in addition to the function keys. One has already been mentioned, the SHIFT-E key, which jumps to the end of the next month.

SHIFT-D switches between showing and not showing the decimals (pennies). Budget Planner defaults to showing the decimals. (The variable DP is initialized in line 1950. Simply change it to DP=0 to change the default.) When decimals aren't being displayed, you see the rounded amount on-screen.

SHIFT-F finds the next negative value in the TOTAL row. If you ever "go negative," the TOTAL value and the offending value will turn red. In such a case, you don't have enough income to cover expenses and it may be necessary to juggle some bills. SHIFT-F saves you from trying to find these trouble spots manually.

Use SHIFT-M to flag monthly expenses. If your rent is $500, for example, you would type in 500 in that row and then, instead of RETURN, press SHIFT-M. The $500 expense will be added to the end of every month. Note that this value is placed only on the same row as the cursor. Also, only the months AFTER the cursor position are affected. (For example, if you are on week 26, only weeks 26–52 are affected. The other 25 weeks will not be changed.) Naturally, this key works for income as well.

SHIFT-W is the weekly expense key (groceries, for example). Pressing it places the line 2 value in each week after the cursor position. Again, you use it in place of RETURN after entering a value. It works much like the monthly expense key and also works for the income fields.

Press SHIFT-T if you use a Datassette. It directs data storage and retrieval (f3 and f4) to the tape drive.

The SHIFT-Q key is the Disk Select key. It causes data to be saved to and loaded from a disk drive. (Note: Disk is the default. Changing DEV=8 to DEV=1 in line 100 will change the default device to tape.) ©

Cataloger

Kevin Mykytyn

This program helps you organize your disk library by making it easy to find any program on any disk. Included are options to print out a master directory and alphabetically sort all filenames.

After owning a computer and disk drive for a while, it doesn't take long before you find yourself inundated by programs and disks. No matter how well your disks are organized, you may still find yourself loading several directories searching for that one program. Cataloger offers a practical solution. It's a straightforward, menu-based program that creates a master directory for all your disk-based programs. Several extra features make it especially useful.

Managing Hundreds Of Files

Cataloger is written entirely in BASIC and runs on the 128 (either 64 or 128 mode), 64, Plus/4, and Commodore 16. As listed, Cataloger can handle up to 800 filenames. If you're using a 128 in 128 mode, it can handle up to 2000 filenames. To change the number of filenames for either of the 128, replace the 800 in lines 10, 370, and 920 with the 2000.)

To use the program, type RUN. A menu of nine choices is displayed:

1. CATALOG A SET OF DISKS
2. SEE ALL FILE NAMES
3. PRINT ALL FILE NAMES
4. SORT NAMES ALPHABETICALLY
5. CREATE A SEQUENTIAL FILE
6. CREATE A PROGRAM FILE
7. LOAD AN OLD FILE
8. DELETE A DISK
9. START NEW FILE

If you're using the program for the first time, choose option 1, Catalog A Set Of Disks. You'll then be asked for a disk name up to 16 characters long. This should be the name on the label of the disk you wish to catalog. Put the disk in the drive and press RETURN. The directory of the disk is displayed on the screen and the filenames are entered into the master directory. Next, you're asked whether you want to continue or quit. If you have more disks to catalog, press any key. Otherwise, type Q to go back to the main menu.

After entering all the disks you wish to catalog, you can view the master directory. Option 2, See All File Names, displays the master directory on the screen. You can also print it out on any Commodore printer using option 3. Make sure the printer is turned on before you choose this option.

The filenames are stored in a format slightly different from a standard disk directory. The filename is followed by the number of blocks used then the disk name.

Once you've viewed the directory, use option 5, Create A Sequential File, to save it. Choose this option if you wish to make changes later to the master directory. (The READ command of "MetaBASIC" or "BASIC Aid" allows you to view the master directory in this format.)

Option 6 also saves your master directory to disk, but instead of creating a sequential file, it creates a program beginning at line 100. Whenever you wish to view the master directory, you can load the program created. (With the FIND command of "MetaBASIC" or "BASIC Aid," you can search the directory for a specific program.)

Adding, Deleting, And Sorting

At some point you'll want to add or delete disks from your master directory. Options 7 and 8 are used for this purpose. Option 7 is used to load any directory stored in sequential file format (with option 5). After loading a file, you can choose option 1 and all new filenames will be appended to the old directory. Make sure you save your changes to disk when you're through.

To delete a disk from the master directory, choose option 8. In order for this option to work correctly, the directory must be *unsorted*. If you wish to have a sorted master directory on disk, you should first save it unsorted.

The delete option removes all filenames from the disk name specified. Therefore it's important that all disks have a unique name. If you've made changes to a disk and wish to enter the changes in the master directory, use option 8 to delete that disk, then use option 1 to enter the newer disk version.

You'll find it easier to locate a specific file if the master directory has been sorted. Once the file has been saved in sequential file format, use option 4 to sort the names.

To delete the directory in memory and start a new directory, use option 9. ©

Mini-Filer

Kevin Martin

This flexible file manager for the Commodore 64 and 128 (in 64 mode) is fast and easy to use. Single key commands allow you to create, add, edit, search, load, save, and print out. A disk drive is required.

Databases are powerful tools for many applications. They can manage any information you need to store and retrieve. "Mini-Filer," while not a full-featured database, allows you to easily create and manage any number of files by providing various single-key commands for creating, adding, editing, searching, and printing out.

Because Mini-Filer is a general-purpose file manager, it can be used for many applications. It's not designed to fulfill a specific purpose, so it's general enough to handle many different types of files. For example, you could create a name and address file (name, street, city, state, zip code, and phone number), a file for your garden (plant name, gestation, watering/feeding needs, and harvest time), and a file for your library (title, author, publication date, publisher, and subject).

Each grouping of information within a file is a *record*. For example, "John Doe, 123 Main Street, Anywhere, USA 11111, 888-999-9999)" could be one record within an address file. When you enter a new name, address, and so on, you're adding another record. In Mini-Filer, there's no restriction to the number of records (outside of the normal limitations of computer memory and disk space), but exercise common sense when creating a file. Remember, the longer your file, the longer the search time to find a record. And if the record size is fairly small, you'll be able to fit more records into available memory.

Finally, each record contains one or more *fields*. In the examples above, name, street, and city are fields within the address file, and author and publisher are fields in the library file. The fields can be *alphanumeric*, meaning any characters are allowed, or *numeric*, containing numbers only.

Think of the structure as a recipe card box. The entire collection of cards is the file. Each card is a record inside the file. And each line on the card is a field.

If you see how all this information is organized, you can see how useful a file manager like Mini-Filer can be. There are a multitude of applications. Aside from those noted above, you can create a file for home inventory, collections (cards, coins, stamps, record albums, and so on), gifts, recipes, Christmas cards, and mailing labels, to name a few.

A Few Steps

Using Mini-Filer requires three basic steps: create a file structure, enter records, and save these records to disk for later use. Mini-Filer also allows you to print out a single record or an entire file at the stroke of a single key. A printer is not required for use with the program, however.

To load and run Mini-Filer, type LOAD"MINI-FILER",8 and RUN.

Creating A File

After running Mini-Filer, you'll see a gray screen with a command line at the top which includes a series of letters, each of which represents a command: A (add or enter), C (create), F (find), P (print a record), R (print a file), f7 (load a file from disk), f8 (save a file to disk), CLR (clear data from memory), up (cursor up), and down (cursor down). For a complete list with more detailed explanations, see the Mini-Filer Commands chart elsewhere in this article.

To get started, press C to create a file. You'll notice that the command line has changed to read "CREATE: Enter record format, f1 to exit." (If you have a file already in memory, this option is not allowed.) You'll notice the cursor at the top left corner of the screen, beneath the command line. Think of the screen as a blank file card on which you can write anywhere. Using the cursor keys, move the cursor to where you wish to begin. Any location on the screen is acceptable, but note that each record is restricted to one screen of information.

First type a label that describes the information in the field that will follow. Determine the maximum number of characters you'll need (up to 128 characters per field). For example, in creating a name and address file, you might type "Name:" and decide that 18 characters would be enough for even the longest name you'd enter. Next, press SHIFT and the space bar to allocate the number of spaces for the field length. In our example, you would hold down the SHIFT key and press the space bar 18 times.

If you've made any typing errors, use the INSerT/DELete key to edit. Press RETURN when you've finished defining the length of a field. (In our example, press RE-

TURN after the eighteenth space has been entered.) The cursor will then jump to the beginning of the line below. Next, move the cursor to where you want to type the name of the next field label. Repeat the procedure just described until you've completed a record.

If you plan to use a numeric field (like a phone number or zip code), press the Commodore key and the space bar rather than a shifted space. (A numeric field appears as reverse periods; an alphanumeric field—letters and/or numbers—appears as reverse spaces.)

The command line at the top of the screen reminds you to press f1 to exit the format creation mode. Now you're ready to enter data. The original command line will reappear. Press A (for adding or editing a record). This puts you in Add mode. You'll see "ADD Record" appear in the command line, and the cursor will immediately go to the first space in the first field you've defined.

Now begin entering information. After typing in the data for one field, press RETURN. The cursor will automatically jump to the first space in the next field. If you reach the last available space in the field and type a character, the cursor will jump to the first space in the next field. The INSerT/DELete key may be used in case of typing errors. If you've defined a field as numeric only (appearing as reverse periods), it will accept only numeric characters.

Continue entering information until all fields in the record have been entered. Pressing RETURN when you're at the last character of the last field will return the cursor to the first character of the first field. At this point, you should proofread the record just entered. If you've discovered a typing error, press RETURN until you're at the beginning of the field that contains an error and move to the unwanted character with the cursor-right key. After making the correction, press RETURN.

To proceed to the next record, press f7. You'll see the formatted fields with the cursor in the first position of the first field. After completing the record, you can proceed to enter another by pressing f7, or you may exit the Add mode by pressing f1. The latter choice will return the original command line. At this point, you can move through all records by pressing the cursor-up key to read the previous record, or the cursor-down key to move to the next record. When the screen shows the format field—the fields with no entries—you're at the last position in the file. At this point, if you wish to enter a new record, press A and type in new data. If you wish to edit a record, press A while the record you wish to edit is on the screen.

Searching For Data

To search for any information, press F (for Find). You're then prompted to enter the characters you wish to search for. After entering them, press RETURN. Mini-Filer will instantly find the requested information and display the record which contains it. The original command line is then returned. To search for any subsequent occurrences, press SHIFT-F. The search will continue from the next record, and so on.

If the information requested is not in the file, the command line will display "Not Found." (When searching, be sure to enter the word you're looking for with the exact spelling, including capital and lowercase letters—"Spanish" will not be found if you enter "spanish.") Also, Mini-Filer will find a string of data within a field. For example, if you're searching for an area code—which is a portion of the phone number field—you would type the three-digit code and Mini-Filer will find it.

Saving And Loading

From the main command line, you can load or save a file at any time. To save a file, press f8. You'll be prompted for a filename. After typing one in, press RETURN and the file will be saved to disk as a program file with the name you specified.

To load a file from disk, press f7. You'll be prompted for a filename. After entering one, press RETURN and Mini-Filer will load the file from disk. After the file has loaded, the first record in the file will be displayed. Press any key to get the main command line.

Should you mistakenly press f7 or f8 and get a LOAD or SAVE prompt, press RETURN to get back to the main command line.

Printing Options

To print the current record, the one displayed, press P. This prints *only* the current record. To print the entire file, press R (for Report). The printed characters will appear just as they do on the screen.

Erasing A File

At some point, you may wish to erase the data in a file from memory and start over. To do this, press CLR (SHIFT–CLR/HOME). You're asked "Are You Sure?" Press Y if you wish to erase it or N if you'd rather reconsider. CLR is also useful if you plan to work with more than one file in a single sitting. You create the first, save to disk, and then press CLR. You can now load (or create) the second file.

Mini-Filer Commands

A	add or edit data
C	create a database file
F	find a field (any string of characters)
SHIFT-F	continue searching for next occurrence (may be used only after F)
P	print the record displayed
R	print the file
f7	add a record (in Add mode); load a file from disk (from main command line)
f8	save a file to disk
CLR	clear all file data from memory
up (cursor key)	move the cursor up (from Create mode) or move to the previous record (from main command line)
down (cursor key)	move the cursor down (from Create mode) or move to the next record (from main command line)

MetaBASIC 64

Kevin Mykytyn

This utility will change the way you program. It adds 32 new debugging and testing commands to Commodore 64 BASIC, working by itself or in conjunction with a machine language monitor/assembler.

An Introduction To MetaBASIC 64

"MetaBASIC 64" commands use English mnemonics, so you don't have to memorize a lot of SYS numbers. And if you forget the new words, you can either refer back to this article or type HELP.

BASIC programmers have 12 new commands at their fingertips. For writing programs, AUTO, KEY, and UNNEW are available. You can use CHANGE, DELETE, FIND, RENUM, and VCHANGE to examine and alter programs. And DUMP, SPEED, TRACE, and TROFF help during debugging sessions. If you're writing in machine language, you can use some of the BASIC problem solvers, as well as MEMORY, MONITOR, NUMBER, and @. To control MetaBASIC 64, you have DEFAULT, HELP, INT, and QUIT. Disk commands include BSAVE, CAT, DLIST, ERR, MERGE, READ, RESAVE, SCRATCH, SEND, and START. Finally, there's LLIST if you have a printer, and TERMINAL if you have a modem.

Special Notes

Always type NEW after loading MetaBASIC 64. One feature that works automatically is LIST Pause. When you're listing a program, hold down CTRL, SHIFT, or the Commodore key to temporarily halt it. RUN/STOP-RESTORE is available in both program mode and direct mode. But if you want to interrupt any of the utilities like RENUM, use the RUN/STOP key by itself (not RUN/STOP-RESTORE).

The commands work only in direct mode; you cannot add them to programs. Also, you're limited to one MetaBASIC command per line (although you can still use multistatement lines inside your programs). Unlike ordinary BASIC commands, there are no abbreviations. You must type out the entire MetaBASIC 64 command. If it seems to be working incorrectly, make sure the syntax is correct.

Machine language (ML) programmers should remember that MetaBASIC 64 occupies memory locations $9000–$9FFF. The 4K which begins at $C000 is available for programs like *Micromon* or for your own ML programs. Be sure to load and run MetaBASIC 64 *before* loading any other programs.

Using MetaBASIC 64

To use MetaBASIC 64, follow these steps:

1. Load the program with the command LOAD"METABASIC 64",8,1.
2. Type NEW
3. Activate the program with SYS 36864 (or SYS 9*4096)

The program uses 4K at the top of BASIC memory (which leaves you with 35K for your programs). The first thing it does is move the top-of-BASIC pointer down to protect itself from variables. After the SYS, it may seem that nothing has changed. But MetaBASIC 64 is active, and you now have 32 new commands to help you write and debug programs.

MetaBASIC 64 Commands

Here's an alphabetical list of the new commands and how to use them, with examples. In the descriptions of syntax, MetaBASIC 64 commands and mandatory parameters appear in boldface. String parameters appear in *italics*. Optional parameters appear in normal printing.

If something is described as a disk command, it won't work unless you have a disk drive. However, some of the ML programming aids can be useful in BASIC and vice versa.

@

Use: ML programming (see also MEMORY)
Syntax: **@ starting address, number,** number

This works like POKE, except it allows you to put a series of numbers into consecutive memory locations. For example, if you want to change border and background colors to white, you would use @53280,1,1. The first 1 goes into 53280, the second into 53281. If you add more numbers, separated by commas, they are POKEd into the next locations: 53282, 53283, and so on.

You can also use this in conjunction with MEMORY. First, display the contents of a series of locations using MEMORY. Then change the infor-

mation there by putting @ before each line you want to change. Cursor over to the numbers you want to change, change them, and press RETURN.

AUTO
Use: BASIC programming
Syntax: **AUTO** starting line number, increment

AUTO can take some of the drudgery out of writing a program. It automatically numbers a program, starting at the first number and incrementing by the second. Separate the numbers with a comma. If you do not specify a starting line number or increment, numbering will start at 5 and increment by 5 for each additional line. If you specify only a starting line number, then that value will also be used for the increment. After you press RETURN over a line, the next number is automatically printed. The current line number can be changed by using the INST/DEL (delete) key and replacing it with another number.

Press RUN/STOP to escape from AUTO.

Example: **AUTO 100,10** starts at 100 and numbers by 10.

BSAVE
Use: disk command (see also RESAVE)
Syntax: **BSAVE *"filename"*, starting address, ending address + 1**

BSAVE (Binary SAVE) saves a chunk of memory to disk, from the starting address to the ending address. Put the program name inside quotation marks and use commas to separate the name, starting address, and ending address. It's important that you add one to the actual ending address. You can use this command to make backups of machine language programs, as long as you know the starting and ending addresses. BSAVE can also function to save sections of screen memory, custom character sets, or high-resolution screens.

The numbers should be in decimal. If you need to translate from hexadecimal to decimal, see NUMBER (below).

After you BSAVE the contents of an area of memory to disk, you can load the data back in with LOAD *"filename"*,8,1.

Example: **BSAVE"METABASIC 64",36864,40805** makes a backup of MetaBASIC 64. To copy the first five lines of screen memory (locations 1024–1223) to disk, **BSAVE "SCREEN",1024,1224**. Screen memory does not include color information—that is stored in color memory and would have to be handled separately.

CAT
Use: disk command (see also DLIST, READ)
Syntax: **CAT**

Anytime you want to look at the entire disk directory, use CAT (for CATalog). The BASIC program currently in memory will remain undisturbed. To see specific portions of the directory, see DLIST.

CHANGE
Use: BASIC programming (see also FIND, VCHANGE)
Syntax: **CHANGE @*old string*@*new string*@**, starting line, ending line

CHANGE @*"old string"*@*"new string"*@, starting line, ending line

CHANGE searches through the program in memory, changing every occurrence of the old string to the new one. The strings can be up to 30 characters long, and must be bracketed by the commercial at sign (@). All lines in which changes are made are listed to the screen.

The first format will change BASIC keywords and variable names. The second format should be used to change strings. If you omit the line numbers, CHANGE affects the whole program. If you want to change only one section, add the starting and ending line numbers, marked off by commas.

Example: **CHANGE @X@QQ@, 1,200** changes the variable X to QQ in lines 1–200. To change the name Charles to John throughout the program, **CHANGE @"CHARLES"@ "JOHN"@**.

DEFAULT
Use: MetaBASIC 64 command (see also INT, QUIT)
Syntax: **DEFAULT** border color, background color, text color, device number

When you press RUN/STOP–RESTORE, the screen always reverts to the default colors of light blue characters on a dark blue screen. And several commands like LOAD and SAVE default to tape. DEFAULT lets you change these values to whatever you prefer.

If you have a disk drive, you can change the device number to 8. If you have a second drive addressed as device 9 that you want to use for SAVEs, change the default to 9. If your 64 is hooked up to a black-and-white TV, change the character and background colors to a more readable combination.

Note: If you change the default device number to 1 (tape), you will be unable to use any of the new MetaBASIC disk commands. To disable the DEFAULT device number setting and go back to normal, use the MONITOR command below. Also, the TERMINAL command will not operate properly after DEFAULT has been used to change the device number. If you use DEFAULT, be sure to issue a MONITOR command before trying to use the TERMINAL command.

Example: **DEFAULT 1,1,0,8** changes the border and background colors to white, the character color to black, and the device number to 8. If you press RUN/STOP–RESTORE, you'll see black characters on a white background. And you'll be able to save to disk by typing just SAVE*"filename"* (without adding a ,8).

DELETE
Use: BASIC programming
Syntax: **DELETE starting line–ending line**

DELETE removes a range of lines from your program. Separate the starting line number from the ending number with a dash (–). Use this command with extreme caution, since it is impossible to recover deleted program lines.

Example: **DELETE 200–250** erases all lines with line numbers in the range 200–250, including lines 200 and 250.

DLIST
Use: disk command (see also CAT, READ)
Syntax: **DLIST *"filename"***

This command lists a BASIC program from disk to the screen, without affecting what's currently in memory. The program name must be enclosed in quotation marks. DLIST enables you to look at a program before using MERGE or SCRATCH.

It also allows you to read portions of the directory. DLIST "$0:A*" displays all disk files beginning with the letter A.

Example: **DLIST "BASICPROGRAM"** reads the file named BASICPROGRAM from disk and lists it to the screen.

DUMP
Use: BASIC programming
Syntax: **DUMP**

Use DUMP to examine the current values of all nonarray variables in a program. If the program is running, press RUN/STOP and type DUMP. To resume, type CONT.

ERR
Use: disk command
Syntax: **ERR**

ERR reads the disk drive error channel and displays the DOS error number and error message from the drive. Use it when the red light on the disk drive starts blinking to determine what caused the problem.

FIND
Use: BASIC programming (see also CHANGE, VCHANGE)

Syntax: **FIND @*string*@**, starting line, ending line

FIND @*"string"*@, starting line, ending line

This allows you to find any word, variable, or other string within a program. Each line containing the search string is listed to the screen. If you wish to search just one section of the program, add the starting and ending line numbers, separated by commas.

If you're trying to find BASIC keywords (like PRINT or REM), use the first format. It also works for variables and numbers. The second format should be used when you're looking for strings or items inside quotation marks.

Example: **FIND @A=@** searches for lines where variable A is defined.

HELP
Use: MetaBASIC 64 command
Syntax: **HELP**

Whenever you're unsure of the commands available in MetaBASIC 64, type HELP for a complete list.

INT
Use: MetaBASIC 64 command (see also DEFAULT, QUIT)
Syntax: **INT**

Some features of MetaBASIC 64 are interrupt-driven. If you reset the interrupts (with the MONITOR command), the function keys and the SPEED function may no longer work. INT puts the MetaBASIC interrupts back in place.

KEY
Use: BASIC programming (see also INT)
Syntax: **KEY key number, *"command or string"***

This command adds a lot of flexibility to MetaBASIC 64, allowing you to define each of the eight function keys as a different command or string. (However, due to a minor bug in MetaBASIC, any definition you assign to the f8 key will be garbled whenever you use the RENUM command.) The command, up to ten letters in length for each key, must be inside quotation marks. There are two special characters: The back arrow acts as a carriage return, so you don't have to press RETURN after BASIC commands. Also, the apostrophe (SHIFT-7) counts as a double quotation mark.

Using KEY, you can load other utilities you may own and SYS to them with a tap of a function key. Or you can do a one-key RUN or LIST. If you want to permanently define the function keys and screen/text colors, you can use KEY and DEFAULT to set up the desired configuration, then save a copy of your customized version of MetaBASIC using BSAVE "METABASIC",36864,40960. The definitions will be saved along with the program.

If the interrupts are accidentally reset, you'll have to use the INT command to reenable the KEY function.

Examples:

KEY 1,"{CLR}LIST100-←" clears the screen and lists from line 100 on whenever you press f1 (the back arrow means RETURN will happen automatically). You can also abbreviate LIST with L SHIFT-I.

KEY 7,"DATA" could be useful with automatic line numbering (see AUTO) if you're writing a program with a lot of DATA statements. After entering a line, press RETURN and you'll see the next line number. Then press f7, and the word DATA automatically appears.

KEY 2,"VERIFY"'"'←" defines f2 to print VERIFY"*" plus a RETURN (note the apostrophes have been changed to quotation marks). If you've used DEFAULT to change the device number to 8, pressing f2 will automatically verify the program most recently saved to disk.

LLIST
Use: printer command
Syntax: **LLIST** starting line–ending line

This command lists a program, but the listing is sent to a printer rather than to the screen. Line numbers are optional. The syntax for LLIST is identical to the regular LIST. As written, LLIST does the equivalent of OPEN 4,4,4 to open a file for output to the printer. Some printers may require a different secondary address (the last number in the OPEN statement)—OPEN 4,4,7, for example. To change the secondary address, POKE the desired value into location 40341. If you are using a printer with a different device number (5, for example) or a plotter (device 6), you can change the device number for LLIST by POKEing the desired value into location 40339. To make the changes permanent, follow the instructions for saving a new copy of MetaBASIC given above in the discussion of the KEY command.

Example: **LLIST 10–20** to list lines 10–20 to the printer.

MEMORY
Use: ML programming (see also @)
Syntax: **MEMORY starting address**–ending address

You can examine any section of memory with this command. Use decimal numbers (not hex) for the starting and ending addresses. The values in memory are displayed, six bytes per line, in decimal. In addition, the equivalent ASCII characters are printed in reverse to the right (if there's no corresponding ASCII character, a period is printed).

If you omit the ending address, MEMORY 43 for example, you'll see the contents of two bytes (43 and 44). This makes it easier to look at two byte pointers—like 43 and 44 which point to the beginning of BASIC memory.

To change memory, you can use the @ command, described above.

Example: Enter **MEMORY 41374–41474** and you'll see the first few error messages in BASIC ROM (note that the ASCII value of the last character is always added to 128). Or, load a BASIC program, and type **MEMORY 2048–2148** to see how programs are stored in memory.

MERGE
Use: disk command
Syntax: **MERGE *"program name"***

MERGE reads a program from disk, lists each line to the screen, and adds the line to the program in memory. If the programs have common line numbers, the program on disk takes precedence. Say both programs contain a line 250. The line 250 from the disk program will replace line 250 in memory.

Before using this command, you may want to use DLIST to make sure you're merging the right program. And if there are conflicting line numbers, you can use RENUM to renumber one of the two programs. If you want to merge just part of one program, use DELETE to eliminate the unwanted lines.

MONITOR
Use: ML programming (see also INT)
Syntax: **MONITOR**

If you have a machine language monitor in memory, you can enter it with MONITOR (providing it is enabled when a BRK instruction is executed). To use MetaBASIC 64 with a monitor, you must load MetaBASIC 64, type NEW, and activate the program with SYS 36864. Next, load the monitor, type NEW, and SYS to the starting address for the monitor (which will set up the BRK vector to point to the monitor).

MONITOR does several other things, as well. It changes border, background, and text colors back to their default values (light blue on dark blue). It also resets the default device number and sets interrupts to normal, which disables the function-key definitions (see KEY) and SPEED command. You can get them back with the INT command.

NUMBER
Use: ML programming
Syntax: **NUMBER $hexadecimal number**
NUMBER decimal number

NUMBER allows you to convert back and forth between decimal and hexadecimal (hex). Put a dollar sign ($) in front of hex numbers. In addition, the number is displayed in low-byte/high-byte format (in decimal) and in binary (preceded by a percent sign).

Examples: **NUMBER $100**
256
0 1
%100000000

NUMBER 34
$22
34 0
%100010

QUIT
Use: MetaBASIC 64 command
Syntax: **QUIT**

This resets all vectors and disables all MetaBASIC 64 commands. The one thing it does *not* do is restore the top-of-memory pointer. MetaBASIC 64 is still protected from BASIC. Reactivate MetaBASIC with SYS 36864 or SYS 9*4096.

READ
Use: disk command (see also CAT, DLIST)
Syntax: **READ *"sequential filename"***

READ allows you to examine sequential disk files. The information in the file is displayed to the screen, without altering whatever program is in memory.

In the rare case that you want to use the BASIC READ statement in direct mode (to see if all DATA statements have been read, for example), you can precede it with a colon.

RENUM
Use: BASIC programming
Syntax: **RENUM** starting line, increment

This command renumbers the entire BASIC program in memory (you can't renumber just part of the program). The first line of the renumbered program will be given the specified starting line number. If you omit the starting line number, the renumbered program will begin with line 10. The increment value specifies how much the starting value will be incremented for each succeeding line. If no increment value is provided, the value defaults to 10.

In addition to renumbering BASIC lines, all references in GOTOs, GOSUBs, ON-GOTOs, ON-GOSUBs, IF-THENs, and so forth are taken care of. One word of caution: GOTO is covered, but GO TO (with a space in the middle) is not. Use FIND before renumbering to look for occurrences of GO TO.

Example: **RENUM 100,20** renumbers a program, starting at line 100, counting up by 20s.

RESAVE
Use: disk command (see also BSAVE)
Syntax: **RESAVE *"filename"***

The save-with-replace disk command (SAVE "@:*filename*") first saves the program and then scratches the older version, so there must always be enough free space on the disk for the new version of the program. This can cause problems if you don't have enough available space. The save-with-replace command is also sometimes unreliable and should be avoided.

RESAVE reverses the order—first it scratches the old version of your program from disk, and then does a regular SAVE, solving both of the above problems.

SCRATCH
Use: disk command
Syntax: **SCRATCH *"filename"***

SCRATCH does the same thing as OPEN 15,8,15: PRINT#15,"S0:*filename*": CLOSE 15, but it's easier to type. It scratches a file from the disk. If you have just inserted the disk into the drive, it's a good idea to initialize it first (see SEND). You can use wildcards to scratch more than one program—SCRATCH "A*" will get rid of all files beginning with the letter *A*. However, you should use such commands with care to avoid accidentally deleting important programs.

Example: **SCRATCH "SPACEGAME"** removes the program named SPACEGAME from the disk.

SEND
Use: disk command
Syntax: **SEND *"command string"***

This is a convenient way to send disk commands to channel 15. SEND"I0" initializes the drive, SEND"V0" validates the disk, SEND "R0:*newname*=*oldname*" renames a disk file, and so on. For more information about disk commands, see the 1541 user's manual.

SPEED
Use: BASIC programming
Syntax: **SPEED number**

SPEED changes the rate at which the 64 prints to the screen. The number supplied with the command must be in the range 0–255. The higher the number, the slower the printing speed. Try typing SPEED 255 (the slowest you can make it) and then list a program. You can get back to normal with SPEED 0. If it doesn't work, try using INT (see above) to correct the interrupts.

SPEED is useful when you're using the TRACE command.

START
Use: disk command
Syntax: **START *"filename"***

If you forget where a machine language program begins, put the disk in the drive and use this command. This can help when you have forgotten the SYS that starts a program. If this command returns the value 2049, the file you are checking is probably BASIC rather than machine language (or it at least has a single line of BASIC, like *SpeedScript*).

Example: **START "METABASIC 64"** should display 36864 on the screen.

TERMINAL
Use: modem command
Syntax: **TERMINAL**

If you own a Commodore modem (and it's plugged into your 64), TERMINAL transforms your computer into a 300 baud "dumb" terminal you can use to talk to standard-ASCII bulletin boards or information services like CompuServe. You can't change any of the default parameters, nor can you upload or download text or programs.

To return to BASIC, press the £ (English pound) key; do not press RUN/STOP–RESTORE. A note of caution: Memory locations 52736–53247 ($CE00–$CFFF) are used for buffers, so any program in this area will be overwritten.

TRACE
Use: BASIC programming (see also TROFF)
Syntax: **TRACE**

If you're debugging a BASIC program, TRACE helps you see what's happening. As each line is executed, its line number is printed on the screen. Use the SHIFT or CTRL keys to temporarily halt the program. SPEED controls the speed of execution, and TROFF turns off TRACE.

TROFF
Use: BASIC programming (see also TRACE)
Syntax: **TROFF**

This command turns off the TRACE function.

UNNEW
Use: BASIC programming
Syntax: **UNNEW**

You may never need this command, but it's nice to have it available. If you accidentally type NEW and you want to retrieve the program, use UNNEW to get it back.

VCHANGE
Use: BASIC programming (see also CHANGE, FIND)
Syntax: **VCHANGE @*old string*-@*new string*@**, starting line, ending line

VCHANGE @"*old string*-"@"*new string*"@, starting line, ending line

VCHANGE (Verify CHANGE) works just like CHANGE (see above), except you get to choose whether or not each change is made. Each line containing the old string is displayed, with each occurrence of the string marked with a filled-in circle. If you press Y, the change is made. Press N if you want to skip to the next occurrence of the old string. ©

MetaBASIC Plus

John Brox Shadle

"MetaBASIC Plus" is a companion program to "MetaBASIC 64." It adds 11 new commands and modifies HELP to print the new MetaBASIC Plus commands in addition to the original MetaBASIC commands. To create MetaBASIC Plus, you must have a working copy of MetaBASIC 64. See the "MetaBASIC 64" article on the preceding pages.

Creating MetaBASIC Plus

MetaBASIC Plus is a collection of routines to add new commands to MetaBASIC 64. Like the original program, the MetaBASIC Plus routines are written in machine language. To install MetaBASIC Plus, follow these steps carefully:

1. Load MetaBASIC 64 using the command LOAD "METABASIC 64",8,1.
2. Load MetaBASIC Plus using the command LOAD "METABASIC PLUS",8,1.
3. Enter a NEW command to reset memory pointers.
4. Type SYS 35126 and press RETURN. This calls a short (19-byte) routine at the beginning of MetaBASIC Plus which patches MetaBASIC Plus into the original MetaBASIC and activates the combined programs.

Now you're ready to save a copy of the new version of MetaBASIC, which has the additional MetaBASIC Plus commands. Before doing so, however, you might want to use the KEY command to set up some default function-key definitions that will be enabled whenever you activate the new MetaBASIC. When you're ready to save a new copy to disk, simply use a command of the form

```
BSAVE "METABASIC+",35145,40960
```

Once you've saved a copy of the combined file, load and activate the new version of MetaBASIC just like you did the old version, with LOAD "METABASIC+",8,1 and SYS 36864 (or SYS 9*4096).

MetaBASIC Plus Commands

Here's a rundown on the new commands.

CURSORFREE
Use: BASIC programming (see also CURSORPRINT)
Syntax: **CURSORFREE**

Disables quote mode until the CURSORPRINT command is given or until RUN/STOP–RESTORE is pressed.

CURSORPRINT
Use: BASIC programming (see also CURSORFREE)
Syntax: **CURSORPRINT**

Re-establishes quote mode, reversing the effect of the CURSORFREE command.

END
Use: BASIC programming
Syntax: **END**

Prints the last memory location used by the BASIC program currently in memory.

FREEMEM
Use: BASIC programming
Syntax: **FREEMEM**

Prints the number of bytes available without performing any garbage collection.

MAKEDATA
Use: BASIC programming
Syntax: **MAKEDATA starting address, ending address**

Converts the contents of the specified area of memory into DATA statements. Any BASIC program in memory will be erased by this statement. MAKEDATA can convert a maximum of 4352 bytes. The DATA statements created will be numbered starting at line zero, and will include 17 data items per line. After creating the DATA statements, you can use the MetaBASIC RENUM command to renumber the lines, then use the MERGE command to add the DATA lines to another program.

POINTER
Use: BASIC programming
Syntax: **POINTER address**

This is especially useful for checking BASIC's pointers. This command does the equivalent of PRINT PEEK(address) +256*PEEK(address + 1).

RESET
Use: MetaBASIC Plus command
Syntax: **RESET**

Turns off MetaBASIC Plus. This is the same as MetaBASIC 64's QUIT, except that all of the memory used by MetaBASIC is deallocated. MetaBASIC can be turned back on (with SYS 36864 or SYS 9*4096) if no files have been loaded and no new strings have been created.

SCREEN
Use: printer command
Syntax: **SCREEN**

Performs a screen dump, sending whatever is on the screen to the printer. This works in normal text mode only and will not dump screens with redefined characters or high-resolution bit-mapped screens.

The SCREEN command routine does the equivalent of OPEN 1,4,255. Some printers may require a different secondary address—7 is common. To change the secondary address used by SCREEN, POKE location 35369 with the desired secondary address value, then BSAVE a new copy of MetaBASIC using the command in the instructions above. If you want to change the device number used by SCREEN to a value other than 4, POKE the desired value into location 35367 and BSAVE a new copy.

SUPERPEEK
Use: ML or BASIC programming
Syntax: **SUPERPEEK starting address, number**

Creates a window at the top of the screen which shows the contents of an

area of memory as decimal values. The display begins with the starting address and includes the specified number of bytes (the number value must be no larger than 50). If you run a BASIC or machine language program while the SUPERPEEK window is active, the memory contents will continue to be displayed as the program runs. Thus, you can see if the program changes the contents of any of the displayed locations. Press RUN/STOP to see the character code equivalents of the numbers. The INT command turns off SUPERPEEK.

TRAP
Use: BASIC programming (see also UNTRAP)
Syntax: **TRAP**

Turns on extended error analysis. When an error occurs during a program run, the line causing the error is listed, with the word *ERROR* in reverse video at the point where the error occured. Error trapping doesn't work in direct mode; it only functions within a program. Use UNTRAP to turn off extended error analysis.

UNTRAP
Use: BASIC programming (see also TRAP)
Syntax: **UNTRAP**

Disables extended error analysis to cancel the effect to the TRAP command.©

MetaBASIC 128

Kevin Mykytyn

"MetaBASIC 128," will change the way you program. It adds 11 new debugging and testing commands to BASIC 7.0—and these commands are instantly at your fingertips for programming sessions.

Using MetaBASIC 128

MetaBASIC 128 commands use English mnemonics, so you don't have to memorize a lot of SYS numbers. Once MetaBASIC 128 is active, you'll have these 11 additional commands: AID, CHANGE, DEFAULT, DLIST, FIND, MERGE, QUIT, READ, RESAVE, START, and UNNEW.

The commands work only in direct mode; you cannot add them to programs. Also, you're limited to one MetaBASIC command per line (although you can still use multistatement lines inside your programs). Unlike ordinary BASIC commands, there are no abbreviations. You must type out the entire MetaBASIC 128 command. If you wish to stop the execution of a command, press the RUN/STOP key (*not* RUN/STOP-RESTORE). If it seems to be working incorrectly, make sure the syntax is correct.

Machine language programmers should remember that MetaBASIC 128 occupies memory locations $1300–$18BF (4864–6335) and uses zero-page locations $FB–$FE (251–254) and $AC–$AF (172–175).

To use MetaBASIC 128, follow these steps:

1. Load the program with the command BLOAD "METABASIC 128".
2. Type SYS 4864 to activate MetaBASIC 128.

After the SYS, it may seem that nothing has changed. But MetaBASIC 128 is active, and you now have 11 new commands to help you write and debug programs.

MetaBASIC 128 Commands

Here's an alphabetical list of the new commands and how to use them, with examples. MetaBASIC 128 commands and mandatory parameters appear in boldface. String parameters appear in italics. Optional parameters appear in normal print.

If something is described as a disk command, it won't work unless you have a disk drive. However, some of the ML programming aids can be useful in BASIC, and vice versa.

AID
Syntax: **AID**

Lists all available MetaBASIC 128 commands.

CHANGE
Syntax: **CHANGE @*old string*@*new string*@**, starting line, ending line
CHANGE @*old string*@*new string*@, starting line
CHANGE @*old string*@*new string*@,,ending line
CHANGE */old string/new string/*, starting line, ending line
CHANGE */old string/new string/*, starting line
CHANGE */old string/new string/*,,ending line

See also FIND.

CHANGE searches through the program in memory, changing every occurrence of the old string to the new one. The strings can be up to 30 characters long and must be bracketed by the commercial at sign (@) or the slash (/). All lines in which changes are made are listed to the screen. The format with @ is the tokenized form and should be used to change BASIC commands and variable names. The ASCII form (the slash format) is useful when you want to change a word in a string without changing keywords. For example:

CHANGE /PRINT/WRITE/

changes all occurrences of the word *PRINT* within quotation marks without changing any PRINT statements.

Use the slash format to change anything inside quotation marks or after a REM statement; use the at sign format to change anything not inside quotation marks or after a REM statement. Remember that mathematical

operators within programs such as +, −, *, /, >, <, and = are stored as tokens, not characters, so you must use the @ format when searching for one of these.

If you omit the line numbers, CHANGE affects the whole program. If you want to change only one section, add the starting and ending line numbers, marked off by commas.

Example: **CHANGE @X@QQ@ „200** changes the variable X to QQ in all lines up to and including 200. To change the name Charles to John throughout the program, **CHANGE /CHARLES/JOHN/**.

DEFAULT
Syntax: **DEFAULT border color, background color, text color**

See also QUIT.

When you press RUN/STOP–RESTORE, the screen reverts to the default colors light green and black. DEFAULT lets you change these values to whatever you prefer. If your 128 is hooked up to a black-and-white TV, change the character and background colors to a more readable combination. The border- and background-color changes affect only the 40-column screen; the text-color change affects both the 40- and 80-column displays.

To disable DEFAULT (and go back to normal colors), use the QUIT command.

Example: **DEFAULT 1,1,0** changes border and background to white, and characters to black. If you press RUN/STOP–RESTORE, you'll see black characters on a white background.

DLIST
Syntax: **DLIST *"filename"***

See also READ.

This command lists a BASIC program from disk to the screen without affecting what's currently in memory. The program name must be enclosed in quotation marks. DLIST enables you to look at a program before using MERGE or SCRATCH.

Example: **DLIST "BASICPROGRAM"** reads the program file named BASICPROGRAM from disk and lists it to the screen.

FIND
Syntax: **FIND @*string*@**, starting line, ending line
FIND @*string*@, starting line
FIND @*string*@,, ending line
FIND /*string*/, starting line, ending line
FIND /*string*/, starting line
FIND /*string*/,, ending line

See also CHANGE.

This allows you to find any word, variable, or other string within a program. Each line containing the search string is listed to the screen. If you wish to search just one section of the program, add the starting and ending line numbers, separated by commas.

If you're trying to find BASIC keywords (like PRINT or REM), use the first format with the @ symbols. It also works for variables and numbers. The second format should be used when you're looking for strings or items inside quotation marks.

Example: **FIND @A=@** searches for lines where variable A is defined.

MERGE
Syntax: **MERGE *"program name"***

MERGE reads a program from disk, lists each line to the screen, and adds the line to the program in memory. If the programs have common line numbers, the program on disk takes precedence. Say both programs contain a line 250. The line 250 from the disk program will replace line 250 in memory.

Before using this command, you may want to use DLIST to make sure you're merging the right program. And if there are conflicting line numbers, you can use RENUMBER to renumber one of the two programs. If you want to merge just part of one program, use DELETE to eliminate the unwanted lines.

QUIT
Syntax: **QUIT**

This resets all vectors and disables all MetaBASIC commands. MetaBASIC is still protected from BASIC. Reactivate MetaBASIC with SYS 4864.

READ
Syntax: **READ *"filename"***

See also DLIST.

READ allows you to examine sequential disk files. The information in the file is displayed to the screen, without altering whatever program is in memory.

In the rare case that you want to use the BASIC READ statement in direct mode (to see if all DATA statements have been read, for example), you can precede it with a colon to distinguish it from MetaBASIC 128's READ command.

RESAVE
Syntax: **RESAVE *"filename"***

The save-with-replace disk command (SAVE "@0:*filename*") first saves the new version of the program and then scratches the older version, so there must always be enough free space on the disk for both versions. Thus, the command can cause problems if you don't have enough available disk space for the new version. The save-with-replace command also has other problems and is best avoided.

RESAVE reverses the order—first it scratches the old version of your program from disk, and then it does a regular SAVE, solving both of the above problems.

START
Syntax: **START *"filename"***

If you forget where a machine language program begins, put the disk in the drive and use this command. This can help when you have forgotten the SYS that starts a program. If the command returns a value of 7169, the program is probably BASIC or a machine language program with a single BASIC line so that it starts with RUN rather than SYS.

Example: **START "METABASIC 128"** should display 4864 on the screen.

UNNEW
Syntax: **UNNEW**

You may never need this command, but it's nice to have it available. If you accidentally type NEW and you want to retrieve the program, use UNNEW to get it back. ©

SpeedScript 3.2

For The Commodore 64

Charles Brannon

Since its introduction in the January 1984 issue of COMPUTE!'s Gazette, SpeedScript *has been the most popular program ever published by COMPUTE! Publications. Written entirely in machine language,* SpeedScript *contains nearly every command and convenience you'd expect from a quality word processor. The latest version of Commodore 64* SpeedScript, *version 3.2, incorporates many improvements, readers' suggestions, and additional debugging. This version also works on a Commodore 128 in 64 mode.*

SpeedScript 3.2, though compact in size (6K), has many features found on commercial word processors. *SpeedScript* is also very easy to learn and use. You can start writing with it the first time you use it. You type in everything first; preview and make corrections on the screen; insert and delete words, sentences, and paragraphs; and then print out an error-free draft, letting *SpeedScript* take care of things like margins, centering, headers, and footers.

SpeedScript is a writing tool. It won't necessarily make you a better writer, but you may become a better writer once the tedium of retyping and erasing is replaced by the flexibility of a word processor. Words are no longer frozen in place by ink; they become free-floating entities. You no longer think about typewriting—you can stand back and work directly with words and ideas. The distinction between rough and final drafts becomes blurred as you perfect your writing as you write.

Loading *SpeedScript*

SpeedScript can be loaded just as if it were a BASIC program. If you load *SpeedScript* and list it, you'll see that it looks like a one-line BASIC program. This BASIC line is included to make the program easy to load, run, and copy. It's a good idea to save a couple of extra copies of *SpeedScript*, just in case the original is destroyed. To do this, type SAVE"SPEEDSCRIPT3.2",8 (or ,1 if you're using tape) after loading *SpeedScript*, just as you would for a BASIC program. Use whatever filename you like.

Before using *SpeedScript*, you should unplug any cartridges and expanders. *SpeedScript* cannot take advantage of any custom hardware configurations except those that do not interfere with normal operations.

Entering Text

When you run *SpeedScript*, the screen colors change to dark gray on light gray except for the top screen line, which is black with white letters. This *command line* is used to communicate with *SpeedScript*. *SpeedScript* presents all messages here. The remaining lines of the screen are used to enter, edit, and display your document. The *cursor* shows where the next character you type will appear on the screen. *SpeedScript* lets you move the cursor anywhere within your document, making it easy to find and correct errors.

To begin using *SpeedScript*, just start typing. When the cursor reaches the right edge of the screen, it automatically jumps to the beginning of the next line, just as in BASIC. But unlike BASIC, *SpeedScript* never splits words at the right edge of the screen. If a word you're typing won't fit at the end of one line, it's instantly moved to the next line. This feature, called *word-wrap*, makes it much easier to read your text on the screen. Even if you make numerous editing changes, *SpeedScript* reformats the screen and rewraps all words.

Scrolling And Screen Formatting

When you finish typing on the last screen line, *SpeedScript* automatically scrolls the text upward to make room for a new line at the bottom. This is similar to the way BASIC works, but with one excep-

tion: The screen can scroll both up *and* down. Imagine the screen as a 24-line window on a long, continuous document.

More than 43K of text space is available in memory, room enough for 20–40 printed pages of text. To check at any time how much space is left, press **CTRL-=** (hold down the CTRL key while pressing the = key). The number which appears on the command line indicates how much room remains for characters of text.

If you're used to a typewriter, you'll have to unlearn some habits. First, since the screen is only 40 columns wide, and most printers have 80-column carriages, it doesn't make sense to press RETURN at the end of each line as you do on a typewriter. *SpeedScript*'s word-wrap takes care of this automatically. Press RETURN only when you want to force a carriage return to end a paragraph or to limit the length of a line. So that you can see these forced carriage returns, they appear on the screen as left-pointing arrows (called *return marks* in this article).

When you print your document, *SpeedScript* automatically formats your text to fit the width of the paper. Don't manually space over for a left margin or try to center a line yourself as you would on a typewriter. *SpeedScript*'s printing routine automatically takes care of all margins and centering and lets you customize the margin settings. Also, don't worry about where a printed page will end. When printing, *SpeedScript* automatically fits your text onto separate pages and can even put short phrases and page numbers at the top or bottom of each page if you want.

Like all good word processors, *SpeedScript* has a wide selection of editing and convenience features. You can move the cursor a single space in either direction, or skip to the next or previous word, sentence, or paragraph. You can also move the cursor to the top of the screen, the top of the document, or the end of the document. The INST/DEL key is used to insert a single space or delete a single character. Other features let you erase a word, sentence, or paragraph, and move or copy sentences, words, and paragraphs to other places in your document. Using Search-and-Replace, you can find any phrase and even automatically change one phrase to another throughout the entire document.

You can save your text on tape or disk, then load it later for additions and corrections. You can transpose (exchange) two characters, change the screen and text colors, send disk commands, read the disk error channel, and automatically tab over five spaces for paragraph indents. You don't need to learn all these commands right away, but you'll be glad they're available as you become more comfortable with word processing.

Using The Keyboard

Most of *SpeedScript*'s features are accessed with control-key commands—you hold down CTRL while pressing another key. In this article, control-key commands are abbreviated **CTRL-*x*** (where *x* is the key you press in combination with CTRL). An example is the **CTRL-=** mentioned above to check on free memory. **CTRL-E** means hold down CTRL and press E. Sometimes you have to hold down both SHIFT and CTRL as you type the command key, as in SHIFT–CTRL-H. Other keys are referenced by name or function, such as back arrow for the left-pointing arrow in the top-left corner of the keyboard, pound sign for the British pound symbol (£), CLR/HOME for the home-cursor key, SHIFT–CLR/HOME for the clear-screen key, f1 for special-function key 1, and up arrow for the upward-pointing arrow to the left of the RESTORE key.

Some keys let you move the cursor to different places in the document to make corrections or scroll text into view. *SpeedScript* uses a unique method of cursor movement that is related to writing, not programming. Programmers work with lines of text and need to move the cursor up and down a line or left and right across a line. *SpeedScript*, however, is oriented for writers. You aren't working with lines of text, but with a continuous document.

Therefore, *SpeedScript* moves the cursor by character, word, sentence, or paragraph. *SpeedScript* defines a word as any sequence of characters preceded or followed by a space. A sentence is any sequence of characters ending with a period, exclamation point, question mark, or return mark. And a paragraph is defined as any sequence of characters ending in a return mark. (Again, a return mark appears on the screen as a left-pointing arrow.)

Here's how to control the cursor:

- The **left/right-cursor key** works as usual; pressing this key by itself moves the cursor right (forward) one space, and pressing it with SHIFT moves the cursor left (backward) one space.
- The **up/down-cursor key** moves the cursor forward to the beginning of the next sentence. Pressing it with SHIFT moves the cursor backward to the beginning of the previous sentence.
- The **f1 function key** moves the cursor forward to the beginning of the next word. The **f2 key** (hold down SHIFT and press f1) moves the cursor backward to the beginning of the previous word.
- The **f3 function key** moves the cursor forward to the beginning of the next sentence (just like the up/down-cursor key). The **f4 key** (hold down SHIFT and press f3) moves the cursor backward to the beginning of the previous sentence (just like pressing SHIFT and the up/down-cursor key).
- The **f5 function key** moves the cursor forward to the beginning of the next paragraph. The **f6 key** (hold down SHIFT and press f5) moves the cursor backward to the beginning of the previous paragraph.
- **CLR/HOME**, pressed once by itself, moves the cursor to the top of the screen without scrolling. Pressed twice, it moves the cursor to the beginning of the document.
- **CTRL-Z** moves the cursor to the bottom of the document.

Correcting Your Typing

One strength of a word processor is that you need never have mistakes in your printed document. Since you've typed everything before you print it, you have plenty of opportunities to proofread and correct your work. The easiest way to cor-

Figure 1: *SpeedScript* Keyboard Map

CTRL A	Change case
CTRL B	Change border color
CTRL D	Delete (Sentence, Word, Paragraph)
CTRL E	Erase (Sentence, Word, Paragraph)
CTRL G	Global search and replace
CTRL H	Hunt for phrase with SHIFT: Select hunt phrase
CTRL I	Enter/exit insert mode
CTRL J	Replace with SHIFT: Select replace phrase
CTRL K	Kill buffer
CTRL L	Change text character color
CTRL P	Print
CTRL R	Restore buffer
CTRL V	Verify
CTRL X	Transpose characters
CTRL Z	Go to end of text
CTRL =	Display amount of free memory
CTRL ↑	Send disk command or read error channel
CTRL 4	Display disk directory
CTRL £	Enter format (printer) commands
CTRL 3	Commodore 64 only: Same as CTRL-£
CLR/HOME	Press once to go to top of screen Hold down to go to top of text with SHIFT: Erase all text
CRSR (left/right)	Move the cursor left one character with SHIFT: Move the cursor right one character
CRSR (up/down)	Got to next sentence with SHIFT: Go to previous sentence
RUN/STOP	Indent 5 spaces with SHIFT: Insert 255 spaces
RESTORE	Exit *SpeedScript* (Commodore 64) with RUN/STOP: Exit *SpeedScript* (VIC-20)
	Backspace with CTRL: Delete character under cursor and close up text with SHIFT and CTRL: Delete all spaces from cursor to next character
RETURN	Return mark with SHIFT: End paragraph, add an extra return mark, and indent next paragraph
INST/DEL	Delete character with SHIFT: Insert space

f1	Next Word
f2	Previous Word
f3	Next Sentence
f4	Previous Sentence
f5	Next Paragraph
f6	Previous Paragraph
f7	Load
f8	Save

rect something is just to type over it, but there are other ways too.

Sometimes you'll have to insert characters to make a correction. Maybe you accidentally dropped a letter, typing *hngry* instead of *hungry*. When you change the length of a word, you need to push over everything to the right of the word to make room for the insertion. Use **SHIFT-INST/DEL** to open up a single space, just as in BASIC. Merely position the cursor at the point where you want to insert a space, and press SHIFT-INST/DEL.

Insert Modes

It can be tedious to use the SHIFT-INST/DEL key to open up enough space for a whole sentence or para-

graph. For convenience, *SpeedScript* has an insert mode that automatically inserts space for each character you type. In this mode, you can't type over characters; everything is inserted at the cursor position. To enter insert mode, press **CTRL-I**. To cancel insert mode, press CTRL-I again. (This kind of command key, one which is used to turn something both on and off, is called a *toggle*). To let you know you're in insert mode, the normally black command line at the top of the screen turns blue.

Insert mode is the easiest way to insert text, but it can become too slow when you're working with a very long document because it must move *all* the text following the cursor position. Although *SpeedScript* uses turbocharged memory-move routines, the 6510 microprocessor can go only so fast. So *SpeedScript* has even more ways to insert blocks of text.

One way is to use the **RUN/STOP** key. It is programmed in *SpeedScript* to act as a five-space margin indent. To end one paragraph and start another, press RETURN twice and press RUN/STOP. Alternatively, you can press **SHIFT-RETURN**, which does this automatically. You can use RUN/STOP to open up more space than SHIFT-INST/DEL. No matter how much space you want to insert, each insertion takes the same amount of time. So the RUN/STOP key can insert five spaces five times faster than pressing SHIFT-INST/DEL five times.

There's an even better way, though. Press **SHIFT-RUN/STOP** to insert 255 spaces. This is enough room for a sentence or two. You can press it several times to open up as much space as you need. And SHIFT-RUN/STOP is *fast*. (You don't want to be in insert mode when you use this trick; that would defeat its purpose.)

Since the INST/DEL key is also slow when you're working with large documents (it, too, must move all text following the cursor), you may prefer to use the back-arrow (←) key to backspace. The **back-arrow** key by itself moves the cursor left one space and blanks out that position. It's more like a backspace than a delete.

After you're finished inserting with these methods, there will probably be some inserted spaces left over that you didn't use. Just press **SHIFT-CTRL-back arrow**. This instantly deletes all extra spaces between the cursor and the start of the following text. SHIFT-CTRL-back arrow is also generally useful whenever you want to delete a bunch of spaces.

Erasing Text

Inserting and retyping are not the only kinds of corrections you'll need to make. Part of writing is separating the wheat from the chaff. On a typewriter, you pull out the paper and throw it away. *SpeedScript* lets you be more selective.

Press the **INST/DEL** key by itself to erase the character to the left of the cursor. All the following text is pulled back to fill the vacant space.

Press **CTRL-back arrow** to delete the character on which the cursor is sitting. Again, all the following text is moved toward the cursor to fill the empty space.

These keys are fine for minor deletions, but it could take a long time to delete a whole paragraph this way. So *SpeedScript* has two commands that can delete an entire word, sentence, or paragraph at a time. **CTRL-E** erases text *after* (to the right of) the cursor position, and **CTRL-D** deletes text *behind* (to the left of) the cursor.

To use the **CTRL-E** (erase) mode, first place the cursor at the beginning of the word, sentence, or paragraph you want to erase. Then press CTRL-E. The command line shows the message *Erase (S,W,P): RETURN to exit.* Press S to erase a sentence, W for a word, or P for a paragraph. Each time you press one of these letters, the text is quickly erased. You can keep pressing S, W, or P until you've erased all the text you wish. Then press RETURN to exit the erase mode.

The **CTRL-D** (delete) mode works similarly, but deletes only one word, sentence, or paragraph at a time. First, place the cursor after the word, sentence, or paragraph you want to delete. Then press CTRL-D. Next, press S, W, or P for sentence, word, or paragraph. The text is immediately deleted and you return to editing. You don't need to press RETURN to exit the CTRL-D mode unless you pressed this key by mistake. *(In general, you can escape from any command in* SpeedScript *by simply pressing RETURN.)* CTRL-D is most convenient when the cursor is already past what you've been typing.

The Text Buffer

When you erase or delete with CTRL-E and CTRL-D, the text isn't lost forever. *SpeedScript* remembers what you've removed by storing deletions in a separate area of memory called a *buffer*. The buffer is a fail-safe device. If you erase too much or change your mind, just press **CTRL-R** to restore the deletion. However, be aware that *SpeedScript* remembers only the last erase or delete you performed.

Another, more powerful use of this buffer is to move or copy sections of text. To move some text from one location in your document to another, first erase or delete it with CTRL-E or CTRL-D. Then move the cursor to where you want the text to appear and press **CTRL-R.** CTRL-R instantly inserts the contents of the buffer at the cursor position. If you want to copy some text from one part of your document to another, just erase or delete it with CTRL-E or CTRL-D, restore it at the original position with CTRL-R, and then move the cursor elsewhere and press CTRL-R to restore it again. You can retrieve the buffer with CTRL-R as many times as you like.

Important: The CTRL-E erase mode lets you erase up to the maximum size of the buffer (12K, or over 12,000 characters). CTRL-E normally removes the previous contents of the buffer each time it is used. Keep this in mind if there's something in the buffer you'd rather keep. If you don't want the current buffer contents to be erased, press **SHIFT-CTRL-E** instead. This preserves the buffer contents and adds newly erased text to the buffer.

Now you can see why CTRL-D lets you delete only a single sentence, word, or paragraph at a time. If it didn't, the deleted text would be added to the end of the buffer, and when you pressed CTRL-R to retrieve the buffer, the deleted text

would be out of order (since CTRL-D deletes backward).

If you ever need to erase the contents of the buffer, press **CTRL-K** (remember *kill buffer*).

It's relatively easy to move blocks of text between documents. Using the buffer, you can load one document, erase some text into the buffer, load another document, and then insert the buffer contents into the new document. You can also use the buffer to save an often-used word or phrase, then repeat it whenever you need it.

Starting From Scratch

If you want to start a new document or simply obliterate all your text, press **SHIFT-CLR/HOME**. *SpeedScript* asks, *ERASE ALL TEXT: Are you sure? (Y/N).* This is your last chance. If you *don't* want to erase the entire document, press N or any other key. Press Y to perform the irreversible deed. There is no way to recover text wiped out with Erase All.

The RUN/STOP–RESTORE reset combination on the Commodore 64 has been disabled in *SpeedScript*.

As mentioned above, pressing RUN/STOP by itself inserts five spaces for indenting paragraphs.

Pressing just RESTORE brings up the message *Exit SpeedScript: Are you sure? (Y/N).* If you press Y for yes, you exit to BASIC. (If you press N or any other key at the prompt, you return to editing text with no harm done.) Once in BASIC you'll still have one chance to reenter *SpeedScript* without losing your text—simply enter RUN and your text should be intact when *SpeedScript* is restarted. (Your chances of recovering text decrease if you execute other commands while in BASIC.)

Search-And-Replace

Here's another feature only a computer can bring to writing. *SpeedScript* has a Hunt command that searches through your document to find a selected word or phrase. A Replace option lets you automatically change one word to another throughout the document. Since on the 64, CTRL-S is synonymous with the CLR/HOME key (try it), and since *SpeedScript* already uses CTRL-R, several command keys which are slightly less than mnemonic have been designated for these functions.

SHIFT-CTRL-H activates the Hunt feature, **SHIFT-CTRL-J** (J is used because it's next to the H) lets you selectively hunt and replace, and **CTRL-G** (Global) is for automatically searching and replacing.

Searching for something is a two-step process. First, you need to tell *SpeedScript* what to search for; then you must trigger the actual search. Press SHIFT–CTRL-H. The command line says *Hunt for:*. Type in what you'd like to search for—the *search phrase*—up to 29 characters. *SpeedScript* remembers the search phrase until you change it. (Incidentally, when you are typing on the command line, the only editing key that works is INST/DEL for backing up. *SpeedScript* does not let you enter control codes or cursor controls when you're typing on the command line.) Press RETURN after you've entered the search phrase. If you press RETURN alone without typing anything, the Hunt command is canceled.

When you're ready to search, press **CTRL-H** (without the SHIFT). *SpeedScript* looks for the next occurrence of the search phrase *starting from the current cursor position.* If you want to hunt through the entire document, press CLR/HOME twice to move the cursor to the very top before beginning the search. Each time you press CTRL-H, *SpeedScript* looks for the next occurrence of the search phrase and places the cursor at the start of the phrase. If the search fails (if the search phrase isn't found before the end of the document), you'll see the message *Not Found.*

CTRL-J (Replace) works together with CTRL-H. After you've specified the search phrase with SHIFT–CTRL-H, press SHIFT–CTRL-J to select the replace phrase. *SpeedScript* also remembers this replace phrase until you change it. You can press RETURN alone at the *Replace with:* prompt to select a *null* replace phrase. When you hunt and replace, this deletes the located phrase. To search and replace manually, start by pressing CTRL-H. After *SpeedScript* has found the search phrase, press CTRL-J (without SHIFT) if you want to replace the phrase. If you don't want to replace the phrase, don't press CTRL-J. You are not in a special search-and-replace mode. You're free to continue writing at any time.

CTRL-G links CTRL-H and CTRL-J together. It first asks *Hunt for:*, then *Replace with:*, and then automatically searches and replaces throughout the document starting at the cursor position.

A few hints and cautions: First, realize that if you use *the* as the search phrase, *SpeedScript* dutifully finds the embedded *the* in words like *therefore* and *heathen.* If you changed all occurrences of *the* to *cow,* these words would become *cowrefore* and *heacown.* If you want to find or replace a single word, include a space as the first character of the word, since almost all words are preceded by a space. Naturally, if you are replacing, you need to include the space in the replace phrase, too. Also, *SpeedScript* distinguishes between uppercase and lowercase. The word *Meldids* does not match with *meldids. SpeedScript* will not find a capitalized word unless you capitalize it in the search phrase. To cover all bases, you will sometimes need to make two passes when replacing a word. Keep these things in mind when using CTRL-G since you don't have a chance to stop an out-of-control search-and-replace.

Storing Your Document

Another advantage of word processing is that you can store your writing. A Commodore disk, with 170K of storage space, can store 80–150 pages of text as several document files. Tapes also have great storage capability—but they're slower.

It's easy to store a document. First, make sure your cassette or disk drive is plugged in and functioning. Insert a tape or disk into the drive. Press the **f8** function key (SHIFT-f7). You'll see the prompt *Save:*. Type in a filename for your document. A filename can be up to 16 characters long and can include almost any characters, but do not use question marks or asterisks. You cannot use the same name for two different documents on a single disk. To replace a document already on disk using the same file-

name, precede your filename with the characters **@0:** or **@:**. You can also precede the filename with either **0:** or **1:** if you use a dual disk drive. *SpeedScript* cannot access a second disk drive with a device number of 9.

After entering the filename, answer the prompt *Tape or Disk* by pressing either the **T** or **D** key. You can cancel the Save command by pressing RETURN without typing anything else at either the *Save:* or *Tape or Disk* prompt.

After you press T for tape, press RECORD and PLAY simultaneously on the cassette drive. *SpeedScript* begins saving. If you press D for disk, your file is stored relatively quickly (if the disk is formatted and has room). After the Save, *SpeedScript* reports *No errors* if all is well, or reads and reports the disk error message otherwise.

The Commodore 64 is not able to detect errors during a tape Save, so *SpeedScript* provides a verify command. Rewind the tape, press **CTRL-V**, and then type the filename. Press T for tape; press PLAY on the recorder. *SpeedScript* will notify you if there is an error.

Loading A Document

To recall a previously saved document, press the **f7** function key. Answer the *Load:* prompt with the filename. Insert the tape or disk with the file you want to load and press T or D. Press PLAY if you're using tape. *SpeedScript* loads the file and should display *No errors.* Otherwise, *SpeedScript* reads the error channel of the disk drive or reports *Load error* for tape.

It's important to position of the cursor correctly before loading a file. *SpeedScript* starts loading at the cursor position, so be sure to press CLR/HOME twice or SHIFT–CLR/HOME (Erase All) to move the cursor to the start of text space, unless you want to merge two documents. When you press f7 to load, the command line turns green to warn you if the cursor is not at the top of the text space.

To merge two or more files, simply load the first file, press CTRL-Z to move the cursor to the end of the document, and then load the file you want to merge. Do not place the cursor somewhere in the middle of your document before loading. A Load does not insert the characters coming in from tape or disk into your old text, but overwrites all existing text after the cursor position. The last character loaded becomes the new end-of-text marker, and you cannot access any of your old text that may appear after this marker.

Disk Commands

Sometimes you forget the name of a file or need to scratch or rename a file. *SpeedScript* gives you full control over the disk drive. To view the disk directory, press **CTRL-4**. The directory will be displayed on the screen without affecting the text in memory. Press any key to pause scrolling. Afterward, press RETURN to switch back to your text. All the other disk commands are also accessible. Just press **CTRL-↑** (up arrow); then type in a 1541 disk command. You don't need to type PRINT#15 or any quotation marks as you do in BASIC—just the actual command. If you press RETURN without typing a disk command, *SpeedScript* displays the disk status. It also displays the status after completing a disk command. Here is a quick summary of disk commands:

n:*disk name,ID* This formats (NEWs) a disk. You must format a new disk before using it for the first time. The disk name can be up to 16 characters. The ID (identifier) is any two characters. You must use a unique ID for each disk you have. Don't forget that this command erases any existing data on a disk.

s:*filename* Scratches (deletes) a file from the disk.

r:*newname=oldname* Changes the name of file *oldname* to *newname.*

c:*backup filename=original name* Creates a new file (the backup copy) of an existing file (original copy) on the same disk.

i: Initializes a disk. This resets several disk variables and should be used after you swap disks or when you have trouble reading a disk.

v: Validates a disk. This recomputes the number of available blocks and can sometimes free up disk space. Always use Validate if you notice a filename on the directory flagged with an asterisk. Validate can take awhile to finish.

uj: Resets the disk drive to power-up state.

Additional Features

SpeedScript has a few commands that don't do much, but are still nice to have. **CTRL-X** exchanges the character under the cursor with the character to the right of the cursor. Thus, you can fix transposition errors with a single keystroke. **CTRL-A** changes the character under the cursor from uppercase to lowercase or vice versa. You can hold down CTRL-A to continue changing the following characters.

Press **CTRL-B** to change the background and border colors. Each time you press CTRL-B, one of 16 different background colors appears. Press **CTRL-L** to cycle between one of 16 character (lettering) colors. The colors are preserved until you change them. In fact, if you exit and resave *SpeedScript*, the program will load and run with your color choice in the future.

Printing

If you already think *SpeedScript* has plenty of commands, wait until you see what the printing package offers. *SpeedScript* supports an array of powerful formatting features. It automatically fits your text between left and right margins that you can specify. You can center a line or block it against the right margin. *SpeedScript* skips over the perforation on continuous-form paper, or it can wait for you to insert single-sheet paper. A line of text can be printed at the top of each page (a *header)* and/or at the bottom of each page (a *footer),* and it can include automatic page numbering, starting with whatever number you like.

SpeedScript can print on different lengths and widths of paper, and single-, double-, or triple-spacing (or any spacing, for that matter) is easy. You can print a document up to the size that can be held on a disk or tape by linking several files together during printing. You can print to the screen or to a sequential disk file instead of to a printer. Other features let you print to most printers using most printer interfaces, and send special codes to the printer to

control features like underlining, boldfacing, and double-width type (depending on the printer).

But with all this power comes the need to learn additional commands. Fortunately, *SpeedScript* sets most of these variables to a default state. If you don't change these settings, *SpeedScript* assumes a left margin of 5, a right-margin position of 75, no header or footer, single-spacing, and continuous-paper page feeding. To begin printing, simply press **CTRL-P**. If your printer is attached, powered on, and selected (online), *SpeedScript* begins printing immediately. To cancel printing, hold down the **RUN/STOP** key until printing stops.

Before printing, be sure the paper in your printer is adjusted to top-of-form (move the paper perforation just above the printing element). CTRL-P assumes a Commodore printer, so it's helpful if your interface simulates the modes and codes of the Commodore 1525, 1526, or MPS-801, -802, or -803 printers. CTRL-P prints with a device number of 4 and a secondary address of 7 (uppercase/lowercase mode).

If CTRL-P doesn't work for you, try another variation, **SHIFT-CTRL-P**. Answer the prompt *Print to: Screen, Disk, Printer?* with the single letter **S**, **D**, or **P**. Press any other key to cancel the command.

If you press P for printer, *SpeedScript* requests two more keystrokes. First, answer the *Device number* prompt with a number from 4 through 7. This lets you print to one of several printers addressed with different device numbers. Next, answer the *Secondary address* prompt with a number from 0 through 9.

Non-Commodore Printers

The secondary address is used on most non-Commodore printer interfaces to control special features. For example, you can bypass the emulation features and use graphics mode to communicate directly with your printer (see the true ASCII command below). Consult the list of secondary addresses in your printer interface manual. *SpeedScript* does not work properly with RS-232 serial printers or interfaces.

An additional note: Some printers and interfaces incorporate an automatic skip-over-perforation feature. The printer skips to the next page when it reaches the bottom of a page. Since *SpeedScript* already controls paper feeding, you need to turn off this automatic skip-over-perf feature (usually, by sending out control codes) before running *SpeedScript*, or paging won't work properly. Remember, sometimes the printer controls the skip-over-perf feature, sometimes the interface, and sometimes even both.

The Commodore 64 version of *SpeedScript* has been tested with the following printers: Commodore 1525 and 1526; MPS-801, -802, and -803; C. Itoh Prowriter 8510; Epson MX-80; Gemini 10-X; Star SG-10, SG-10C, and SD-10; Okimate 10 and 20; Okidata 82 and 92; and Hush-80 CD. *SpeedScript* has also been tested with these printer interfaces: Cardco A/B/G+ and G Wiz, Tymac Connection, Xetec, Turbo-Print, and MW-350. *SpeedScript* should work even if your printer or interface is not on this list. These are just the ones that have been tested.

Be sure your printer or interface supplies its own linefeeds. Again, consult your manuals and insure that either your printer or interface (but not both) supplies an automatic linefeed after carriage return. To test this, print a small sample of text with CTRL-P. Since the default is single-spacing, you should not see double-spacing, nor should all printing appear on the same line. If you still aren't getting linefeeds, use the linefeed command discussed below.

Printing To Screen And Disk

SHIFT–CTRL-P prints to the screen when you press S. The screen colors change to white letters on a black background, and what appears on the screen is exactly what would print on the printer. It takes two screen lines to hold one 80-column printed line, of course. If you use double-spacing (see below), it's much easier to see how each line is printed. With this screen preview, you can see where lines and pages break. To freeze printing, hold down either of the SHIFT keys, or engage SHIFT LOCK. The border color changes to white while SHIFT is held down. When printing is finished, press any key to return to editing.

SHIFT–CTRL-P prints to a disk file when you press D. Enter the filename when it's requested. *SpeedScript* sends out all printer information to a sequential file. You can use other programs to process this formatted file. Try this simple example:

```
10 OPEN 1,4
20 OPEN 2,8,8,"filename"
30 GET#2,A$:SS = ST: PRINT#1,A$;: IF
   SS = 0 THEN 30
40 PRINT#1: CLOSE1
50 CLOSE2
```

This program dumps the disk file specified by the filename in line 20 to any printer. You can use it to print *SpeedScript* files (produced with SHIFT–CTRL-P) on another Commodore computer and printer without running *SpeedScript*. Change line 10 to **OPEN 1,2,0, CHR$(6)** to dump the file to a 300-baud modem or RS-232 printer, or **OPEN 1,3** to display it on the screen.

Formatting Commands

The print-formatting commands must be distinguished from normal text, so they appear onscreen in reverse field with the text and background colors switched. You enter these reverse-video letters by pressing **CTRL-£** (pound sign) or **CTRL-3**, which is easier to type with one hand. Answer the prompt *Enter format key:* by pressing a single key. This key is inserted into text in reverse video. All lettered printer commands should be entered in lowercase (unshifted). During printing, *SpeedScript* treats these characters as printing commands.

There are two kinds of printing commands, which will be called stage 1 and stage 2. Stage 1 commands usually control variables such as left margin and right margin. Most are followed by a number, with no space between the command and the number. Stage 1 commands are executed before a line is printed.

Stage 2 commands, like centering and underlining, are executed while the line is being printed. Usually, stage 1 commands must be

on a line of their own, although you can group several stage 1 commands together on a line. Stage 2 commands are by nature embedded within a line of text. A sample stage 1 line could look like this:

l10r50s2

Embedded stage 2 commands look like this:

cThis line is centered.←
This isuunderliningu.←

Stage 1 Commands

l Left margin. Follow with a number from 0 to 255. Use 0 for no margin. Defaults to 5. See Figure 2 for an illustration of margin settings.

r Right margin position, a number from 1 to 255. Defaults to 75. Be sure the right-margin value is greater than the left-margin value, or *SpeedScript* will become extremely confused. Some printer interfaces force a certain printing width, usually 80 characters wide. You'll need to disable this in order to permit *SpeedScript* to print lines longer than 80 characters.

t Top margin. The position at which the first line of text is printed, relative to the top of the page. Defaults to 5. The header (if any) is always printed on the first line of the page, before the first line of text.

b Bottom margin. The line at which printing stops before continuing to the next page. Standard 8½ × 11-inch paper has 66 lines on most printers (6 vertical lines of text per inch is standard for Commodore printers). Bottom margin defaults to the fifty-eighth line. The footer (if any) is always printed on the last line of the page, after the last line of text.

p Page length. Defaults to 66. If your printer does not print six lines per inch, multiply lines-per-inch by 11 to get the page length. European paper is usually longer than American paper—11⅞ or 12 inches. Try a page length of 69 or 72.

s Spacing. Defaults to single-spacing. Follow with a number from 1 to 255. Use 1 for single-spacing, 2 for double-spacing, and 3 for triple-spacing.

@ Start numbering *at* page

Figure 2: Graphic Representation Of Margin Settings

Values shown are default settings

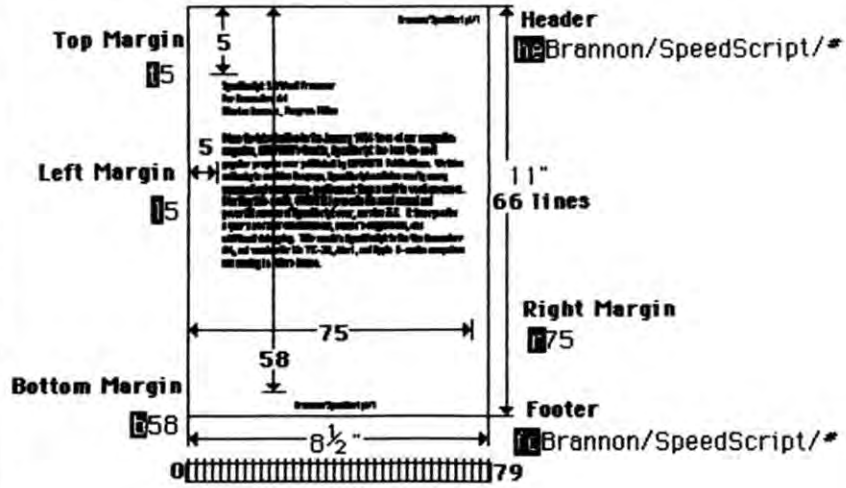


Figure 3: Quick Reference Chart Format (Printer Commands)

Enter these commands with CTRL-£ or CTRL-3:

Command	Description	Default	Command	Description	Default
a	True ASCII	off	n	Next page	
b	Bottom margin	58	p	Page length	66
c	Centering		r	Right margin	75
e	Edge right		s	Spacing	1
f	Footer		t	Top margin	5
g	Link file*		u	Underline	
h	Header		w	Page wait	
i	Information		x	Columns across	80
j	Linefeeds on		@	Initial page #	1
l	Left margin	5	?	Skip pages	
m	Margin release		#	Print page #	

Examples:

hcSpeedScript/#←	Centered Header with page number
l10r70s2←	Left margin 10, right margin 70, double spacing.
g9:SpeedScript.←	Goto and continue printing with filename "SpeedScript."

* new format: gdrive#:filename

number given. Page numbering normally starts with 1.

? Disables printing until selected page number is reached. For example, a value of 3 would start printing the third page of your document. Normally, *SpeedScript* starts printing with the first page.

x Sets the page width, in columns (think *a cross).* Defaults to 80. You need to change this for the sake of the centering command if you are printing in double-width or condensed type, or if you're using a 40-column or wide-carriage printer.

n Forced paging. Normally, *SpeedScript* prints the footer and moves on to the next page only when it has finished a page, but you can force it to continue to the next page by issuing this command. It requires no numbers.

m Margin release. Disables the left margin for the next printed line. Remember that this executes before the line is printed.

a True ASCII. Every character is assigned a number in the ASCII (American Standard Code for Information Interchange) character set. Most printers use this true ASCII standard, but Commodore printers exchange the values for uppercase and lowercase to match Commodore's own variation of ASCII. Some printer interfaces do not translate Commodore ASCII into true ASCII, so you need to use this command to tell *SpeedScript* to translate. Also, you will sometimes want to disable your interface's emulation mode intentionally in order to control special printer features that would otherwise be rejected by emulation. Place this command as the first character in your document, even before the header and footer definitions. Don't follow it with a number.

Since, in effect, the true ASCII command changes the case of all letters, you can type something in lowercase and use true ASCII to make it come out in uppercase.

w Page wait. Like the true ASCII command, this one should be placed at the beginning of your document before any text. With page wait turned on, *SpeedScript* prompts you to *Insert next sheet, press RETURN* when each page is finished printing. Insert the next sheet, line it up with the printhead, and then press RETURN to continue. Page wait is ignored during disk or screen output.

j Select automatic linefeeds after carriage return. Like **a** and **w**, this command must be placed before any text. Don't use this command to achieve double-spacing, but only if all text prints on the same line.

i Information. This works like REM in BASIC. You follow the command with a line of text, up to 255 characters, ending in a return mark. This line will be ignored during printing; it's handy for making notes to yourself such as the filename of the document.

h Header define and enable. The header must be a single line of text ending with a return mark (up to 254 characters). The header prints on the first line of each page. You can include stage 2 commands such as centering and page numbering in a header. You can use a header by itself without a footer. The header and footer should be defined at the top of your document, before any text. If you want to prevent the header from printing on the first page, put a return mark by itself at the top of your document before the header definition.

f Footer define and enable. The footer must be a single line of text ending in a return mark (up to 254 characters). The footer prints on the last line of each page. As with the header, you can include stage 2 printing commands, and you don't need to set the header to use a footer.

g Go to (link) next file. Put this command as the last line in your document. Follow the command with the letter *D* for disk or *T* for tape, then a colon (:), and finally, the name of the file to print next. After the text in memory is printed, the link command loads the next file into memory. You can continue linking in successive files, but don't include a link in the last file. Before you start printing a linked file, make sure the first of the linked files is in memory. When printing is finished, the last file linked to will be in memory.

Stage 2 Commands

These commands either precede a line of text or are embedded within one.

c Centering. Put this at the beginning of a line you want to center. This will center only one line, ending in a return mark. Repeat this command at the beginning of every line you want centered. Centering uses the page-width setting (see above) to center the line properly. To center a double-width line, either set the page width to 40 or pad out the rest of the line with an equal number of spaces. If you use double-width, remember that the spaces preceding the centered text will be double-wide spaces.

e Edge right. This command will cause a line to be aligned with the right margin when it is printed. That is, spaces will be inserted in front of the line so that the last character in the line will be printed at the right margin. Place the command at the beginning of the line you want aligned; it will only affect one line at a time, each ending with a return mark. Repeat this command at the beginning of every line you want aligned to the right. Note that this is *not* the same as *right justification*—a feature found on some word processors that adjusts printing to align both the left and right margins—since the edge-right command aligns only one line, and only at the right margin. *SpeedScript* has no right-justification feature.

When *SpeedScript* encounters this command, it prints the current page number. You usually embed this within a header or footer.

u A simple form of underlining. It does not work on Commodore printers, but only on printers that recognize CHR$(8) as a backspace and CHR$(95) as an underline character. Underlining works on spaces, too. Use the first **u** to start underlining and another one to turn off underlining.

Fonts And Styles

Most dot-matrix printers are capable of more than just printing text at ten characters per inch. The Commodore MPS-801 can print in double-width and reverse field. Some printers have several character sets, with italic and foreign language characters. Most can print in double-width (40 characters per line), condensed (132 characters per line), and in either pica or elite. Other features include programma-

ble characters, programmable tab stops, and graphics modes. Many word processors customize themselves to a particular printer, but *SpeedScript* was purposely designed not to be printer-specific. Instead, *SpeedScript* lets you define your own stage 2 printing commands.

You define a programmable *printkey* by choosing any character that is not already used for other printer commands. The entire uppercase alphabet is available for printkeys, and you can choose letters that are related to their function (like *D* for double-width). You enter these commands like printer commands, by first pressing **CTRL-3**.

To define a printkey, just press CTRL-3, then the key you want to assign as the printkey, then an equal sign (=), and finally the ASCII value to be substituted for the printkey during printing. For example, to define the + key as the letter *z*, you first look up the ASCII value of the letter *z* (in either your printer manual or user's manual). The ASCII value of the letter *z* is 90, so the definition is **+ = 90←**

Now, anywhere you want to print the letter *z*, substitute the printkey:

Gad+ooks! The +oo is +any!←

This will appear on paper as

```
Gadzooks! The zoo is zany!
```

More practically, look up the value of reverse-on and reverse-off. Reverse-on, a value of 18, prints all text in reverse video until canceled by reverse-off (a value of 146) or a carriage return. So, define SHIFT-R as 18 and SHIFT-O as 146. Anywhere you want to print a word in reverse, bracket the word with printkey R and printkey O.

You can similarly define whatever codes your printer uses for features like double-width or emphasized mode. For your convenience, four of the printkeys are predefined, though you can change them. Printkey 1 is defined as a 27, the value of the ESCape code used to precede many two-character printer commands. (With some printer interfaces, you must send two ESCape codes to bypass the interface's emulation.) For example, the Epson command for double strike is ESC-G. You can select it in *SpeedScript* with

1G

Printkey 2, a value of 14, goes into double-width mode on most printers, and printkey 3, a value of 15, turns off double-width on some printers and selects condensed mode on others. Printkey 4 is defined as 18, which selects reverse field with Commodore printers (and on some graphics interfaces in emulation mode) or condensed mode on some other printers.

With so many codes available, you can even design custom logos and symbols using your printer's graphics mode. For example, on the 1525/MPS-801, you can draw a box (perhaps for a checklist) by first setting the appropriate codes:

13444432Toothpaste←

Then display the box with text by typing

1=8 2=25 3=255 4=193←

This appears on paper as

```
□ Toothpaste
```

Keep one thing in mind about printkeys. *SpeedScript* always assumes it is printing to a rather dumb, featureless printer, the least common denominator. *SpeedScript* doesn't understand the intent of a printkey; it just sends its value out. So if you make one word within a line double-width, it may make the line overflow the specified right margin. There's no way for *SpeedScript* to include built-in font and type-style codes without being customized for a particular printer, since no set of codes is universal to all printers.

SpeedScript Mastery

It may take you awhile to fully master *SpeedScript*, but as you do you'll discover many ways to use the editing and formatting commands. For example, there is a simple way to simulate tab stops, say, for a columnar table. Just type a period at every tab-stop position. Erase the line; then restore it multiple times. When you are filling in the table, just use word-left/word-right to jump quickly between the periods. Or you can use the programmable printkeys to embed your printer's own commands for setting and jumping to tab stops.

SpeedScript can also be used as a simple database manager. Type in the information you need; then store it as a *SpeedScript* document. The search feature lets you quickly find information, especially if you use graphics characters to flag key lines. You can search for the graphics characters and quickly skip from field to field.

You don't have to change or define printer commands every time you write. Just save these definitions as a small text file, and load this file each time you write. You can create many custom definition files and have them ready to use on disk. You can create customized "fill in the blank" letters. Just type the letter, and everywhere you'll need to insert something, substitute a graphics symbol. When you're ready to customize the letter, just hunt for each graphics symbol and insert the specific information.

SpeedScript does not work with any 80-column video boards or software 80-column emulators. *SpeedScript* also wipes out most kinds of resident (RAM-loaded) software, including most software-simulated printer drivers. However, you can print to disk using SHIFT–CTRL-P, and then dump the disk file to the printer from BASIC.

File Compatibility

SpeedScript documents are stored as program files (PRG type on disk). Naturally, you can't load and run a *SpeedScript* file from BASIC. The characters are stored in their screen code (POKE) equivalents. Several commercial word processors store text similarly, including *WordPro* 3+ and *PaperClip*. As a matter of fact, two commercial spelling checkers designed for *WordPro* also work with *SpeedScript*: *SpellRight Plus* (from Professional Software) and *SpellPro 64* (from Pro-Line Software). ©

Fontmaker
For Commodore 64
SpeedScript

Charles Brannon

Special fonts add character to any screen display. This article shows how to use custom character sets with any version 3.0 or higher of Commodore 64 SpeedScript.

Writing with a word processor often means staring for hours at a video screen. For word processing, screen clarity is especially vital. It's best to have a good-quality color or monochrome monitor, but a clear, readable character set helps, too. Commodore's built-in character set works well and is especially designed for the low resolution of the average TV. However, it can be improved. Besides, it's just plain fun to use your own custom character set. A custom font personalizes your computer and sets it apart from the crowd. There are many font editor programs to design character sets for use with BASIC, but until now there was no way to use them with *SpeedScript*.

To use the program LOAD and RUN "Fontmaker Boot," which configures the memory for "Fontmaker,"the actual workhorse.

Fontmaker only installs a character set that has been previously created; it has no provisions for creating the custom characters. You can easily define your own fonts with a character editor such as "Ultrafont+".

When you run Fontmaker, it prompts you for the name of the character set you'd like to use. By default, the cursor blinks on the filename SPEED.SET. If you'd like to use a font with that name, just press RETURN. Otherwise, type in a new name, overwriting SPEED.SET. If you want to run *SpeedScript* without a custom set, just type X at the prompt (you don't need to erase SPEED.SET; just enter an X).

The character set you've previously created with a font editor program must be on the same disk as the *SpeedScript* program. Fontmaker looks for *SpeedScript* under the filename SS. Either insert a different filename in line 140 of Fontmaker or rename your copy of *SpeedScript* to SS. Fontmaker loads in *SpeedScript*, bumps up the start of text space (reducing available memory by about 11K), loads the character set into that gap, switches the screen to the new character set, then runs *SpeedScript*.

It's Only Temporary

Fontmaker does not permanently change *SpeedScript* unless you resave the word processor at this point (not recommended). In other words, Fontmaker installs the custom character set only for the current session. If you exit *SpeedScript* by pressing the RESTORE key, type POKE 53272,26 to restore the set before you type RUN to reenter *SpeedScript*.

When designing your custom character set, remember that vertical lines appear thinner and fuzzier than horizontal lines. Notice that every vertical line is doubled on the normal Commodore character set, making characters appear bold. You'll probably want to follow the same rule when designing your own sets. This is not a problem with crisp monochrome monitors. You can use the full 8 × 8 resolution of the character grid to design clean, well-formed characters.

Another guideline for readability is that uppercase characters are of uniform height. All lowercase characters are the same height, except for tall characters such as b, d, f, h, i, k, l, and t, which are the same height as uppercase letters. Normally you'll keep the rightmost column and the lowest row blank to keep characters from running into each other and to reserve room for the lowercase descenders on the g, j, p, q, and y. Naturally, an exception is when you design cursive or script characters that should link together.

You'll also want to customize the punctuation marks and symbols. *SpeedScript* uses the back-arrow symbol as the carriage-return mark. If you don't like to see return-marks, just blank out that character. You can put a tiny dot in the SHIFT-SPACE character to distinguish it from a real space. It can also be convenient to define some of the graphics characters to their printing equivalent on the printer. For example, some graphics characters print as italic or foreign-language characters. Just edit the graphics characters to look like their printing equivalents.

You can also create your own custom cursor. *SpeedScript*'s cursor just alternates between the normal and reverse-video version of whatever character it's sitting on. The last 128 characters of a character set are the reverse-video ones. If you want an underline cursor, just copy the normal set down to the reverse-video area and draw a line through the bottom row of every character. Special characters can even have a unique cursor.

Mail Merge
For *SpeedScript*

Jerry Starling

This productivity booster lets you produce personalized form letters by merging a SpeedScript *word processing document with a file of names and addresses. A disk drive and printer are required.*

Many word processors include a powerful feature known as *mail merge*, which allows you to create personalized form letters. If you're not familiar with a mail merge, here's how it works. First you create a file containing names and addresses (if you operate a business or conduct lots of correspondence, you probably have such a file already). Then you write a form letter with the word processor, leaving special markers in the places where each person's name and address will appear. To perform the mail merge, you tell the word processor which document and address file to use. The program automatically prints out a personalized letter for each person in the file, filling in the special blank spaces with each person's name and address. Although the technique is most often used to print form letters, you can use the same feature to merge any sort of data into a standard form document.

SpeedScript doesn't have a mail merge feature, but you can accomplish the same goal with this program, "Mail Merge." With it, you can create computer-generated mail with a personal touch. You'll also be relieved of much of the tedium of preparing personalized letters for your club, church, or business.

How To Use The Program

Before you can use Mail Merge, it's necessary to understand some basic facts about how it works. To produce personalized form letters, Mail Merge reads two sequential files. The first file is a document created with *SpeedScript* and the second is created by an address or database program. I use a commercial database program, but many programs can do the job. If you don't have a database program, Mail Merge can also create the address file (however, it does not have the ability to edit an existing address file). Another option is to use *SpeedScript* itself to create the address file: Simply type in the data using the format described below; then print the file to disk (note that this file must be *printed*, not saved, to disk). If you also save the address file from *SpeedScript* (using a different filename), you'll have the ability to edit the file.

The address file, of course, contains several items of information for each person. In database terminology, the term *record* is used to refer to each set of information (one person's name, address, city, state, zip code, and so on). Each item within the record is called a *field*. In a simple address file, the first field in the record might contain the person's salutation (Ms., Mr., Dr., or whatever); the second field could contain the person's name; the third, the person's street address; and so forth.

Mail Merge expects the address file to have a very simple record and field structure. Each field within the record ends with a carriage return (character 13) and the end of each record is marked with an extra carriage return. For instance, say that your address file contains records consisting of a name, address, and city for each person. In each record, a single carriage return appears after the name and address, while two carriage returns appear after the city to mark the end of the record. The complete file consists of a number of these records in sequence.

The form letter is written with *SpeedScript* and *printed* (not saved) to disk with *SpeedScript's* SHIFT–CTRL–P command. If you have some other file that creates sequential files, you can probably use those files with Mail Merge, as well.

The form letter must include special markers to show Mail Merge where to insert information from the address file. Each such marker is a number within square brackets ([ ]). When Mail Merge finds a number in square brackets, it reads a field from the address file. The number inside the brackets tells Mail Merge which field from the current record to print at that place in the form letter. Thus, at various places in the letter you might have the markers [1] and [3]. The marker [1] tells Mail Merge to insert field 1 at that place in the letter. The marker [3] tells the program to insert field 3 at the place where that marker appears. Note that the markers can appear in any order (marker [3] can appear before or after marker [1], and so on) and you can use the same marker as often as you want. For instance, you might want to print the person's name at several different places in the letter.

Another special indicator must appear as the very first line in the form letter. This marker tells Mail Merge how many fields each record contains. It consists of a left square

bracket followed by the number of fields and a lowercase *v* character. For example, this indicator tells Mail Merge that each record in the address file contains five fields:

[5v

Mail Merge ignores everything in the document which appears *before* this indicator, so it also serves as a "start printing" command for the program. Mail Merge stores this value in a string array which it DIMensions with 25 elements in line 120. This means you can merge an address file whose records contain as many as 25 fields. To merge files with more fields, increase the value 25 in line 120 as needed.

A Walk Through

Here's a brief description of the prompts you encounter when using Mail Merge.

• *Enter name of letter file.* At this prompt, Mail Merge expects you to enter the name of the *SpeedScript* document (form letter) or other sequential form file which you previously printed to disk.

• *Enter name of list file.* Enter the name of the address file. If you have not already created an address file, enter a new filename for the file you are about to create.

• *Create file or input from disk?* Enter C to create a new address file or I to use a file that already exists. If you use an existing file, you'll skip ahead to the *Enter date* prompt (see below).

• *How many variables per letter?* This prompt appears only when you are creating a new address file. When it creates the file, Mail Merge needs to know how many fields (variables) are contained in a record. Enter that number (it should be the same as the number in the indicator at the beginning of the *SpeedScript* form letter).

• *Enter name for variable.* This name is used as a prompt while you are entering address file data. It will be repeated for each of the fields in the record.

• *Enter END when all entries are entered.* This is the data entry portion of Mail Merge. The program prompts you with the names you entered in the last step, storing the data you enter for the fields in each record. When you've finished entering all the data, enter END to terminate this section of the program.

• *Enter date for letter (M/D/Y)?* Mail Merge can insert a date wherever the form letter has the special indicator **[date]**. You must always enter a date, even if your form letter doesn't use it. Enter the date in the format M/D/Y. The year can be entered as either two digits or four digits. If you enter two digits, Mail Merge adds 19 in front of the digits you enter. For example, the year 1986 can be entered as either 86 or 1986. The month and day can be entered as either one or two digits. Enter 2 for the month of February, 12 for December, and so on. When it prints the letter, Mail Merge converts that date to the usual written format (December 24, 1986). The year can be entered as two digits (which assumes 19*nn*), or as four digits. The date is checked for validity, but February 29 is not considered a valid date by Mail Merge.

• *Press RETURN when ready to input letter.* This prompt indicates that Mail Merge is ready to store the form letter in memory. Insert the disk containing the form-letter file and press RETURN. Mail Merge reads the document, examining each line for special Mail Merge markers. As it reads each line, Mail Merge prints a period (.) on the screen. This process takes longer than simply loading the document, so be patient.

• *Continuous form or single sheet feed?* Mail Merge gives you the option of using continuous form (fanfold) paper or feeding sheets in one at a time. Press C for continuous form or S for single sheet paper. Note that letters which exceed one page in length cannot be used with the single sheet option, since Mail Merge makes no provision for pausing except at the end of the document.

• *Press RETURN when ready to begin printing.* At this point Mail Merge is ready to print the final letters. Insert the disk containing the address file and make sure the printer is turned on. When the system is ready to print, press RETURN. Mail Merge proceeds to print a letter for each record in the address file.

• *Press RETURN when ready.* This prompt appears only if you select the single sheet paper option. It alerts you when it's time to insert each new sheet of paper.

• *End of Job.* Mail Merge has finished printing all the letters. ©

Programming Notes

Here's a description of the various sections of Mail Merge.

Lines	Notes
100–110	Opening screen display; set maximum size for arrays for number document lines and number variables in list file.
120	Dimension arrays; read data for date conversion routine.
130–160	Input file information.
170–270	Create list file (optional).
280–300	Date formatting.
330–380	Read number of variables list file will contain.
390–500	Read document from; flag lines containing variables.
510–530	Select continuous or single sheet paper feed.
540–550	Request list file disk.
560	Open list file and printer.
570	Read a variable set from the list file; initialize the document line counter.
580	Check for a variable flag in the document line.
590	If no variable is found, print the line as is.
600–690	If a variable is found in the document line, insert variables and print the line.
700	Update document line counter; process next line.
710–720	End of document routine (entered from line 600). Eject page or wait for new sheet. Check for end of job. If not at end of job, read next variable set for next list file.
770–780	Wait for RETURN from keyboard.
790–810	Check for errors in input.
820–840	Data for date conversion routine.
850–880	Input list file variable set and print to screen.
890–910	Break down date elements from M/D/Y format.
920–980	Error message for improper document preparation.

ScriptSave
Automatic Disk Saves
For Commodore 64
SpeedScript 3.0

J. Blake Lambert

Have you ever watched your computer suddenly blink off due to an unexpected power failure, and then realized that you haven't saved your text for an hour or more? All that work down the drain. But with "ScriptSave" these accidents won't be quite so disastrous. The machine language program is designed to work with a Commodore 64, a disk drive, and the SpeedScript *word processor.*

While you are working with a computer, you're tethered to a lifeline. That lifeline is the computer's power cord. If the lifeline is disconnected or interrupted for even a brief moment, your computer suffers an attack of amnesia. Random Access Memory (RAM) chips need a constant flow of electricity to maintain their information—the information you put into the computer. Usually a power failure does not damage the computer, but it does obliterate the program or text you were working on.

Luckily, most people live in areas with reliable power sources. However, electrical service in some locales is subject to frequent interruptions. And sometimes your wayward foot, a passer-by, a small child, or even a pet can accidentally knock a power cord loose. A split-second is all it takes for the computer to forget.

Unfortunately, the writer is often forgetful, too. To protect yourself against power interruptions, you should periodically save your work on disk. But when you're working intensely, it's easy to forget this important duty. If the power does fail, you can generally remember where you left off, but it's often impossible to remember how you got there. Even if you frequently rewrite your documents, losing any of the intermediate versions interferes with the creative process.

An Extra Rope

"ScriptSave" is the solution. ScriptSave is a short (less than 256-byte) utility that ties into the Commodore 64 version of *SpeedScript 3.0*. Every ten minutes, it waits for you to finish the paragraph you're working on, and then automatically saves your text (except for the final return character) on disk with a special filename. That way, if a power failure unexpectedly strikes, you can later recover all but the last few minutes of your work.

ScriptSave is a BASIC loader and boot program: It loads and executes *both* the machine language automatic-save routine and *SpeedScript 3.0*. Make sure both programs are on the same disk, and change the filename in line 30 of ScriptSave to the filename for *SpeedScript 3.0* as stored on your disk. Generally, it is best to start with a blank disk and place ScriptSave on the disk first. This way, you can use LOAD"*",8: followed by SHIFT–RUN/STOP to boot up for a writing session.

Once you load and run ScriptSave, this prompt should appear:

File:

Type in a legal Commodore filename, but limit it to 14 characters or less. Press RETURN. ScriptSave automatically loads and runs *SpeedScript 3.0*. Now you can start writing and stop worrying about periodic saves.

When ScriptSave stores your work, it precedes the filename you specified with a two-digit version number. For example, if you choose the filename ARTICLE, the first version will be called 01ARTICLE, the second version 02ARTICLE, and so on.

Of course, you can still save manually anytime you wish. *SpeedScript 3.0* functions normally except for one detail—it assumes that all your saves are on disk. You no longer have to press T or D to specify Tape or Disk after selecting the f8 SAVE option.

Additional Notes

When you finish writing, you will probably want to save a final version of your text. Later, if you want to scratch the intermediate versions off your disk, there's a quick method using *SpeedScript*'s disk commands. First, press CTRL–↑. When *SpeedScript* prompts Disk Command:, type s:??*article* and press RETURN (substitute for *article* the filename you specified in ScriptSave).

There's another trick you can use to give yourself more time between saves or to force an early save. Since ScriptSave uses the internal time-of-day clock, you can

DISCOUNT SOFTWARE

ABACUS SOFTWARE

ADA Training Course	$25.95
Anatomy of the 1541 or C-64	ea. 9.95
Assembler Monitor	25.95
Basic Compiler	25.95
Basic 128	39.95
Cadpak	25.95
Cadpak 128	39.95
Chartpak 64 or 128	ea. 25.95
Cobol	25.95
Cobol 128	39.95
Forth Language	25.95
Personal Portfolio Manager	25.95
Personal Porfolio Manager-128	39.95
Super C 64 or 128	ea. 39.95
Super Pascal 64 or 128	ea. 39.95
TAS 64 or 128	ea. 39.95
XREF 64 or 128	ea. 14.00

ACCESS

Leader Board	$25.95
L.B. Tournament #1	12.95
World Class L.B.	25.95
W.C. Famous Courses #1	12.95
Mach 5	22.95
Mach 128	32.95
Triple Pack	12.95
10th Frame Bowling	25.95
Echelon	25.95

ACCOLADE

Ace of Aces	$19.95
Deceptor	9.95
Desert Fox	9.95
Fight Night	19.95
Hardball	19.95
PSI-5 Trading Company	19.95

ACTIONSOFT

Thunder Chopper	$19.95
Up Periscope	19.95

ACTIVISION

Aliens	$23.49
Gamemaker	26.49
Gamemaker Library Disks	
Sports	13.49
Science	13.49
Ghostbusters	9.95
Hacker	9.95
Little Computer People	9.95
Murder on the Mississippi	23.49
Music Studio	19.95
Portal	26.49
Tass Times	19.95

BERKELEY SOFTWORKS

GEOS	$39.95
GEOS 128	49.00
Desk Pak I	23.95
Desk Pak II	33.95
Font Pak I	19.95
GEO Calc	33.95
GEO Chart	26.49
GEO Dex	26.49
GEO File	33.95
Writer's Workshop	33.95

BASEMENT BOYS

Fast Hack'em	$29.00

BRODERBUND SOFTWARE

Carmen Sandiego U.S.A.	$29.95
Carmen Sandiego World	23.49
Karateka	19.95
Print Shop	29.95
Print Shop Companion	23.49
P.S. Graphics Library I, II or III	ea. 16.95
Toy Shop	39.95
Syncalc	33.95
Syncalc Templates	13.49
Holiday P.S. Graphics	16.95

CENTRAL POINT

Copy II 64/128	$29.00

CAPCOM

Ghosts 'n Goblins	$19.95
Gunsmoke	19.95
Trojan	19.95
1942	19.95

DATA EAST

Break Thru	$23.49
Ikari Warrior	23.49
Karate Champ	13.49
Kung Fu Master	13.49
Tag Team Wrestling	23.49

DAVIDSON

Alge-Blaster	33.95
Spell-It	33.95
Word Attack	33.95
Math Blaster	33.95

DIGITAL SOLUTIONS, INC.

Super Pack II	65.95
Pocket Filer II	39.95
Pocket Planner II	39.95
Pocker Writer II	39.95
Pocket Dictionary	9.95

ELECTRONIC ARTS

Alternate Reality "The Dungeon"	$26.49
Arctic Fox	23.49
Auto Duel	33.95
Bard's Tale	27.95
Bard's Tale II	27.95
Hint Book Bard's Tale I or II	ea. 9.95
Chessmaster 2000	27.95
Dan Dare	14.49
Earth Orbit Station	20.95
Legacy of the Ancients	20.95
Marble Madness	20.95
Moebius	27.95
Movie Maker	10.95
One on One	10.95
Touchdown Football	10.95
Ultima IV	42.00

EPYX

Championship Wrestling	$26.49
Create A Calendar	19.95
Temple of Apshai Trilogy	13.49
Street Sports Baseball	26.49
Street Sports Basketball	26.49
Sub Battle	26.49
Winter Games	26.49
World Games	26.49

FIREBIRD

Elite	$22.95
Golden Path	26.49
Sentry	26.49
Starglider	26.49
Tracker	26.49
Pawn	29.95

GAMESTAR

Championship Baseball	$19.95
G.F.L. Football	23.49
On-Court Tennis	19.95
Star Rank Boxing	19.95
GBA Championship Basketball	23.49

H & P

Final Cartridge	$44.95

MICROPROSE

Conflict in Vietnam	$26.49
F-15 Strike Eagle	23.49
Gunship	23.49
Silent Service	23.49
Top Gunner Series	16.49
Stealth Fighter	CALL
Pirates	26.49

MINDSCAPE

American Challenge	$19.95
Bop 'n Wrestle	19.95
Defender of the Crown	23.49
Fist	19.95
Indoor Sports	19.95
Infiltrator	19.95
Super Star Hockey	23.49
Trail Blazer	19.95

ROGER WAGNER

Merlin 64	$33.95
Merlin 128	48.00

SPRINGBOARD

Certificate Maker	$33.95
C.M. Library #1	23.49
Clip Art Vol. #1 or 3	ea. 19.95
Clip Art Vol. #2	26.49
Graphics Expander	23.49
Newsroom	33.95

SSI

B-24	$26.49
Battlecruiser	39.95
Kampfgruppe	39.95
Phantasie III	26.49
President Elect '88	16.49
Realms of Darkness	26.49
Rebel Charge: Chickamauga	33.95
War in the South Pacific	39.95

SUBLOGIC

Jet	29.00
Flight Simulator II	35.95
F.S. Scenery Disk 1-7	ea. 14.50
Star Scenery - San Francisco	14.50
Star Scenery - Japan	14.50
Western Scenery Disk Pack	74.95

TIMEWORKS

Partner	$39.95
Sideways	19.95
Swiftcalc w/Sideways	26.49
Word Writer III	33.95
Data Manager 128	49.00

UNICORN

Decimal Dungeon	$19.95
Fraction Action	19.95
Percentage Panic	19.95
Ship Ahoy	19.95
Ten Little Robots	19.95

XETEC, INC.

Fontmaster II	$33.95
Fontmaster 128	39.95

SAVE SAVE

ABBY'S SUPER BUYS

INFOCOM
COMMODORE
& MASTERTRONICS

Buy Four - Get One FREE

Zork II	$4.99	Vegas Jackpot	4.99
Zork III	4.99	Golden Talisman	4.99
Suspended	4.99	Quest For Holy Grail	4.99
Starcross	4.99	1985 - The Day After	4.99
Deadline	4.99	Ant Attack	4.99
Captive	4.99	Challenger	4.99
Big Mac	4.99	Spooks	4.99
Finders Keepers	4.99		

WE CARRY A FULL LINE OF SOFTWARE & ACCESSORIES — PLEASE CALL

ACCESSORIES

VIC 1750 Expansion	CALL
VIC 1764 Expansion	CALL
VIC 1351 Mouse	CALL
Avatex Modems	CALL
Xetec Super Graphics	$59.00
Xetec Super Graphics Jr.	35.95
Wico Boss Joystick	15.00
Epyx 500XJ Joystick	15.00
Disk Notchers	4.50

NASHUA DISKS

DS/DD 5¼" With Sleeves

Per Hundred	$ 39
Per Thousand	$325

RIBBONS

Commodore 1525	$7.50
Commodore 1526/802	7.00
Commodore MPS-801	7.50
Commodore MPS-803	9.00
Epson MX/FX/RX-80	ea. 5.00
Panasonic 1080/1091/1092	ea. 8.50
Star Gemini Black	2.50
Star Gemini Color*	3.50
*Please State Choice of Color	
Stan NX-10, NP-10, NL-10	ea. 8.00
Okimate 120	6.50
Okimate 10/20 Black	4.50
Okimate 10/20 Color	5.00

DUST COVERS

C-128 Computer	$7.00
Vic 20/64 Computer	7.00
1571 Disk Drive	7.00
1541 Disk Drive	7.00
1525 Printer	9.00
1526 Printer	9.00
MPS 1000 W or W/O Tractor	ea. 9.00
MPS 801/802 Printer	ea. 9.00
MPS 803 W or W/O Tractor	ea. 7.00
Amiga Computer	9.00
Amiga Keyboard	$ 6.50

BOOKS

1541 Repair & Maintenance	$15.00
Anatomy of the C-64	15.00
Anatomy of the 1541	15.00
GEOS Inside & Out	15.00
GEOS Tricks & Tips	15.00
Peeks & Pokes for the C-64	12.00
Tricks & Tips for the C-64	15.00
1571 Internals	15.00
C-128 Basic 7.0 Internals	18.95
C-128 Tricks & Tips	15.00
Peeks & Pokes for C-128	12.50
C-128 CP/M User's Guide	15.00
C-128 Internals	15.00

ALL TITLES ON DISK UNLESS MARKED (R) FOR ROM CARTRIDGE & (C) FOR CASSETTE TAPE

Order Line
1-800-282-0333
ORDERS ONLY! Sales Personnel Do Not Have Information on Previous Orders or Product Specs.

WE CHECK FOR STOLEN CHARGE CARDS
M-F 10 a.m.-8 p.m.; Sat. 10-3 ET
37 S. Broad Street, Fairborn, OH 45324

Customer Service & Ohio Residents
1-513-879-9699

DISCOVER

MasterCard

VISA

SOFTWARE ONLY — Prepaid orders over $50 receive free shipping via UPS in continental U.S. Please add $2 orders under $50. HARDWARE and all orders requiring shipment via U.S. Post Office are subject to additional freight charges. C.O.D. orders under $100.00 accepted - add $5. Charge card orders add $2 service charge. Ohio residents add 6% sales tax. Personal checks require a three-week waiting period. No waiting when paid by credit card, certified check or money order. All items subject to availability and price change. PLEASE CITE AD NUMBER WHEN ORDERING. ALL RETURNS MUST BE ACCOMPANIED BY AN AUTHORIZATION NUMBER.

AD #CP-BEST

exit *SpeedScript 3.0* by tapping RESTORE and pressing the Y key, and then POKE 56330,0 to reset the timer and delay the save. Or you can POKE 56330,16 to set the timer for an immediate save, which will be activated the next time you press RETURN while in *SpeedScript*. You can toggle ScriptSave off and on by entering SYS 52993. Each time ScriptSave is toggled on, it resets the version number to 01 and prompts you to enter a new filename. All of these commands (except for toggling ScriptSave on) should be followed by RUN to reenter *SpeedScript*. One caution, however: While these manipulations are usually safe, there is a chance that exiting and reentering *SpeedScript* will erase your text. C

SpeedView

An 80-Column Preview Program For *SpeedScript*

Mark Schreiner

"SpeedView," an enhancement for COMPUTE!'s Commodore 64 SpeedScript *word processor (versions 3.0 and higher), provides 80-column previewing of text before the actual printing. Using 4 × 3 high-resolution blocks, the program shows exactly how each page of the document will look when printed.*

SpeedScript, COMPUTE!'s popular word processor, becomes an even more useful resource with this program. "SpeedView" lets you preview your text file in 80 columns so you can see precisely how a printout of your document will look.

To use SpeedView, load *SpeedScript* but *do not run it yet*. Next, load SpeedView with the command LOAD"SPEEDVIEW",8,1. Finally, type SYS 9480 and press RETURN. This command activates *SpeedScript* with the SpeedView enhancement.

Two For One

If you plan to use SpeedView regularly, you have the option of creating a combined file which contains both *SpeedScript* and SpeedView. Like *SpeedScript*, the unified file loads and runs just like an ordinary BASIC program. To create this file, reset the computer by turning it off and on; then enter this command and press RETURN:

```
POKE 44,48:POKE 48*256,0:NEW
```

Load SpeedView.Unify into memory and insert a disk containing copies of both *SpeedScript* and SpeedView. When you run SpeedView.Unify, it asks you to enter the names of the *SpeedScript* and SpeedView files on the disk in the drive. After you've entered these filenames, you are prompted to enter a name for the new, combined file. Speedview.Unify reads both files into memory, modifies the BASIC portion of *SpeedScript*, then writes the unified package back to disk using the filename you selected.

To use this combined program, load and run it as you would any BASIC program. You now have a copy of *SpeedScript* with SpeedView permanently installed.

SpeedView Operation

To use SpeedView, press CTRL-SHIFT-P, followed by S, the command sequence to direct *SpeedScript*'s output to the screen. Instead of the usual jumble of scrolling lines, SpeedView presents a neatly formatted representation of the document's first page. Press RETURN to view succeeding pages. SpeedView shows exactly how each page will look when printed on paper. When the last page has been displayed, press any key to return to *SpeedScript*. The screen preview option is the only *SpeedScript* command changed by SpeedView. All others function normally.

You should never press RESTORE while previewing a document with SpeedView. If you must exit *SpeedScript*, press RETURN until you have reached the end of the SpeedView display and return to *SpeedScript*; then press RESTORE to exit. If you do not use the combined file, remember that the proper sequence is to load *SpeedScript*, load SpeedView, then activate SpeedView with the SYS command described above.

Some printkey values may not show up during the 80-column preview. SpeedView displays only those characters whose Commodore ASCII values are in the range of 32–90, or 193–218, inclusive. This includes the upper and lowercase alphabets, numerals, and punctuation marks. The SpeedView program code occupies 1.5K of space that's otherwise available for text memory. As a result, you may not be able to preview a very long document without breaking it into two smaller files. SpeedView behaves erratically if you change the page length to any value other than 66, or the right margin to any value greater than 80. To obtain the best results, make sure to use those settings. C

SpeedCalc
For Commodore 64 And 128

Kevin Martin

Written completely in high-speed machine language, SpeedCalc *has all the important features you'd expect from a commercial spreadsheet program. In addition, its data files can be merged into text files created with the* SpeedScript 3.2 *word processor.*

Have you ever planned a budget for your home or office? If so, you probably used some sort of worksheet divided into rows and columns. Perhaps you wrote the months of the year along the top of the sheet and listed categories for earnings and expenses along one side. After entering data for each category and month of the year, you could calculate total income figures by adding or subtracting numbers in each of the sheet's "cells."

That's a classic example of a worksheet. It lets you enter and organize data, then perform calculations that produce new information. A *spreadsheet* program is an electronic version of the familiar paper worksheet. Since it does all the calculations for you at lightning speed, an electronic spreadsheet is far more convenient than its paper counterpart. And spreadsheet programs also offer built-in editing features that let you enter and manipulate large amounts of data with a minimum of effort.

SpeedCalc is an all machine language spreadsheet program for the Commodore 64. Though relatively compact in size, it's fast and easy to use, and has many of the features found in commercial spreadsheet programs. Even better, the "*SpeedScript* Integrator" program (also included here) lets you merge your *SpeedCalc* files into word processing documents created with *SpeedScript*, COMPUTE!'s popular word processor. Working together, *SpeedCalc* and *SpeedScript* make a powerful team. You can merge a chart of sales figures into a company report, create a table of scientific data for a term paper, and manipulate numeric information in many other ways. In a sense, a spreadsheet program brings to arithmetic all of the flexibility and power that a word processor brings to writing.

The *SpeedCalc* Screen

To use the program, simply enter LOAD"SPEEDCALC",8, then type RUN.

SpeedCalc uses the top line of the screen as the *command line*. This is where *SpeedCalc* displays messages and asks you questions.

Screen lines 2–4 are the *input buffer* area. This is the work area where you enter and edit data. As you'll see in a moment, the input buffer also displays the data contained in the current cell.

The lower 21 screen lines are your window into the spreadsheet. Though the spreadsheet contains many rows and columns, only a few can fit on the screen at one time. By scrolling the screen back and forth with the cursor, you can move the display window to any part of the spreadsheet.

The *SpeedCalc* worksheet consists of 50 vertical columns labeled with letters (AA, AB ... BX) and 200 horizontal rows numbered from 1–200. The rectangle where a row and column intersect is called a *cell*. Cells are where you store data. With 50 columns and 200 rows, the *SpeedCalc* spreadsheet has a maximum of 10,000 (50*200) cells. Due to memory limitations, however, only about a third of these can actually contain data. But you may spread out the data over all 10,000 cells if necessary, depending on what format you need.

If you don't like the spreadsheet's screen colors, they're easily changed with the special function keys. Press the f1 key to cycle through the 16 border colors until you find one you like. The f3 key changes the background color and f5 changes the character color.

Moving The Cursor

Each cell is identified with the letters of its column and the number of its row. For example, the cell at the extreme upper-left corner of the sheet is called AA1, since it's in column AA and row 1. The cell below that is AA2. Moving one cell to the right from AA2 puts you in cell AB2, and so on. (For the sake of clarity, this article uses uppercase letters for cell names. Note, however, that you must use lowercase letters such as aa1 when entering cell names within *SpeedCalc*.)

Your current position in the spreadsheet is shown by the highlighted cursor. The simplest way to move around the sheet is with the cursor keys, which work just as they do in BASIC. Another way to move the cursor is with the HOME key (press CLR/HOME *without* pressing SHIFT). Press HOME once to "home" the cursor on the current screen: The cursor moves to the upper-left cell. Press HOME twice in succession to move the cursor to cell AA1, the home position for the entire sheet.

SpeedCalc also has a *goto* com-

mand for moving over long distances. Press CTRL-G (hold down CTRL and press G). The command line turns blue and displays GOTO: followed by an underline cursor. The underline cursor generally indicates that *SpeedCalc* is waiting for data—in this case it expects the name of the cell where you wish to go. If you enter **ba188** at this point, *SpeedCalc* moves the cursor to cell BA188, adjusting the screen window as needed. Take a few moments to practice moving around the spreadsheet with all three methods: You'll be using them a lot. In a later section, we'll discuss how to change the size and format of a cell.

Keyboard Commands

SpeedCalc offers many different commands, a few of which are entered by pressing one key. However, most commands are entered by pressing CTRL along with another key. CTRL-G, as you've seen, is the goto command. CTRL-A displays the amount of free memory available, and so on. The most drastic command is CTRL-X, which exits *SpeedCalc* and returns you to BASIC. Since leaving the program effectively erases all data in memory, *SpeedCalc* prompts you with ARE YOU SURE Y/N? before it shuts down. To cancel the command, simply type N and press RETURN.

A few commands require you to press three keys at once. This sounds more awkward than it is in practice, since two of the three keys are SHIFT and CTRL. For instance, the *recalculate* command is performed by pressing SHIFT-CTRL-R (hold down SHIFT and CTRL, then press R). The accompanying table lists all the *SpeedCalc* commands. We'll be discussing each command in more detail below.

Three Data Types

Before entering any data, you must know what kind of data *SpeedCalc* accepts. There are three different types: numbers, text, and formulas. Let's look at each type in turn:

1. Numeric data consists of numbers—the basic stuff that spreadsheets work with. *SpeedCalc* has a few simple rules for numeric data: A number must be a decimal value (base 10, not hexadecimal) composed of one or more digits from 0–9, with an optional plus or minus sign. A decimal point is also optional. If you include any other characters in numeric input, *SpeedCalc* treats the entire input as text data (as explained below). Thus, the numbers 123, .001, and −65535 are valid numeric data. The numbers 65,535 (which includes a comma) and 358E6 (scientific notation) are not.

For example, let's enter the number 123 in cell AA1. No special commands are required to enter data: Just move the cursor to AA1 and begin typing. While you're entering the number, it appears only in the input buffer near the top of the screen. As soon as you press RETURN, the number appears in AA1 and the letter N appears at the upper right of the screen. The N signifies *numeric*, meaning that *SpeedCalc* has accepted the entry as valid numeric data. Move the cursor to a vacant cell, then move it back to AA1. The input buffer displays whatever data is found in the cell under the cursor. When the current cell is empty, the buffer is empty as well.

As you can see, pressing RETURN enters a data item into the current cell. You can also end the input by pressing a cursor key. The data is entered as if you had pressed RETURN, and the cursor moves in the indicated direction. This feature is handy for entering a lot of data: Simply type the entry, move the cursor to the next cell, enter more data, and so on.

2. Text data is not "data" in the strict sense, since *SpeedCalc* doesn't use it in calculations as it does numbers and formulas. Text data is there only to help humans understand what the other data means. Text may consist of comments, titles, column headings, subheadings, or whatever you need to interpret the numbers and formulas. As an example, move the cursor to cell AA2 (just under AA1) and type the following line. Note that both uppercase and lowercase letters are acceptable:

```
This is a piece of text data.
```

You can use the DEL key to erase mistakes while you're typing. When you press RETURN, *SpeedCalc* displays T (for text) in the upper-right corner. In this example, the cell isn't long enough to accept all the text, so only the leftmost portion appears in AA2. But even though you can't see it, all of the text is there. Move the cursor to another cell, then move it back to AA2. As soon as you return to AA2, *SpeedCalc* displays all the text in the input buffer.

3. Formula data is a mathematical expression or formula. It may be as simple as 2+2 or as complex as your imagination (and mathematical prowess) allows. The first character in a formula must always be an equal sign (=). If you omit this symbol, *SpeedCalc* either signals an error or treats the data as text. The true power of a spreadsheet is that a formula in one cell can refer to another cell. This is easier to demonstrate than to explain. Move the cursor to cell AA3 and type the following line:

```
=aa1*25.01+@sqr(4)
```

As soon as you press RETURN, *SpeedCalc* displays F (for formula) in the upper-right corner and puts the *result* of the formula (not the formula itself) in AA3. If AA1 contains 123, the value 3078.23 appears in AA3. In plain English, this formula means "multiply the contents of cell AA1 by 25.01 and add the square root of 4." Before we examine the formula more closely, here's a quick demonstration of what makes a spreadsheet such a powerful tool. Move the cursor back to AA1 and press CTRL-R. The command line displays the message RECALCULATION IS ON, meaning *SpeedCalc* now automatically recalculates the entire sheet whenever you make a change. Now change the number in AA1 to 456 (simply move to the cell and start typing). The new result (11406.53) automatically appears in cell AA3. We'll explain more about automatic recalculation later.

When you enter the name of another cell in a formula, the letters must be lowercase (enter **aa1**, not **AA1**). The referenced cell must contain data that *SpeedCalc* can evaluate: a number or another formula. If the formula refers to an empty cell, or one that contains text, *SpeedCalc* signals an error.

Mathematical Operators

These symbols can be used as *operators* in a formula:

Operator	Function
+	addition
−	subtraction
*	multiplication
/	division
↑ (up arrow)	exponentiation
=	equality

One factor that affects formulas is *precedence*, or the order in which operations are performed. In *SpeedCalc*, formula operators have the same precedence as in ordinary math (BASIC, as you may know, uses a somewhat different precedence).

The first operators to be evaluated—those with the highest precedence—are those enclosed in parentheses. Where one set of parentheses encloses another, the expression in the innermost set is evaluated first. The next operators to be evaluated are exponents. Multiplication and division have equal precedence; both operations are lower than exponentiation. Addition and subtraction have the lowest precedence of all. To take one example, *SpeedCalc* evaluates the formula =5*(8+3*−2)↑2−10/+2 as the value 15, just as in ordinary math. Note how the result is affected by the plus and minus signs before the two 2s.

Functions

Formulas may also include any of the functions listed here:

@abs( )	absolute value
@atn( )	arctangent
@ave( )	average of a block of cells
@cos( )	cosine
@exp( )	value of *e* (2.7182318...)
@int( )	integer
@log( )	natural logarithm
@sgn( )	sign
@sin( )	sine
@sqr( )	square root
@sum( )	sum of a block of cells
@tan( )	tangent
pi	value of pi (3.14159265)

All the functions except pi begin with the @ symbol and are followed by parentheses. Within the parentheses of a function you may use a number or formula. For example, the formula =@sqr(4) generates the square root of 4. The formula =@sqr(aa1) returns the square root of whatever value cell AA1 contains. Note that the argument (value within parentheses) of the functions @tan( ), @sin( ) and @cos( ) must be expressed in radians; the result of the function @arc( ) is expressed in radians. The function @int( ) generates an integer (whole number) by truncating (discarding the fractional part of) a numeric value; note that this is different from rounding.

The function @ave( ) calculates the average of the values in a block (group) of cells. The function @sum( ) calculates the sum of a block. Both functions require that you define the block so that *SpeedCalc* knows which cells to include in the calculation. This is done by putting two cell names separated by a colon in the parentheses. The first cell name defines the upper-left corner of the block, and the second defines the bottom-right corner. For instance, @ave(aa1:ad20) calculates the average of all the cells from AA1 to AD20. The function @sum(aa1:ad20) calculates the sum of AA1 through AD20, and so on. An error results if any cell in the block is blank or contains text data.

Editing The Sheet

Editing is a very important spreadsheet function. The simplest way to change what a cell contains is to move to it and start typing. The old data in that cell is replaced by whatever you enter. For instance, to replace the contents of cell AA1 with the number 456, move to that cell, type 456, and press RETURN or exit with a cursor key. Press CTRL-B (think of *blank*) to erase what's in the current cell. You can also clear a cell by typing a space and pressing RETURN. To erase everything in the sheet, press SHIFT–CLR/ HOME. Before carrying out this drastic operation, *SpeedCalc* asks you to confirm it by pressing Y or N.

In some cases, only a minor change is needed. *Edit mode* lets you change the data in a cell without retyping the entire entry. To activate edit mode, move to the desired cell and press CTRL-E. In this mode, the up/down cursor key is disabled, and the left/right cursor key moves within the input buffer. Erase unwanted characters with the DEL key. Typing in edit mode inserts new characters in the line: Everything to the right of the new character moves right one space (unless the buffer is already full). Since the cursor keys have a different function in edit mode, you cannot use them to end the input. Press RETURN to enter the new data and escape from edit mode.

As you may have learned already, *SpeedCalc* displays *ERROR* in a cell when you enter an erroneous formula. The usual cause is that you have made a typing error in that cell, or the formula refers to text or an empty cell. A line of asterisks (*********) signals that a number is too large to be printed in the cell. Though these messages appear in the cell area, no data is lost. You may move to the affected cell, view its contents in the input buffer and make whatever correction is needed.

Recalculation

This feature is the very core of *SpeedCalc*. As you know, entering or editing a piece of data causes *SpeedCalc* to perform a calculation and put the result in the cell under the cursor. In most cases, the new data relates to data in other cells, so you'll ultimately want to recalculate the entire spreadsheet as well. This can be done in two different ways: manually or automatically.

To recalculate the spreadsheet manually, press the back arrow key (←, at upper-left on the keyboard). *SpeedCalc* begins at AA1 and recalculates every cell that contains data, placing fresh results wherever needed. If you switch to automatic recalculation mode, *SpeedCalc* automatically recalculates the entire spreadsheet each time you enter new data or edit what exists. When you press CTRL-R, *SpeedCalc* changes the recalculation status and displays it at the top of the screen. If automatic recalculation was turned off before, it is now on (and vice versa). If you aren't sure which mode you're in, press SHIFT-CTRL-R; *SpeedCalc* displays the recalculation mode without changing it.

Automatic recalculation can be fun to watch in a large spreadsheet: Every time you make a change, new results ripple all the way down the screen. However, the more data your spreadsheet contains, the

longer it takes to update the entire sheet. For this reason, you may want to turn automatic recalculation off most of the time, recalculating with the back arrow key whenever you need to view results.

One problem with recalculation arises from the order in which cells are calculated. Because only one cell can be calculated at a time, you must sometimes recalculate the entire spreadsheet two or three times to get correct results in every cell (this is common to all spreadsheet programs). For instance, say you have a formula in AA1 which refers to a formula in AB15. When *SpeedCalc* calculates AA1, it must use the existing data from AB15—which is probably out of date, since the formula in AB15 hasn't been recalculated yet. To avoid this problem, you should always press the back arrow key two or three times before printing a spreadsheet or saving it to disk.

SpeedCalc offers a number of other features. Before experimenting with them, you should spend some time typing in a hypothetical spreadsheet—perhaps a fictitious yearly budget—to become thoroughly familiar with the basic commands covered so far. Most importantly, create formulas, using all the operators in different combinations. Try doing things that you know will cause errors. Then correct the errors in edit mode, and so on. It takes a thorough grasp of the fundamentals to get the most out of *SpeedCalc*'s advanced features.

Change Type And Format

The default (normal) format for numeric data is flush right with rounding to two decimal places. In other words, the number is displayed in the rightmost part of the cell, with two numbers after the decimal point. Text and formulas are flushed left (shown in the leftmost part of the cell). *SpeedCalc* offers several commands for changing cell formats.

Change Type (CTRL-T). As noted earlier, *SpeedCalc* displays a T, N, or F in the upper-right corner of the screen to show the data type of the current cell. Occasionally you may need to change a cell's data type. Press CTRL-T for the change type command: *SpeedCalc* displays the question CHANGE TO: TEXT, NUMERIC, FORMULA? in the command line. To change the data type, press T, N, or F. Be careful not to create new errors when doing this.

Change Format (CTRL-F). This command changes the location of data in the cell and the number of decimal places. When you press CTRL-F, *SpeedCalc* displays the question FORMAT: Left, Center, or Right justify? in the command line. Press L, C, or R to move the data to the left, center, or right of the cell. Next, you must select the number of decimal places for that cell. The default value is 2, but you may change it to anything from 0–15. If you choose zero decimal places, any number in that cell is rounded off to the nearest integer (whole number). If you choose 15, a number in that cell is not rounded off at all—*SpeedCalc* displays it exactly as you entered it or as it was calculated from a formula.

Width (CTRL-W). The width command changes the width of an entire column of cells. Move the cursor to any cell in the desired column, then press CTRL-W. When *SpeedCalc* displays the prompt Width: you should respond with a number from 4–36. The entire screen is redrawn to accommodate the new format, and may look very different depending on what value you chose. For instance, if you increase a column's width, the rightmost column of the former display may disappear: *SpeedCalc* only displays as many complete columns as it can fit on the screen. If you decrease the width of a column, you may see asterisks where numbers used to be (indicating the cell is now too small to display the entire number). To get rid of the asterisks, expand the column as necessary.

Global Format (SHIFT-CTRL-F). This is the same as the ordinary format command, but operates globally, changing every cell in the sheet instead of just one. To alert you to the difference, *SpeedCalc* changes the color of the command line to blue.

Global Width (SHIFT-CTRL-W). This is a global version of the width command. The command line turns light green to signal the difference. Every column in the sheet changes to the designated width.

Macro Editing

After typing in a large spreadsheet, you may decide to make a major change. You may want to add new data somewhere in the middle, delete a section, or move a group of cells from one location to another. *SpeedCalc*'s macro (large-scale) editing commands simplify such operations, affecting an entire block of cells at once. A *block* is simply a group of cells connected in rectangular fashion: You can define it as a single cell, a row or column, or any rectangular area within the spreadsheet.

There are two ways in which macro commands can work: *verbatim* or *relative*. To take a simple example, say that cell AA2 contains the formula =aa1*5 and you want to move its contents to cell AB2. When this is done in verbatim mode, AB2 contains an exact copy of what was in AA2 (=aa1*5). Note that the cell name used in the formula does not change: The formula still refers to AA1. If you perform the same operation in relative mode, the cell name in the formula is adjusted to fit the new location. In this case, AB2 would contain the formula =ab1*5.

Copy (CTRL-C). The copy command copies a block of cells into a different location without disturbing the original cells. Place the cursor on the upper-left corner of the block you want to copy, then press CTRL-C. *SpeedCalc* changes the command line to purple and prompts you to move the cursor to the lower-right corner of the block you want to copy. Once the cursor is in place, press RETURN. Now *SpeedCalc* prompts you to move the cursor to the place where you want to put the block: This is the upper-left corner of the new position. Once the cursor is there, press RETURN again. The new data replaces whatever was contained in the designated cells. Note that if you define an impossible block (for instance, moving the cursor to the upper-left of the original position, rather than below and to the right), *SpeedCalc* does not copy any data.

Move (CTRL-M). This command works like a copy, but it fills the

Sensational Prices!
. . . On Our Most Popular Items!

from MicroPal® . . .

THE 39¢ DISKETTE!

Are you paying too much for diskettes? Try our first quality, prime, 5¼" diskettes (no rejects, no seconds) at these fantastic sale prices and save, save, SAVE! Disks are packaged in boxes of 50; including diskettes in sleeves, labels, and write-protect tabs.

Each diskette is certified to be 100% error free and comes with a lifetime warranty (if you have a problem, we'll replace the diskette). All diskettes include hub reinforcement rings and write-protect notch.

All diskettes are double density and work in either single or double density drives.

SS, DD Diskettes, Box of 50
32391 **$19.50–39¢ ea.!**

DS, DD Diskettes, Box of 50
32403 **$24.50–49¢ ea.!**

Unleash The Power of CP/M™ On Your C128!

ONLY $22.95

Includes Free CP/M Programs

• Word Processor • Disk Utilities • Communications

CP/M Kit includes a helpful guide to using CP/M such as: stressing the unique features of Commodore's CP/M, using your 1541 or 1571 drive with CP/M, and using your modem to access electronic bulletin boards that offer free, downloadable software.

67392 2 Disks & Manual for C128 . . . **$22.95**

$159 FSD-2 DISK DRIVE

Reviewed by *RUN Magazine:*

". . . higher-than-average quality . . . 10/ faster than a 1541 . . . half the size . . . you can rest assured that the FSD-2 will do a fine job for you."

Runs all C64 Commercial Software
Full 1 Year Warranty
66166 Disk Drive . . . **$159**

★ THE BEST PRICES ★
★ THE BEST SERVICE ★
WHY SHOP ANYWHERE ELSE?

THE FAMOUS SLIK STIK™ — **ONLY $6.95!**

The Slik Stik™ has been a favorite for years . . . and for good reason. It's just the right combination of responsiveness and accuracy. And the price can't be beat! From Suncom. 90 day warranty. Connects directly to Commodore Computers.
42086 **$6.95**

POWER and PROTECTION FOR YOUR C-64!®

POW'R PAK 64

from MicroPal®

Pow'r Pak is a replacement power supply (1.5 amp) for the Commodore 64® . . . but that's not all! Pow'r Pak also supplies two additional surge protected outlets (120V) for monitor, disk drive, or other peripherals. On/off switch. Fuse protection. Sturdy all-metal casing is ventilated for heat dissipation. Full 1 year warranty.

34910 **$49.95**

Space Saving Printer Stand

NEW LOW PRICES!

At last a printer stand priced for homes and small offices! This great stand from Suncom is elevated so you can place your paper underneath. Features heavy duty welded steel wire construction and has a vinyl coating to reduce noise and vibration. Another plus . . . the printer is held at an angle that makes it much easier to see your output as it is being printed. Available in two widths: 12" for most 80 column printers, and 18" for most 132 column printers.

33232 12" Printer Stand . . . . . (Was $14.95) **ONLY $ 6.95**
33246 18" Printer Stand . . . . . (Was $22.95) **ONLY $12.95**

Paper Tray

Ever wondered what to do with the printout that comes out of the printer and tends to spill all over the table and floor? The Suncom Paper Tray attaches to the rear of all Suncom stands and receives the printed paper as it comes out of the printer. Helps to neatly fold and stack it!

Sug. Retail $14.95
42091 . . . . . **$11.95**

Super Deal On Diskette Storage!

ONLY $9.95

- A whopping 100 disk (5¼") capacity.
- Lock and keys for extra security and easy carrying.
- Includes 8 index dividers with labels for organization of filing and retrieval.
- Made of durable anti-static, high impact plastic.
- Attractive smoked color lid.

Sug. Retail $19.95
66826 **NOW ONLY $9.95**

FREE

DUST COVER & EVERYTHING BOOK

Discover the savings and easy shopping available from TENEX Computer Express PLUS receive a **FREE** dust cover for your C64 or C128! Cover is anti-static, 8-gauge vinyl sewn to our exacting standards with reinforced seams. Get to know our great products, extensive selection and fast service with a **FREE** copy of our *Everything Book for Commodore Computing.* ($2.95 Shipping Charge)

31627 C-64 Dust Cover and Catalog (G4F)
38464 C-128 Dust Cover and Catalog (G4F)
65180 64C Dust Cover and Catalog (G4F)

The Right Interface For All Your Printing Needs!!

PPI with Graphics

". . . a sound investment for your Commodore." RUN, Dec. '85 This high-performance graphics parallel printer interface from DSI for C-64 and VIC-20 emulates a Commodore printer. Comes with cables and user's manual.
33565 . . . . . **$39.95**

Super Graphix. Features 8K buffer, 10 printing modes, 3 internal screen dumps and top mounted dip switches. Supports superscript, subscript, underlining, bold face, and a choice of 9 character widths. From Xetec.
41769 . . . . . **$CALL**

Super Graphix Jr. A more economical version of the Super Graphix, featuring graphics, normal and correspondence quality printing. Compatible with all major printers. From Xetec.
41774 . . . . . **$CALL**

From Your Friends At

TENEX Computer Express

We gladly accept mail orders!
P.O. Box 6578
South Bend, IN 46660
Questions? Call 219/259-7051

Ad G4F

SHIPPING CHARGES

ORDER AMOUNT	CHARGE
less than $20.00	$3.75
$20.00–$39.99	4.75
$40.00–$74.99	5.75
$75.00–$149.99	6.75
$150.00–$299.99	7.75
$300 & up	8.75

MasterCard VISA

NO EXTRA FEE FOR CHARGES
WE VERIFY CHARGE CARD ADDRESSES

ORDER TOLL FREE
1-800-348-2778
INDIANA ORDER LINE 1-800-225-6838

COMMODORE 64 is a registered trademark; and COMMODORE 128 is a trademark of Commodore Electronics, Ltd. Foreign Orders add $4.00; heavy items ship at actual cost.
NOTE: Due to publishing lead-times, product prices and specifications are subject to change without notice.

original cells with blanks. Though *SpeedCalc* has no express insert command, you can use this command to make space for new data in the middle of a spreadsheet. Simply move everything below the insertion point down as far as you need.

Relative Copy (SHIFT-CTRL-C). This form of the copy command adjusts the cell names used in formulas within the copied block (see explanation above).

Relative Move (SHIFT-CTRL-M). This is the relative form of the move command. Cell names in formulas are adjusted to reflect the move.

Memory Management

SpeedCalc makes about 10K (over 10,000 characters) of memory available for data. As noted earlier, *SpeedCalc* lets you spread your data out over a much larger number of cells than you can actually fill with data. The extra space is provided to give you full control over the final format of the spreadsheet—for example, you could have a 15 × 150 spreadsheet—and to leave some elbow room for move and copy operations.

Because memory is limited, you should keep careful track of how much is free while using the program. Press CTRL-A to display the amount of free memory. We suggest limiting your spreadsheets to 1,296 cells (equivalent to 36 rows by 36 columns). If you have filled nearly all of free memory, you may have to break the spreadsheet into two smaller sheets.

Although *SpeedCalc* checks the amount of available memory, and displays an error message if you run out of memory, you should be careful not to exhaust free memory. Any move or copy operation in process will be aborted if sufficient memory is not available.

Disk Operations

SpeedCalc has four disk commands which allow you to save a spreadsheet to disk, load it, display the disk directory, and send commands to the disk drive. The directory command is the simplest to use. Press CTRL-4 (think of the dollar sign): The screen clears and the directory is displayed. Press RETURN to return to the normal screen. You may pause the directory display with the space bar.

To save a spreadsheet to disk, press the f8 function key (SHIFT-f7). *SpeedCalc* prints SAVE: on the command line, followed by an underline cursor. Enter a valid Commodore filename and press RETURN. (If you change your mind and decide not to save anything, press RETURN without typing a filename.) The disk drive spins for a few moments, then *SpeedCalc* prints the drive status in the command line. The message 00,OK,00, 00 means there were no errors.

To load a saved file from disk, press the f7 key. Again, *SpeedCalc* prompts you to enter the filename and displays the disk status when the operation is complete.

You can perform other Commodore disk commands by pressing CTRL-↑—press CTRL and the ↑ (up arrow) key together. *SpeedCalc* prompts you to enter a disk command. If you press RETURN without typing a command, *SpeedCalc* displays the drive status and sends no command. You need not enclose the command in quotation marks or type ,8 after it. For example, press CTRL-↑, then enter V0 to validate a disk. Consult your disk drive manual for more information about Commodore disk commands.

Printing

SpeedCalc lets you print data to three different devices: to the screen for previewing output, to a printer for permanent documentation, or to a disk file for integrating the data with a *SpeedScript* document.

To preview your spreadsheet on the screen, press SHIFT-CTRL-P, then press S (screen output) when prompted.

To print a hardcopy of the spreadsheet, press CTRL-P. If your printer is configured like most, this should produce a satisfactory printout. This command sends output to the printer as device number four with a secondary address of seven (uppercase/lowercase on most systems). *Before using this command, you must position the cursor below and to the right of the block of cells you wish to print.* The upper-left corner of the printout starts at cell AA1.

To send output to a printer with a device number other than four or a secondary address other than seven, enter SHIFT-CTRL-P, then enter the device number and secondary address when prompted. During a printout, you can pause the output by pressing SHIFT or SHIFT LOCK: The screen border turns white and printing ceases until you release SHIFT. Press RUN/STOP to abort printing.

You can also print *SpeedCalc* data to a disk file for use in a *SpeedScript* document. Select the D option after pressing CTRL-SHIFT-P, then enter the filename you wish the new file to have. The data is saved as a disk file of that name. Note that printing to disk creates a different file than saving to disk: You should save files which you wish to reload into *SpeedCalc*, and print files which you wish to convert for *SpeedScript*. While you may pause this operation with SHIFT as with printer output, *do not use RUN/STOP to abort printing to disk.* This may create a "poison" (unclosed) file which can safely be removed only by validating the disk.

Table 1: *SpeedCalc* Commands

Command	Action
CTRL-A	available memory check
CTRL-B	blank (erase) current cell
CTRL-C	copy block verbatim
CTRL-E	edit current cell
CTRL-F	change cell format
CTRL-G	goto selected cell
CTRL-M	move block verbatim
CTRL-P	print file on printer
CTRL-T	change data type of cell
CTRL-W	change column width
CTRL-X	exit *SpeedCalc*
CTRL-4	disk directory
CTRL-↑	send disk command
CLR/HOME	home cursor
SHIFT-CTRL-C	copy block relative
SHIFT-CTRL-M	move block relative
SHIFT-CTRL-P	print to screen, disk, or printer
SHIFT-CTRL-W	change width of all columns
SHIFT-CLR/HOME	erase entire sheet
f1	change border color
f3	change background color
f5	change character color
f7	load *SpeedCalc* file
f8	save *SpeedCalc* file
←	recalculate sheet

SpeedScript Integrator

SpeedCalc sends data to the printer in simple, plain vanilla form. That may be fine for personal use, but if you're creating a document for others to view, you may want special features such as boldface, underlining, etc. Since *SpeedScript* already offers a way to access these features (and many more), no attempt has been made to include them in *SpeedCalc*. All that's needed is a simple program to convert *SpeedCalc* files into a form that *SpeedScript* can load. Then you can edit the file with *SpeedScript* as you would any other document—inserting printer control codes, reformatting the text, merging it with other text, and so on.

Here are the steps to follow to convert a *SpeedCalc* file for *SpeedScript:*

1. After creating a spreadsheet with *SpeedCalc*, print it to disk as described above.

2. Exit *SpeedCalc*, then load and run the "SS Integrator". The program prompts you to enter the name of the *SpeedCalc* file you printed to disk. Then it asks you to enter the name of the *SpeedScript* file you want to create (of course, this name should be different from the first). The Integrator then constructs a *SpeedScript*-loadable disk file from the *SpeedCalc* file.

3. After the Integrator is finished, load and run *SpeedScript*, then load the new *SpeedScript* file as you would any *SpeedScript* document. The data appears on the screen, ready to be edited in any way you wish. ©

Sequential File Converter

For *SpeedScript*

Ron Carnell

SpeedScript *becomes an even more valuable word processor with this program that converts sequential files into* SpeedScript *format. It's written in machine language for fast results. For the Commodore 64.*

I've been using the Commodore 64 version of *SpeedScript*, in one version or another, since it was originally published in the January 1984 issue of COMPUTE!'s Gazette. The only drawback I've ever found is its incompatibility with sequential files. (*SpeedScript* uses program files.)

My database uses sequential files, and many downloaded programs are in sequential file format. And I have a few friends that use other word processors that create sequential files. In short, there are many instances when I'd like to load a sequential file into *SpeedScript*.

Over the years, COMPUTE! and COMPUTE!'s Gazette have published several BASIC programs designed to convert sequential files into *SpeedScript* files. And I've used them, mostly with great success. But BASIC is inherently slow; when my database started creating sequential files more than 100 blocks long, it became obvious that I needed something faster.

"Sequential File Converter" (SFC), the program accompanying this article, is the solution. SFC accomplishes the same thing as the earlier BASIC conversion programs, but at the speed of machine language.

To use SFC, you need only load it and type RUN. You'll be prompted for the name of the source file (the sequential file) and for the name you'd like for the new *SpeedScript* file. (Be sure to have the correct disk in the drive when you enter the destination name.)

If you load SFC and type LIST, you'll find several lines of BASIC. With one exception, you should *never* change any of this code. The machine language is attached to the end of the BASIC program, and changing any of the BASIC code will almost certainly cause SFC to crash. The exception is line 140. In this line, SFC is adding ",s,r" to the end of your source filename. The *s* tells the disk drive to access a sequential file, and the *r* indicates that it should be read. You can, in this instance, change the *s* to another letter without changing the length of SFC and therefore without causing the machine language to crash. You might wish to do this if you want to change a file format other than sequential into a *SpeedScript* file. For example, a few word processors store their text as USR files. By changing that *s* to a *u*, you could make SFC a USR converter able to access USR files. ©

ScriptRead

Buck Childress

This versatile utility allows you to read any SpeedScript *file (version 2.x or 3.x) at high speed. Among its features are automatic word-wrapping and word counting. It also has a scratch command to remove unwanted files. A disk drive is required.*

After a while, your *SpeedScript* disks can become overcrowded with files. Even with the most descriptive of filenames, it's hard to remember the contents of every file. "ScriptRead" lets you read through any number of *SpeedScript* files, scratching any that are no longer needed. And since ScriptRead works so fast, it can also help you to locate a particular file—without your having to waste time loading file after file into *SpeedScript*.

Using The Program

To use the program, load it and type RUN. The program asks what you would like to do. Press M to see a menu of your options.

You may change the screen colors for easier viewing by pressing T to change the text color and B to change the background color.

Press D to see the disk directory. Press CTRL to slow the directory listing. Press Q to quit the listing.

To read the contents of a file, press R and then type the name of the file. ScriptRead displays the file on the screen. Since ScriptRead includes an automatic word-wrap function, you'll never have to contend with words that are split between two lines. Press CTRL to slow the listing, any function key to pause it, or Q to quit reading the file. When the end of the file is reached, the number of words in the file is displayed.

Press S to scratch a file. ScriptRead asks for the name of the file to scratch. Type in the name of the file you wish to erase, or press RETURN if you don't wish to erase any files.

If you encounter a disk error (your disk drive light blinks on and off), press E to read the error channel. See your disk drive manual for an explanation of disk errors.

You may press RUN-STOP at the "What would you like to do?" prompt to exit ScriptRead. ©

Free-Form Filer

David Myles

It's never been easier to store recipes, magazine indexes, financial information, or addresses. This free-form database features power through simplicity—an impressive tool for the 64. A disk drive is required.

"Free-Form Filer" is like a deck of index cards that you can easily add to and edit. Advanced features let you arrange the cards in the order you want, hunt for key phrases, and print a card to a printer. When you have everything in the file just the way you want it, simply press one key to save the entire file to disk.

The program is written to be intuitive, so it's very easy to learn. Type information any way you want—the program's automatic centering will keep all your data nice and neat.

Free-Form Filer stores information by adding to itself. That is, the information you enter into the Free-Form Filer database becomes part of the program, and the program grows longer as you add more information. For this reason, *you must use a separate copy of Free-Form Filer for each database you create.* For example, if you want to set up an address file, a recipe file, and a household inventory, then each set of information needs to be entered into its own copy of the Free-Form Filer program. Furthermore, because the program always saves itself back to disk with the same name, *there can be only one copy of the Free-Form Filer program on a disk.*

You should save one copy of the program with no data added as a master copy, then save a working copy on each disk on which you wish to create a database. Because the program is written to load the same way a BASIC program does, you can save additional copies simply by using the standard SAVE command. Remember that all copies *must* have the name FREE-FORM FILER. It would be wise to indicate on the disk's label what type of information is contained in the copy of Free-Form Filer on that particular disk.

Making A Note

Load Free-Form Filer the way you would load a BASIC program (LOAD "FREE-FORM FILER",8); then type RUN. You'll see a menu at the bottom of the screen:

Hunt Re-try Add Print Fwd Back
Change Dlete Tobuff Save Insbuff

This menu is always visible, so you don't have to worry about memorizing a set of commands. At this point, Free-Form Filer asks you to select from the menu. To start, we want to add to the file, so press A. You'll see a message which says that a carriage return (RETURN) stores the line, and that a RETURN on a blank line ends data entry. To see how it works, type these lines, pressing RETURN after each:

FREE-FORM FILER
FEATURES AUTOMATIC CENTERING

Then, press RETURN on a blank line. Free-Form Filer makes a *card* with the information that you entered. This card is the first in the deck. (In Free-Form Filer, a card corresponds to one screen of information—up to 23 lines of text.) Press A again to make another card. Then type

TYPIGN ERRORS ARE EASY TO CORRECT.

and press RETURN twice. Now press B (Back) to view the first card. It's impossible to back up beyond this card. Press F to go forward. You'll see the second card again. To correct the misspelled word, *TYPIGN*, press C (Change). Correct the word by using the cursor keys to move to the error, and then typing over the incorrect letters. When you've made the change, be sure to press RETURN before moving the cursor off the screen line.

Each time you enter a line, it's saved in the computer's memory. The computer uses all these lines to build the index card. It's easy to enter duplicate lines—just press RETURN several times on the same line. Be sure not to press RETURN on a blank line until you've pressed RETURN on all the lines you want to be saved on the card—this applies when you're entering data for the first time and when you're editing it.

If, while entering information on a card, you notice a mistake in a previous line, do not cursor up and

correct the error. Instead, finish entering the rest of the information for the card, press RETURN on a blank line to enter the card, and then use the C option to go back and change the mistake.

Because blank lines are used to terminate data entry, it's impossible to use them to separate text on the screen. If you want to separate lines, enter a shifted space (hold down SHIFT while pressing the space bar) as the only character on the line. Alternatively, you can use a character like a period or minus sign.

Free-Form Filer makes it easy to arrange the order of the cards. Press B until you back up to the first card. Then press T (To buffer). The first card is now being held in a buffer. Now press F (Forward) to go past the second card. Press I (Insert buffer). The contents of the buffer are now placed after the second page. (Whenever you press I, the contents of the buffer are stored in a new card *in front of* the card currently on the screen.) Use B and F to see that the first card has been moved to the end.

You should be aware that anytime you use the C option to change a card, that card will be moved to the end of the deck. If the order of the cards is important to you, then you'll have to follow the procedure to move the card back to its desired position after changes are made.

If you decide that you want to throw out a card, press D (Delete). Free-Form Filer will ask if you're sure. If you press Y, the card will be deleted.

Printing, Searching, And Saving

Sooner or later, you'll probably want a printout of your data. When you do, just press P for a printed version of the card currently displayed on your screen.

The most powerful feature of Free-Form Filer is *Hunt*. When you want to search for a word or phrase, just press H. (For Hunt to work properly, you should be positioned at the first card in the deck before pressing H.) The program will ask for the phrase. Type it in, following it with a RETURN. Free-Form Filer will find and display the first card with the phrase you indicated. To search for the next occurrence, press R (Retry).

When you've finished editing your cards, press S to save them to disk. Free-Form Filer scratches its program file from disk and replaces it with the program plus all the notecards you have entered. The next time you want to edit or read your cards, just load Free-Form Filer—all your cards will be there.

Be very careful that you don't hit the RUN/STOP–RESTORE combination while using Free-Form Filer. If you do, all the changes made since the program was loaded will be lost when you run the program again to restart it. The cards will still appear on the screen if you flip through the file, but the changes will not be saved to disk. To reenter the changes, you must move to the first card in the file, then use the C option and press RETURN on every line of every card.

By saving Free-Form Filer to several disks, you can keep several different types of files. If you ever want to clear out all the cards in a deck to start a new file, just use D (Delete) on each card. ©

TurboDisk 64

Don Lewis

Impossible as it seems, "TurboDisk 64" speeds up the 1541 disk drive's loading time 300 percent or more—in fact, the longer the program, the more the gain. Also in this issue, you'll find several programs which expand the power of TurboDisk: "TurboDisk Relocator" and "Turbo Boot Maker."

If you've ever used a really fast disk drive, you know that the Commodore 1541 drive leaves something to be desired—namely, speed. True, it's much faster than a Datassette, but it's still annoyingly slow compared to other floppy disk drives with high-speed parallel interfaces. Now there's a solution: "TurboDisk 64."

Once you start using TurboDisk, you'll wonder how you got along without it. TurboDisk turbocharges the loading process by a factor of three times or more. In fact, the longer the program, the more improvement you'll see.

TurboDisk requires no modifications to your disk drive or computer. It loads programs saved in the usual manner; no special Turbosave is required. It works with most BASIC and machine language programs, including the DOS Wedge. It does not compromise reliability. And you can switch it on or off at any time by typing a single command.

If you're still skeptical, give TurboDisk a try—it delivers what it promises.

To load TurboDisk, use the command LOAD "TURBODISK 64",8,1. Next, type NEW and press RETURN to reset important memory pointers, then SYS 49152 to activate the program. The message TURBODISK ACTIVATED signals that you're ready for high-speed loading.

Turbocharged LOADs

Once TurboDisk is activated, no special commands are necessary. Just type LOAD *"filename"*,8 or LOAD*"filename"*,8,1 as usual. You'll be amazed at the difference.

One thing you'll notice immediately is that the red light on the disk drive doesn't come on at all during a Turboload. Don't panic; this is normal. It's also normal for the 64's screen to blank out as TurboDisk works. When the program is loaded, the screen reappears unaltered.

You may occasionally find it necessary to deactivate TurboDisk and use a normal LOAD instead. For example, 1541 disk drives are prone to head-alignment problems, so if you have a disk formatted on a drive other than your own, you may find that your drive has difficulty loading programs from it. Since the Turboload routine gives up more easily on difficult LOADs, you may have to switch to the more forgiving standard LOAD to get the program into your computer. You can switch off TurboDisk at any time without erasing it from memory by entering SYS 49155 (you should see the message TURBODISK DISABLED). To reactivate TurboDisk, enter SYS 49152.

You'll also find it necessary to use the SYS to reactivate TurboDisk after pressing RUN/STOP–RESTORE, which effectively disconnects TurboDisk.

TurboDisk resides in the 4K block of free memory starting at address 49152 (hex $C000), so it's completely safe from BASIC. However, many other machine language programs or subroutines also use this memory space and may overwrite TurboDisk. Don't attempt to use TurboDisk to load any program which occupies locations 49152–50431 ($C000–$C4FF). If the program you wish to Turboload uses this area of memory, you can use "TurboDisk Relocator," found elsewhere in this issue.

TurboDisk speeds up LOADs, but it can't speed up SAVEs or VERIFYs. It also doesn't affect the speed of disk file handling with OPEN, PRINT#, GET#, and so forth. It's not compatible with certain features of some programs, such as loading text files with *SpeedScript*, although you can use TurboDisk to load *SpeedScript* in the first place. It also may not work with some commercial software.

How TurboDisk Works

The machine language for TurboDisk is unusual in that only half of it works within your computer—the rest is actually executed within the 1541 drive itself. Commodore disk drives are *intelligent* units, containing their own microprocessors, RAM, and ROM. This means that they can be programmed for special effects, like turboloading.

During the brief delay you notice between the time you enter the LOAD command with TurboDisk and the time the drive starts spinning, 444 bytes of machine language are transferred from the computer to the drive's RAM. In the

64, it is stored in locations 49664–50107 ($C200–$C3BB). This required transfer of data before each Turboload adds a certain amount of overhead time, which explains why TurboDisk gives less speed improvement for short programs.

TurboDisk operates by changing the ILOAD vector at locations 816–817 ($330–$331) to point to itself, bypassing the normal LOAD routines in ROM. TurboDisk first checks to see whether a disk directory (LOAD "$",8) or a VERIFY was requested. In either of these cases, control is returned to the ROM routines for normal processing. If a program LOAD was requested, the routine adds the filename to the code for the disk drive portion, then transfers that data to the drive's memory.

The portion of TurboDisk in the disk drive uses routines in the drive's ROM to locate the desired program and read it from the disk sector by sector. To improve speed, routines like the one that turns on the red light are omitted, and only the essential ones are used. The 256 bytes of data from each disk sector are transferred two bits at time to a 256-byte buffer within the computer. In the 64, this buffer is at locations 50176–50431 ($C400–$C4FF).

TurboDisk machine language in the computer reads the incoming data from the serial port's DATA and CLK lines, instead of just the DATA line as in normal serial data transfers. Thus, TurboDisk temporarily converts your serial bus into a two-bit parallel bus. When the entire 256 bytes from a disk sector have been transferred into the computer's buffer, data from the buffer is added to the program in memory while the drive is reading the next sector from the disk.

Just How Fast Is It?

Despite a few limitations, TurboDisk is one of the most valuable general-purpose utilities a disk user can own. To discover exactly how fast it is, we ran some tests. The results, below, demonstrate how TurboDisk yields the most improvement with medium to long programs. (Results with different disk drives may vary.)

Program	Blocks	Normal LOAD	Turbo-load	Factor
Program 1	31	21 sec	7 sec	3.0
Program 2	17	13 sec	5 sec	2.6
Program 3	45	31 sec	9 sec	3.4
Program 4	8	7 sec	5 sec	1.4
Program 5	25	18 sec	6 sec	3.0
Program 6	122	75 sec	17 sec	4.4

Special Note To 128 Owners

For those 128 owners who use a 1541 disk drive: When you're in 64 mode, you can load and run TurboDisk 64 as listed. For 128 mode (and a 1541), see "TurboDisk 128" elsewhere in this issue.

If you own a 1571 disk drive, you don't need a Turbo program while you're in 128 mode—the 128 and 1571 are capable of using the fast serial transfer hardware built into the disk drive and computer. The 128/1571 combination is twice as fast as TurboDisk at its best, and about eight times faster than the 1541 alone. The 1571 slows down to the 1541's sluggish pace when you're in 64 mode, however. To speed up the 1571/64 mode combination, you can use TurboDisk 64. But first you must tell the 1571 to act like a 1541 with the following line:

```
OPEN 15,8,15, "U0>M0": CLOSE 15
```

To go back to 1571 operations, change the second zero to a one ("U0>M1"). However, you must make a decision. You can use double-sided disks while the 128 is in 64 mode and the drive is in 1571 mode, but not while using TurboDisk 64. Or you can use TurboDisk to speed up the 1571 drive in 1541 mode, but then you cannot use both sides of a disk. It's up to you: fast operations limited to one side of the disk or slow loads and access to both sides of a disk.

Note to readers outside North America: High-speed TurboDisk data transfers rely on precise timing, so the program may fail to operate on systems that use the European PAL video system instead of the North American NTSC system. The reason is rather technical—64s with PAL video use a slightly different microprocessor clock frequency.

Commodore Product Potpourri

Hardware, Software & Firmware for your C-64, C-64C, C-128, AMIGA

BASIC PROGRAMMING AID

VICTREE is a BASIC programming aid cartridge for the VIC-20 and C-64/C-64C computer. **VICTREE** adds 42 extra commands for BASIC programming ease and full DOS control. **VICTREE** commands include CHAIN, EXECUTE, MERGE, SEND, USE, CONCAT, DOPEN, DCLOSE, RECORD, HEADER, COLLECT, BACKUP, COPY, APPEND, DSAVE, DLOAD, DIRECTORY, RENAME, SCRATCH, CATALOG, INITIALIZE, AUTO, DELETE, RENUMBER, MERGE, LCOPY, LMOVE, PRINT USING, FIND, CHANGE, PAGE, HELP, DUMP, and TRACE.

VICTREE 64 Cartridge, C-64/C-64C . . $49.95*
VICTREE 20 Cartridge, VIC-20 . . . . . $29.95*

Prevent and preclude premature pratfalls, by perplexed programming postulants

BASIC PROGRAM COMPILER

BLITZ! is the fastest, easiest to use, most popular, BASIC program compiler available for the **Commodore 128, C-64, and C-64C.** Your **BLITZ!** compiled programs will run from 5 to 20 times faster after you have **BLITZ** them. **BLITZ!** translates and reduces your BASIC programs in to a much, much faster code.

BLITZ! C-128, Disk $59.95*
BLITZ! C-64, Disk $49.95*

The periodically puzzling, precarious, path to perspicacious program perfection and prowess

ASSEMBLER CARTRIDGE

If your C-64 programming needs have extended beyond BASIC, Skyles Electric works now offers **MIKRO**, a machine language assembler cartridge for the **Commodore 64/64C**. The **MIKRO** cartridge contains everything you need for machine language programming.

MIKRO Cartridge, C-64/C-64C . . . $49.95*

M.L. UTILITY TRIO

ZOOM is the perfect machine language monitor for the Commodore **C-64/C-64C**, with 23 editing commands and 11 disk commands. **TD** allows the full power of **VICTREE** to be applied to editing and writing machine language programs. **STP** is a ML Step-Wise Executor that is one of the nicest ML debuggers we have ever seen, for the **Commodore 64. ZOOM—TD—STP** the tremendous trio now in one package.

ZOOM-TD-STP, C-64/C-64C $29.95*

DISK ALIGNMENT KIT

1541/1571 Disk Alignment Kit is the do-it yourselfer's delight. You furnish a voltmeter and 1 hours time and the **1541/1571 Disk Alignment Kit** furnishes; 1-Alignment Reference Disk, 2-test and alignment program disk, 3-video detector, 4-Instruction manuals, 5-1541 Maintenance Guide. All for less than the cost of a disk alignment.

1541/1541C/1571 Disk Alignment Kit $29.95*

2 for 1 MONITOR CABLE/C-128

The**2 for 1 Monitor Cable** allows all composite video monitors to be used with the **Commodore 128** in all modes of operation. Don't throw out your present green or amber monitor, buy a

2 for 1 Monitor Cable for C-128 . . . . . $24.95*

1541 FLASH!

The new Skyles Electric Works **1541 FLASH!** loads programs and files to your **Commodore 64/64C or Commodore 128** (64 mode) three times faster than an unenhanced Commodore 1541 disk drive. **1541 FLASH!** saves programs 20 to 50 percent faster. **1541 FLASH!** formats a diskette in 25 seconds, a real flash.
The **1541 FLASH!** is a permanent hardware installation in your **Commodore 64/64C** and **Commodore 128** (64 mode) and **1541 disk drive.** No programs to load, no cartridge hassles.

We have special versions of the **1541 FLASH!** for the **SX-64** and two **1541s**.

In addition to its blinding speed of program and file loading, the **1541 FLASH! adds over 50 extra commands for the Commodore 64/64C/128** user. These include a built-in DOS/Wedge, Easy Editor, and FLASHMON! machine language monitor.

1541 FLASH! C-64/C-64C & 1541 . $69.95*
1541 FLASH! C-64/C-64C & two 1541s $109.95*
C-128 FLASH! C-128 & 1541 . . . . . $79.95*
C-128 FLASH! C-128 & two 1541s $119.95*
SX-64 FLASH! SX-64 & 1541 . . . . $69.95*
SX-64 FLASH! SX-64 & two 1541 . . . . . $109.95*

A powerful panoply of pertinent, potent, peripheral products

QUICKSILVER 128

QUICKSILVER 128 our premier IEEE-488 Interface for the **Commodore 128** is now in stock and even better than we had planned. **Quicksilver 128** offers an IEEE Interface for the **Commodore 128** in the C-128 mode (40 or 80 columns) and in the C-64 mode as well. **QUICKSILVER 128** will inter-connect your **Commodore 128** to Commodore SFD 1001, 2031, 2040, 3040, 4040, 8050, 8250, 9060, 9090 Disk Drives, and 2022, 2023, 4022, 4023 and 8023 Printers.

QUICKSILVER 128 C-128 . . . . . $119.95*

A priceless programmers pallett of practical products and programs

IEEE Flash! 64

IEEE Flash! 64 our premier IEEE-488 Interface for the **Commodore 64** is now in stock and even more transparent (program compatible) than QUICKSILVER 64. **IEEE Flash! 64** will inter-connect your **Commodore 64** to Commodore SFD 1001, 2031, 2040, 3040, 4040, 8050, 8250, 9060, and 9090 Disk Drives, and 2022, 2023, 4022, 4023, and 8023 Printers.

IEEE Flash! 64 C-64/C-64C $99.95*

2+1, 4+1 CARTRIDGE EXPANSION

Our 2 + 1 Cartridge Expansion gives you 2 vertical and 1 horizontal fully switchable cartridge ports. Our 4 + 1 Cartridge Expansion gives you 4 vertical and 1 horizontal fully switchable cartridge ports.

2 + 1 Cartridge Exp. C-64/C-64C or C-128 . . . . . $39.95*
4 + 1 Cartridge Exp. C-64/C-64C or C-128 . . . . . $69.95*

1571 FIX ROM

Relative File problems? "DEVICE NOT PRESENT" errors? I/O error 5 when using Superbase? Major problems when you have 2 or more files open? Painfully slow writing to the second side of the diskette? SAVE with replace problems? Takes forever to recognize a "Flippy" disk? If you have one or more of these problems with your **Commodore 1571 disk drive**, don't despair, don't trash your 1571, **do get a:**

1571 FIX ROM $24.95*

A powerful panoply of pertinent, potent, peripheral products.

VIDEO DIGITIZER

The **VIDEO-DIGITIZER** module plugged into the user port of your **C-64/C-64C/C-128** allows you to digitize video signals, of whatever kind, show them on the screen, store or process them and print them out. The digitizer can be connected to any standard; video camera (b&w or color), VCR, or video disk player. The applications are unlimited. An accompany diskette contains programs to; Digitize a 256 by 256 pixel picture in 4 shades of gray, replace gray shades with colors, save complete 256 by 256 pixel picture, or 160 by 200 **Koala** format, printout on **Commodore 1515, 1525, 801, 802, 803, 1526, Epson, StarMicronix**, (and compatibles) **Canon PJ1080A, Okimate 20,** and **GP-700A** color printers.

All this is for an amazingly low price.

VIDEO-DIGITIZER, C-64/C-64C/C-128 . . . . . $149.95*

BUSIDATA™

BUSIDATA features up to 1000 records per 1541 data disk, 2000 records per 1571 data disk, 6000 per SFD-1000 data disk, and over 20,000 records per SEW/JCT-1002 hard disk drive (see page 5). Each record can contain up to 254 characters and 20 fields of up to 79 characters per field. **BUSIDATA** features up to 20 keys and five levels of sorting on these on these keys. Also featured are easy to use menu driven data selection and handling features, that include RS-232 printer output.

BUSIDATA 64, C-64 Disk $39.95*

A major manufacturers magnificent multitude, magnanimously merchandised for minimal money.

FAMILY TREE

Truly a program for the whole family, your wife's family, your parent's family, grandparent's family, and as many generations back as you wish. FAMILY TREE is the best genealogy program for the **C-64/C-64 Cor C-128.** Features unlimited genealogies, 4, 5, 6, generation pedigrees, family record sheet and much more.

FAMILY TREE, C-64/C-64C-1541 . . . . . $49.95*
FAMILY TREE, C-64/C-64C/1541 (LDS) $49.95*
FAMILY TREE, C-128/1541/1571 . . . . . $49.95*

Skyles Electric Works

*Due to trade games being played by U.S. and Japan *all prices are subject to change without notice.* There is an additional $4.50 U.S. and Canada, $15.00 Europe and Asia, shipping charge per order. California residents add sales tax.

TO ORDER: Call 1-800-227-9998 or 1-415-965-1735 (CA) or write to:
For More Information: Send Stamped Self Addressed Envelope to:
231-E South Whisman Road, Mountain View, CA 94041

Turbo BootMaker

Bert Rozenberg

This program makes "TurboDisk 64" even more useful. It creates a boot program that lets you load any program at turbo speed—without using TurboDisk. And after turboloading, it automatically runs. For the Commodore 64 and 1541 disk drive.

One of my favorite programs from COMPUTE!'s GAZETTE is "TurboDisk" for the Commodore 64. It greatly speeds up the loading time of the ordinarily slow 1541 disk drive. To enhance the use of this excellent utility, I wrote "Turbo BootMaker," which lets you autoboot your programs at turbo speed without loading TurboDisk.

Making The Boot

After typing in Turbo BootMaker, save a copy. To use it, load it and type RUN. (If you're using TurboDisk, be sure to *disable* it before you run Turbo BootMaker.) The screen prompts make the program very easy to use. First, insert your disk with a copy of TurboDisk. You're asked to enter the TurboDisk filename (the name you used to save it on your disk). Be sure to use an appropriate version of TurboDisk. The normal version works for turbobooting BASIC programs and some machine language (ML) programs. However, it cannot be used to turboboot ML programs that load at its own address, 49152 ($C000 hexadecimal). To turbo boot an ML program that resides at 49152, you must use a version of TurboDisk that resides elsewhere in memory. The "TurboDisk Relocator" program elsewhere in this issue lets you create versions of TurboDisk that can work from other memory locations. After you enter the filename, there will be a short delay as the TurboDisk file is read into memory.

You're then asked to insert the disk on which you wish to create the turbobooter file. The next prompt asks for the filename of the program you want to turboboot. (If you press RETURN only at this prompt, the wildcard (*) character will be used as the filename.) If a copy of that program is not already present on the disk, you'll be reminded to add a copy after you finish creating the turbobooter. Turbo BootMaker does not alter the program to be turbobooted; instead, it creates a file that automatically loads TurboDisk, then turboloads the specified program and starts it running. A copy of the program to be turbobooted must be on the same disk as the turbobooter file. That copy can always be loaded and run normally independently of its associated turbobooter file.

You're asked if the program to be turbobooted is BASIC or ML (machine language). Press B or M. If it's ML, you're asked for the starting address, which can be entered in either decimal or hexadecimal (you must precede hex values with a $). For example, if the command to start the program is SYS 49152, you can answer the starting address prompt with either 49152 or $C000. You should note that many of the major machine language programs published in the GAZETTE are designed to be loaded and run as if they were BASIC. Such programs usually start at address 2049 ($0801). For these programs you should select B (BASIC), not M (machine language), at this prompt. If you select M and specify 2049 or $0801 as the starting address, the program will crash after loading. As a rule, if you start a program with RUN, select option B; if you start it with a SYS, select option M.

Last, you're prompted to give a filename to the new turbobooting program (do not use the wildcard convention). It would be wise to use a filename which indicates that this is a turbobooter file, and which program it turboboots. For example, you might use the name SS.TBOOT for a file which turboboots *SpeedScript*. After you enter the filename, there will be another short delay as the turbobooter file is written to disk.

At this point, you've created a boot program that will automatically load the specified program at turbo speed every time you run it. It also runs the program after it's been loaded. You don't need to have TurboDisk on the same disk—all you have to do is LOAD *"filename"*,8,1 (be sure to include the ,1) where *filename* is the name of the turboboot file you created with the procedure described above.

There are a few things you should note about using the turbobooter file. First, the screen will appear to fill with garbage characters for a second or two after you enter the LOAD command. Don't panic; this is normal, and the screen is cleared when the turboload begins. Second, even though the turbobooting process usually is a good deal faster than a regular LOAD, it isn't quite as fast as turboloading directly with TurboDisk already in memory. This is because the turbobooter must load the TurboDisk machine language before it can turboload the program. However, you'll probably find that the convenience of having your programs start automatically outweighs the slight speed decrease. Finally, when turbobooting machine language programs like "MetaBASIC" that are designed to be used in conjunction with BASIC, you must remember to type NEW and press RETURN after the turboboot is finished so that important memory pointers will be reset. ©

TurboDisk Relocator

Dino Bavaro

"TurboDisk 64" becomes a more versatile and powerful utility when used with this short program. Now you can use TurboDisk 64 with most any machine language program. For the Commodore 64.

"TurboDisk 64" is an excellent utility program that radically speeds up the 1541 for load operations. Since TurboDisk may be activated in program mode, it can be incorporated quite easily with other programs. The only drawback is its location in memory—$C000 or 49152, quite a popular spot for other machine language programs. You may often find yourself forced to use a normal load because the program you want to turboload is at that same address.

Because of the numerous memory configurations available on the 64, many programs are written as *modules*. When the main program is executed, it in turn loads in other modules such as machine language routines, hi-res screens, and so on. All the modules required are loaded in only once when the program is first run. TurboDisk (in program mode) is also needed only once at the beginning of the module loading. By being able to relocate TurboDisk, we could put it in BASIC RAM (or wherever we want) and not worry about it being overwritten, because once the modules are loaded, we can deactivate it since it's no longer needed. Care, however, must be taken to relocate TurboDisk to an area of memory which does not conflict with the modules being loaded.

User's Choice

The program accompanying this article, "Turbo Relocator," allows you to select the area of memory for TurboDisk. The DATA statements in this program reflect the key memory locations that have to be altered whenever TurboDisk is relocated to a different area of memory.

When you run it, the program prompts you for a new program name to be assigned to the new version. I find it helpful to append the program's starting address to the end of the program name. If you were creating a TurboDisk version to be executable at location 4096, you could name it "TBDISK.4096". This makes things easier, especially if you have other versions of TurboDisk that execute at different locations.

The program prompts you to enter the memory address of where to assemble the new version of TurboDisk. You should respond with a decimal address on a full-page boundary (that is, an even multiple of 256). It's advisable not to select memory areas under ROM or other key memory pages such as addresses below 2048.

Once the program knows the address of where to locate the new TurboDisk version and the new name, it proceeds to read the original version from disk and write a new version modified for the specified starting address back to the same disk. (Be sure the name you use for the new version is not already used by another file on the disk.)

To activate the new version, use a SYS statement to the new starting address you selected. For example, if you created a version of TurboDisk relocatable to address 32768, then you would start it with SYS 32768 (in either program or direct mode). To deactivate the new version, SYS to the address three bytes beyond the new starting address. For the example of starting at 32768, you could use SYS 32768 + 3 (or just SYS 32771). To incorporate TurboDisk into one of your programs which loads modules, you can use the following technique.

```
1 ON A GOTO 3,5,6
2 A=1:LOAD "TBDISK.32768",8,1: REM
  LOAD TBDISK VERSION 32768
3 SYS 32768: REM ACTIVATE
  TURBODISK
4 A=2:LOAD "HI-RES SCREEN",8,1
5 A=3:LOAD "ML-SORT",8,1
6 SYS 32768+3: REM DEACTIVATE
  TURBODISK
7 REM REST OF PROGRAM
```

TurboDisk 128

Don Lewis

Are you using a 1541 disk drive with your Commodore 128? Here's a powerful utility that can reduce the time you spend waiting for programs to load by 300 percent or more.

If you've upgraded to a Commodore 128 from a VIC-20 or 64 and are still using a 1541 disk drive, you're probably envious of those fellow 128 owners whose 1571 drives can load programs in the blink of an eye. Perhaps you've used "TurboDisk 64" in 64 mode and wished for an equivalent speedup for 128 mode. Here's the answer: "TurboDisk 128," a new and improved version specifically for the 128/1541 combination.

TurboDisk 128 works only in 128 mode; you'll still need to use TurboDisk 64 in 64 mode. And the program works only with a 1541; it isn't useful in conjunction with a 1571. If you own a 1571 disk drive, you don't need a Turbo program while you're in 128 mode: The 128 and 1571 can use the fast serial transfer hardware built into both disk drive and computer, which is as much as eight times faster than a standard 1541—about twice as fast as TurboDisk.

But even if TurboDisk 128 doesn't permanently cure your desire for a 1571, it will make your life with the 1541 more bearable. In fact, once you start using TurboDisk, you'll wonder how you got along without it. TurboDisk turbocharges the loading process by a factor of three times or more. In fact, the longer the program, the more improvement you'll see. Like TurboDisk 64, the 128 version requires no modifications to your disk drive or computer. It loads programs saved in the usual manner; no special Turbosave is required. It works with most BASIC and machine language programs. It does not compromise reliability. And you can switch it on or off at any time by typing a single command.

To load TurboDisk, use a command of the form BLOAD "TURBODISK 128":SYS DEC("1300"). The message C128 TURBODISK ENABLED signals that you're ready for high-speed loading.

Turbo LOADS

Once TurboDisk is activated, no special commands are necessary. Just type LOAD *"filename"*,8 or DLOAD *"filename"* or BLOAD*"filename"* as usual. You'll be amazed at the difference.

One thing you'll notice immediately is that the red light on the disk drive doesn't come on at all during a turboload. Don't panic; this is normal. It's also normal for the 40-column screen to blank while TurboDisk works. When the program is loaded, the screen reappears unaltered.

You may occasionally find it necessary to deactivate TurboDisk and use a normal LOAD instead. For example, 1541 disk drives are prone to head alignment problems, so if you have a disk formatted on a drive other than your own, you may find that your drive has difficulty loading programs from it. You can switch off TurboDisk at any time without erasing it from memory by entering SYS DEC("1303"), or the equivalent SYS 4867. You should see the message C128 TURBODISK DISABLED. To be safe, it would be wise to include a BANK 15 before the SYS to ensure that the system is in its normal BASIC configuration. To reactivate TurboDisk, enter SYS DEC ("1300"), or the equivalent SYS 4864 (again, it would be wise to precede this with a BANK 15). You should see the message C128 TURBODISK ENABLED to indicate that turboloading is now available.

You'll also find it necessary to use the SYS to reactivate TurboDisk after pressing RUN/STOP–RESTORE, which effectively disconnects TurboDisk.

TurboDisk resides in the currently unused area of free memory starting at address 4864–5839 (hex $1300–$16CF), so it's completely safe from BASIC. However, this memory area is rapidly becoming popular with 128 machine language programmers and you may find other programs that use these locations. Such programs cannot be

used with TurboDisk because loading them will overwrite the TurboDisk program. Turbodisk also uses the block of memory at 3072–3327 ($0C00–$0CFF) as a buffer for the data read from disk. This area is the RS-232 input buffer, but since the 128 can't turboload and receive RS-232 input simultaneously, this dual usage should cause no conflict. However, you should be aware that some programmers use the RS-232 buffers for machine language routines. Such routines cannot be used with TurboDisk.

TurboDisk speeds up LOAD, DLOAD, BLOAD, and the monitor's L command, but it can't speed up SAVE or VERIFY. It also doesn't affect the speed of disk file handling with PRINT#, GET#, and so forth. It's not compatible with certain features of some programs and may not work with some commercial software.

How It Works

The machine language for TurboDisk is unusual in that only half of it works within your computer—the rest is actually executed within the 1541 drive itself. Commodore disk drives are *intelligent* units, containing their own microprocessors, RAM, and ROM. This means that they can be programmed for special effects, like turboloading.

During the brief delay you notice between the time you enter the load command and the time the drive starts spinning, 464 bytes of machine language are transferred from the computer to the drive's RAM. In the 128, this data is stored in locations 5376–5839 ($1500–$16CF). This required transfer before each turboload adds a certain amount of overhead time, which explains why TurboDisk gives less speed improvement for short programs.

The 128-resident portion of TurboDisk operates by changing the ILOAD vector at locations 816–817 ($330–$331) to point to itself, bypassing the normal LOAD routines in ROM. TurboDisk first checks to see whether a disk directory or a verify operation was requested. In either of these cases, control is returned to the ROM routines for normal processing. If a program load was requested, the routine adds the filename to the code for the disk drive portion, then transfers that data to the drive's memory.

The portion of TurboDisk in the disk drive uses routines in the drive's ROM to locate the desired program and read it from the disk sector by sector. To improve speed, routines like the one that turns on the red light are omitted, and only the essential ones are used. The 256 bytes of data from each disk sector are sent to a 256-byte buffer within the computer. As mentioned above, this buffer is at locations 3072–3327 ($0C00–$0CFF). TurboDisk sends data over both the DATA and CLK lines on the serial port, instead of just the DATA line as in normal serial data transfers. Thus, TurboDisk temporarily converts your serial bus into a two-bit parallel bus. When the entire 256 bytes from a disk sector have been transferred into the computer's buffer, data from the buffer is added to the program in memory while the drive is reading the next sector from the disk.

The Longer, The Faster

Despite a few limitations, TurboDisk is one of the most valuable general-purpose utilities a disk user can own. To discover exactly how fast it is, we ran some tests. The results, below, demonstrate how TurboDisk yields the most improvement with medium to long programs. (Results with different disk drives may vary.)

Program	Blocks	Normal LOAD (seconds)	Turboload (seconds)	Factor
Program 1	7	7	3	2.33
Program 2	16	13	4	3.25
Program 3	28	20	6	3.33
Program 4	55	40	10	4.00
Program 5	138	94	25	3.76

Note to readers outside North America: High-speed TurboDisk data transfers rely on precise timing, so the program may fail to operate on systems that use the European PAL video system instead of the North American NTSC system. The reason is rather technical—128s with PAL video use a slightly different microprocessor clock frequency. ©

TurboSave 128

Michael Henry

This easy-to-use utility makes all your saves at turbo speed—up to six times faster than normal—on the Commodore 128 with a 1571 disk drive. Other useful features are also included.

If you own a 128 and a 1571 disk drive, you're probably still amazed at the speed with which your programs load. Unfortunately, the 1571 is as slow as the 1541 when it comes to saving programs. "TurboSave 128" offers a solution. It allows you to save 64K of memory to a blank, formatted disk in 23 seconds—six times the speed of a normal save. Here's a chart showing the difference between TurboSave and the standard SAVE routine:

	TurboSave	Commodore SAVE
10 blocks	3½ sec	9½ sec
100 blocks	12 sec	60 sec

TurboSave is invisible to the operating system and the user, so it works equally well with all save commands—BASIC's SAVE, DSAVE, and BSAVE, and the monitor's S command. Except for its lack of support for the bug-ridden Save-with-Replace, TurboSave is completely compatible with Commodore's standard SAVE. And it will never create an unclosed *splat file*, either.

TurboSave is simple to use. Just load it with a statement like this:

```
BOOT "TURBOSAVE",B1
```

Don't forget to add the extra ,B1 at the end of the command. This causes the data to be loaded into bank 1 of the 128's RAM. If you omit the bank specification, the program will load into bank 0 by default, where it will overwrite important system information such as function-key definitions. As an alternative, you can type in and save this short program to act as a boot program:

```
10 SCNCLR:PRINT"{DOWN}BOOTING
   TURBOSAVE"
20 BOOT"TURBOSAVE",B1
```

Pressing RUN/STOP-RESTORE disables TurboSave. You can also disable TurboSave with the following command:

```
BANK 1:SYS DEC("FFA8")
```

To reactivate it, enter:

```
BANK 1:SYS 12*256
```

TurboSave assigns this command to the SHIFT-RUN/STOP key combination, so you can also reactivate TurboSave simply by pressing SHIFT-RUN/STOP.

TurboSave causes no conflicts with BASIC, but machine language programmers should be aware of the memory locations used by the program. It occupies locations 3072–5563 ($0C00–$15BB) in block 1 RAM (bank 1), and uses locations 1024–1791 ($0400–$06FF) in that block for buffer storage. In block 0 RAM (bank 0), the program uses locations 3072–3327 ($0C00–$0CFF). This block is normally used as the RS-232 input buffer; any programs that use this area won't work with TurboSave. The following zero-page locations are used temporarily during a save operation:

6–8	($06–$08)
100–101	($64–$65)
106–110	($6A–$6E)
174–175	($AE–$AF)
193–194	($C1–$C2)

TurboSave eliminates automatic verifying to achieve its speed. If you wish to verify that your program was saved correctly, use the VERIFY statement following the SAVE. The VERIFY will occur as quickly as the SAVE did, so you'll still be way ahead of the normal save routine.

Altering TurboSave

TurboSave is designed to be flexible. Although it normally saves data in PRG files, you can change this with the following:

```
BANK 1:POKE DEC("D29"),x
```

where *x* is one of the following the values:

Value	File type
129	sequential (SEQ)
130	program (PRG)
131	user (USR)
132	relative (REL)

Add 64 to these values if you wish the file to be automatically locked after it is saved. For example, a value of 194 (130 + 64) would cause TurboSave to create locked program files. TurboSave normally writes the starting address of the data being saved as the first two bytes of the file. This is standard practice for Commodore program files, but you may want to disable this feature if you are creating another file type. To prevent the load address from being saved, use this statement:

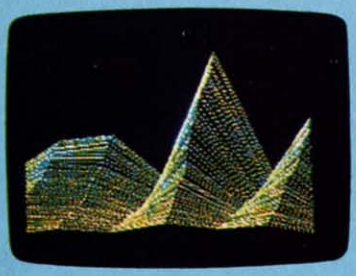
```
BANK 1:POKE DEC("D47"),0
```

This is especially useful when creating text files, which need no load address. However, remember that program files can't be loaded normally unless they contain a starting address. You can reenable the starting address feature with

```
BANK 1:POKE DEC("D47"),1
```

GET UP TO 200 FUN-FILLED PROGRAMS EACH YEAR–

when you subscribe now to COMPUTE!'s Gazette



Subscribe to COMPUTE!'s Gazette today through this special introductory money-saving offer, and you'll be getting a lot more than just another computer magazine. That's because each big issue of COMPUTE!'s Gazette comes complete with up to 20 all-new, action-packed programs.

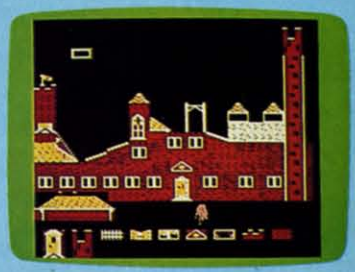
Subscribe now and you can depend on a steady supply of high quality, fun-filled programs for your Commodore computer—programs like Number Quest, Address File, Treasure Hunt, Castle Dungeon, Vocab Builder, SpeedScript and hundreds of other educational, home finance and game programs the entire family can use all year long.

The superb programs you'll find in each big issue are worth much, much more than the low subscription price. But there's more to COMPUTE!'s Gazette than just exciting new programs.

Month after month, COMPUTE!'s Gazette's superb articles deliver the latest inside word on everything from languages to interfaces...programming to disk drives.

Whether you're a novice or an experienced user, COMPUTE!'s Gazette is the magazine for you. So subscribe today. Return the enclosed card or call 1-800-727-6937.

Do it now.



ACT NOW AND SAVE!

COMPUTE!™ Publications, Inc. abc

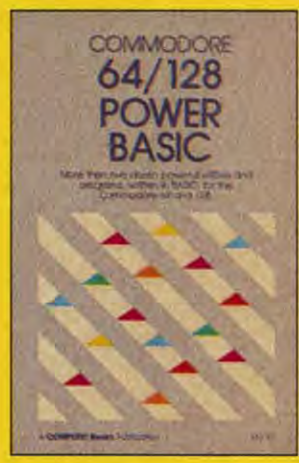
One of the ABC Publishing Companies

If attached order card is missing, write: COMPUTE!'s Gazette P.O. Box 1095 [illegible]

Exciting & New!

Commodore 64/128 Books

from COMPUTE!



Get more power, education, and relaxation from your Commodore 64 or 128 with these new books from COMPUTE!. And, as always, the programs are thoroughly tested and easy to enter using our special program entry system. (Disks are available, too.)

COMPUTE!'s Commodore 64/128 Power BASIC
Edited
Collected in this one volume are more than two dozen short BASIC programs from *COMPUTE!'s Gazette*'s popular monthly column, "Power BASIC." Included are utilities that add an 8K RAM disk; read and write to disk sectors; make programs read joysticks more quickly; and automatically generate line numbers. There are also programs such as "Stop and Go," which creates a pause button to temporarily halt a program, and "Time Clock," which puts a digital clock on your screen. *There is a companion disk available for $12.95 which includes all the programs from the book.* (998BDSK)
$16.95 ISBN 0-87455-099-8

COMPUTE!'s More Machine Language Games for the Commodore 64
Edited
Seven of the best machine language games for the 64 have been gathered into one volume in this follow-up to the popular *COMPUTE!'s Machine Language Games for the Commodore 64*. Selected from recent issues of *COMPUTE!* and *COMPUTE!'s Gazette*, the games range from the frantic "Prisonball" to the delightful "Biker Dave." This is more than just a collection of exciting fast-action games, though, because complete and commented source code for each program is included in the book. Machine language programmers can see exactly how each game is written and what design techniques are used. *A disk is available for $12.95 which includes all the programs in the book, including source code.* (947BDSK).
$16.95 ISBN 0-87455-094-7

COMPUTE!'s Third Book of Commodore 64 Games
Edited
COMPUTE! Publications, the leading publisher of programs for the Commodore 64, has brought together another exciting collection of nerve-tingling games that will delight the whole family—from preschoolers to teenage arcade fans to those who enjoy games of logic. Clearly written, with non-technical instructions, this book contains hours of challenging entertainment for beginning computer fans as well as experienced programmers. *There is a companion disk available for $12.95 that includes all the programs in the book.* (955BDSK).
$15.95 ISBN 0-87455-095-5

These books are now available at your local book or computer store.
You can also order directly from COMPUTE! by calling toll free 800-346-6767
(in NY call 212-887-8525) or mailing your order to COMPUTE! Books, P.O. Box 5038,
F.D.R. Station, New York, NY 10150.
Customer Service hours are 10:00 a.m.–12:30 p.m. and 1:30 p.m.–3:00 p.m. EST Mon.–Fri.

Please include $2.00 postage and handling per book or disk. NC residents add 5 percent sales tax and NY residents add 8.25 percent sales tax. Please allow 4–6 weeks for delivery.

TurboSave 64

William Voosen

Now you can save files three-to-five times faster on the 1541 disk drive. (Commodore 128 users see the following article, "TurboSave 128," a version that works with the 1571 disk drive.)

"TurboSave 64," is a machine language program that makes saves three to five times as fast by reprogramming the 1541 disk drive. Here's a chart showing the speed difference between "TurboSave 64" and the standard SAVE routine:

	TurboSave	Standard SAVE
8 blocks	3 sec	9 sec
128 blocks	17 sec	87 sec

Once TurboSave is activated, all saves take place at high speed. To load the utility, type LOAD-"TURBOSAVE 64",8. To activate TurboSave, just type RUN as you would for a BASIC program. Pressing RUN/STOP–RESTORE disables TurboSave. Enter SYS 710 to reenable it.

This utility maintains all the error checking of the standard SAVE routine except automatic verification. If you wish to verify the save, use the VERIFY command. To increase the speed of the save, the screen is turned off briefly during the high-speed save.

Compatibility

Some printer interfaces interfere with TurboSave 64. If your printer interface causes problems with TurboSave, you may have to disconnect it while using the program. Be sure to test the program carefully before using it with an important disk.

TurboSave 64 is compatible with the DOS 5.1 wedge. The program uses 27 bytes at locations 710–736 ($02C6–$02E0), but the bulk of the program is stored in the RAM under the BASIC ROM. Because TurboSave reprograms the disk drive, a portion of the program is also transferred into the disk drive's internal memory. The transfer of data to the drive RAM requires about 1.5 seconds. That overhead is the same for all saves, so there is less speed improvement for saving small programs than for saving large ones. ©

Turbo Format

Ross Ouwinga

If you're interested in saving time, and wear and tear, on your 1541 disk drive, you'll put this program high on your list of utilities. It formats a disk in only 9-1/2 seconds—a super speed-up from the usual minute and 20 seconds—and it eliminates head knocking. Also, it's easy to use—just answer a few prompts, and you're seconds away from a formatted disk.

"Turbo Format" is a fast, easy-to-use disk formatting program. It formats a disk in only 9-1/2 seconds—the standard 1541 format program requires one minute and 20 seconds. That's a speed increase of over eight times. It's easy to use because you don't have to remember the cryptic command line required by the 1541 to format a disk. And you're prompted through each step of the procedure so there's no need to memorize anything.

To use Turbo Format, type LOAD "Turbo Format",8 and RUN. There will be a short delay (while most of the program is transferred to the memory in the disk drive), and a title screen with the first prompt will be displayed.

Three Easy Steps

You're first prompted to enter the disk name (up to 16 characters). Virtually any character may be used in the name, but some may cause undesirable side effects, so caution is recommended whenever characters other than letters or numbers are used.

The second prompt is for the disk ID. This is the number used by the disk drive primarily to determine when a disk has been changed. Normally this number is two characters long, but Turbo Format will allow up to five characters. More than five characters may be typed in, but only the first five will be used. Only the first two characters will actually be used by the disk drive to identify the disk after it has been formatted, but all five will appear in the BAM (Block Allocation Map).

Next, you'll be instructed to insert the disk to be formatted. *Be sure that you do not leave your disk with Turbo Format in the disk drive.* Formatting the disk will erase everything. A recommended safeguard is to put a write-protect tab on your disk immediately after you have a working copy. Turbo Format does check the write-protect tab, and if one is present, the disk will not be formatted. Also, an error message will appear to notify you that the disk is write protected.

Along with the prompt to insert a disk is an abort option. Press the f1 key to abort. This causes the program to start again at the beginning, prompting you to re-enter the disk name. This option allows you to change your mind and enter a new disk name or ID or correct any spelling errors before formatting the disk.

Two Kinds Of Formatting

The formatting process does not actually begin until you press the space bar or RETURN. Note, however, that the space bar and RETURN do not perform the same function. If you press the space bar, the disk will be formatted and all data written to the disk will be checked for errors. If an error occurs, the formatting will stop and the type of error will be displayed on the screen. If you press RETURN, the disk will be formatted as quickly as possible without checking for errors.

The reason for offering the two options is to allow you to use the program in a manner which most closely suits your needs. If the disks are verified, the procedure takes 17 seconds rather than 9-1/2 seconds. Verifying provides more assurance that the disk is formatted correctly. The additional 7 or 8 seconds may not make much difference to you. On the other hand, if speed is important, you may choose not to verify your disks. Errors occur very rarely and most are self-correcting, provided the disk itself is not damaged. Also, if you have a disk which you know is faulty, but wish to format anyway, you must use the no-verify option. I strongly recommend using the verify feature if you're formatting the back side of a disk certified only for single-side use, or if the disk is well used, or if you use an inexpensive, off-brand disk.

No Knock

After pressing the space bar or RETURN to format, the process begins. A feature that will be noticed immediately is that there is no loud knocking noise at the beginning of the format routine. A single short click is heard instead to indicate that the routine is working correctly. This feature is possible since Turbo Format is loaded from a disk at the start and the exact location of the read/write head is saved in memory in the disk drive. The program is then able to calculate the exact position of the first track without the excessive knock. The standard format routine does not necessarily know the position of the read/write head; it therefore assumes the worst-case condition

and tries to step the head down 45 tracks. If the head reaches track one before it has stepped 45 times (which is usually the case), it bangs the head against the stop until the count of 45 is complete. It works, but it makes an awful racket and, in some cases, eventually works the disk drive out of alignment.

When the format process is complete, a message is displayed and will indicate OK if there were no errors. At this point you have the option of formatting another disk or quitting. It's important that you press f1 to quit. Do not reset the computer by turning it off and back on. The reason for this is that the format routine leaves the read/write head on track 35 when it finishes. By using f1 to quit, the disk drive is sent an initialize command and also a reset command to set everything back in order. If you accidentally turn the computer off, you should initialize the disk drive using the IO: command. Failure to do so could cause erratic operation. If you accidentally restarted when you intended to quit, it will be necessary to press RETURN for the enter name and ID prompts and press the space bar with no disk in the disk drive. An error will be detected immediately and you will again be given the prompt to quit by pressing f1.

Using Turbo Format does not mean your disks will become less reliable. It's much faster because it eliminates several very unnecessary and time-consuming routines. Most of the time used to format a disk is spent by a routine whose only function is to space the sectors evenly around the disk. Turbo Format sets this space the same for every sector on every track, but skips a time-consuming calculation process used in the standard formatting process. It's amazing how much time is saved by this one modification. To a lesser extent, time is saved by reducing delays when stepping between tracks and by eliminating unnecessary data-conversion routines. A technical discussion of this is beyond the scope of this article, but it should be noted that only those procedures that do not affect reliability were changed. Disks formatted with Turbo Format work just as well as those formatted the standard way.

Intermediate or advanced machine language programmers might be interested to know that the data recorded to the BAM of the disk is at locations $0B00 (2816) through $0BFF (3071). If you have a thorough understanding of the BAM and its contents, you can modify this area of the program to customize or personalize the BAM on your disks. Be aware, however, that modifying the BAM may have undesirable side effects and is not recommended for the beginner or someone who knows no more than the basics of disk drive operation.

For those of you with disk drives other than the 1541, Turbo Format will likely work if the drive claims to be 100-percent compatible. Most of the program is transferred to and operates in the disk drive and calls on several routines which are part of the standard program in the drive. If these routines are not at the same addresses as in the 1541, the program will not operate correctly. No damage will occur to your disk drive if this should happen, but it may be necessary to turn it off and on again to regain control. ©

Unicopy
Single Disk Copying For The Commodore 64

Jim Butterfield, Associate Editor

Copying programs and sequential files can sometimes be difficult. There are backup programs, of course; but they require that you copy the whole disk, and sometimes you just want to copy one or two programs. LOAD and SAVE work for simple BASIC programs, but not for anything complex. Well-equipped users have two disk drives and can use utilities that will transfer from one to the other.

Dual disk units (notably the Commodore 4040) don't quite do the job for Commodore 64 owners. There's a slight format difference between 4040 and 1541 that makes it undesirable to write on a disk formatted on the other unit. Thus, a 4040-copied disk is not well-suited for the 1541 if you wish to write further material to the disk.

"Unicopy" will help to solve these problems. It will take your choice of programs or sequential files from the disk and hold them in memory. When you're finished, or when memory is full, you can then write the files to a new disk or to cassette tape.

Buffering Files

As you scan the input disk, you'll be offered programs and sequential files for copying. You may tap the Y or N key to signal: yes, you want to copy this file; or no, you don't want this one. More on this in a moment.

Before presenting you with the filename, Unicopy looks at the size of the file and the amount of memory space that is left. If the file is too big to fit, the program won't offer it to you; instead, it will signal MORE and quit scanning the directory. That way, you know that there is more on the disk, but it can't be fit in this time. Unicopy could catch the missing files on a subsequent run. The program will not skip by the big ones to look at smaller files later in the directory, since it might be important to you to keep a group of files in the same order as they were shown on the previous disk.

There's one exception to the big files rule. If the file won't fit into the whole memory buffer area, there's no way to copy it with this program. Unicopy will skip such jumbo files.

How big is the buffer area? It depends on how your Commodore 64 is configured. If it's a "clean" system with no other resident programs, Unicopy will throw out BASIC (temporarily) and use all the memory it can get—about 48K, or the equivalent of 192 disk blocks. On the other hand, a program in residence—monitor, DOS wedge, interface package, or whatever—must not be disturbed; in this case, UNICOPY will become conservative and restrict itself to less than 36K or 144 disk blocks. The exact amount of space will depend on the other program's location and size.

You'll be asked "Any resident programs?" at the start of Unicopy; buffer size will be set accordingly.

One more thing: If you plan to direct the output to cassette tape, Unicopy must insure that no more than 28K or 112 blocks are used. Tape routines forbid writing a program from above address 32766; so Unicopy will trim accordingly. By the way, this solves a subtle problem with cassette tape: Normally, you can never save memory above hex $7FFE; but Unicopy will move the program down and save it successfully from where it is held in lower memory.

Initial Tasks And First Questions

If you're copying to another disk, be sure that the destination disk is formatted before running Unicopy. The disk may already be formatted if you're just adding some new files; but if not, remember to NEW it, since Unicopy won't do the job for you. You may copy files to more than one disk; format them all as necessary.

We've already mentioned ANY RESIDENT PROGRAMS?; answer Y or N.

OUTPUT TO TAPE OR DISK? calls for touching the T or D key.

If you select tape, you'll be asked WRITE END-OF-TAPE MARK? If you respond with N, you'll copy the programs to tape and that's all. If you answer Y, you'll copy the programs to tape and then write a special block called a "tape mark." Here's what the tape mark does: At some later time, if you're searching through this tape for a particular file, the tape mark will stop the search.

Thus, if you don't find the file you want on tape, you won't go running through the whole tape, most of which is blank. An unsuccessful search will terminate early, thanks to the tape mark.

DISK INPUT PATTERN? allows you to use pattern matching. You're prompted with the asterisk; if you want to see everything, just press RETURN. But there are many other combinations. AR* will present only programs that begin with AR, such as: AR, ARCHER, ARM, or ARRRRGH. R?D? will present you with such names as REDS, RIDE, or R2D2, but not RIDDLE. *=P will offer programs only. And

PLUTO will offer you only a file called PLUTO.

When you are presented with names from the directory, you may tap Y or N to accept or reject the files for copying. If you know that you want to take or reject a sequence of files, you may hold down the appropriate key. The RETURN key acts to "lock in" the previous key, so that pressing Y, RETURN will accept everything and N, RETURN will reject everything.

The Output

Eventually the questionnaire will stop, and the computer will advise READING FILES. The programs or sequential files will be brought into the buffer area. The programs won't be in their usual place in memory, but that doesn't matter; we just want to copy them, not to run them.

After the files are loaded, the computer will say READY TO WRITE FILES; PRESS ANY KEY. Don't press a key—yet.

If you're writing to cassette tape, place the tape in the drive. If you have time and think it's necessary, fast forward and rewind the tape to even up the tension. Finally, press PLAY and RECORD and touch any key on the keyboard. The tape will start to write: The screen will go blank, of course.

If you're writing to another disk, take the source disk out of the drive and insert the destination disk. (You did make sure that the destination disk was preformatted, didn't you?) Now touch any key on the keyboard. As the files write to the disk, you'll see their names displayed.

If any errors are encountered during input or output, you'll be told about them.

When the copying job is done, you'll be asked ANOTHER OUTPUT? If you want to write to another tape or disk, put it into the drive and press Y for "yes." Otherwise, press N and the job is done.

Other Types Of Files

Unicopy does not attempt to copy USR or REL type files, nor does it try to copy "direct" data. This type of job should be done by the programs which use these types of files. ©

Advertisers Index

"...dramatic improvements over 1541 in quality and reliability... while keeping 1541 compatibility."
COMPUTE!'s *Gazette* August '87

DRIVE THE BEST

NOTHING DRIVES THE COMMODORE 64 OR 64C BETTER THAN THE FSD-2 EXCELERATOR+ PLUS.

COMPATIBLE. RELIABLE. GUARANTEED.

*The FSD-2 Excelerator+Plus Commodore compatible disk drive. Faster, quieter and more reliable than the 1541 and 1541C. Breakthrough Direct Drive technology. Guaranteed 100% compatibility. It even **enhances** GEOS! Full One-Year Warranty.*

ECI

Emerald Components International
P.O. Box 1441/Eugene, OR 97440
In Oregon: 683-1154/Fax: 503-345-5037
Telex: 23-499-6100 EUG SECSVC

1-800-356-5178

*Commodore is a registered trademark of Commodore Business Machines, Inc.

WORK WITH US ON A PROFESSIONAL LEVEL.

Some people really like working on their machines. And then there are some who prefer to do their tinkering under the hood. For those of you who can't wait to get your hands greasy, we proudly introduce geoProgrammer. The most sophisticated machine language programming tool on the market.

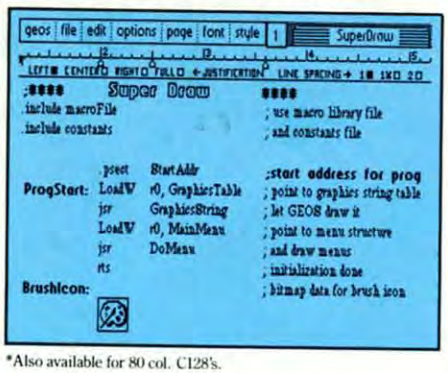
With geoProgrammer, you get the same kind of technology we use for developing our other products. Which means you can write supercharged software. Pump your own programs. And assemble just about any kind of application you can imagine.

It reads and writes and stomps on bugs.

The first three tools that any serious programmer needs are an assembler, a linker and a debugger. So we've installed all three, complete with sample GEOS applications that teach you the latest programming tricks in seconds.

geoAssembler not only reads directly from geoWrite files, but contains enough Pseudo Ops to allow all kinds of conditional assemblies. It has all the state-of-the-art features you'd expect, including some you probably never thought possible. For example, integrating graphics is as simple as cutting and pasting the image from geoPaint directly into your program.

geoLinker ties your program modules together, supporting GEOS SEQ and VLIR applications and desk accessories. You can even use geoAssembler and geoLinker to create non-GEOS applications.



*Also available for 80 col. C128's.

geoDebugger allows your program to be tested in memory with full symbolic disassembly, along with line assembly for patching code in memory. It also allows your code to be single-stepped or top-stepped, with sub-routines fully executed. It can stop a running program with one key, or use up to eight conditional breakpoints. When your program hits the breakpoint, it prominently displays the error in an overlay window, leaving the applications screen intact.

Pro enough for you yet? Well, that's only a *partial* list of what you're in for.

If you understood all that, read on.

Chances are that we lost a lot of readers by now. But if you're still with us, hang in there. You have the makings of a real GEOS pro. All you need now are the right tools.

And all of them come in this one handy box.

So if you're serious about programming, consider geoProgrammer. After all, you've got nothing to lose —except your amateur status.

To order call 1-800-443-0100 ext. 234

geoProgrammer $69.95

(California residents add 7% sales tax.)

$2.50 US/$5.50 Foreign for shipping and handling. Allow six weeks for delivery.

GEOPROGRAMMER

GEOS COMPATIBLE

Berkeley Softworks

The brightest minds are working at Berkeley.