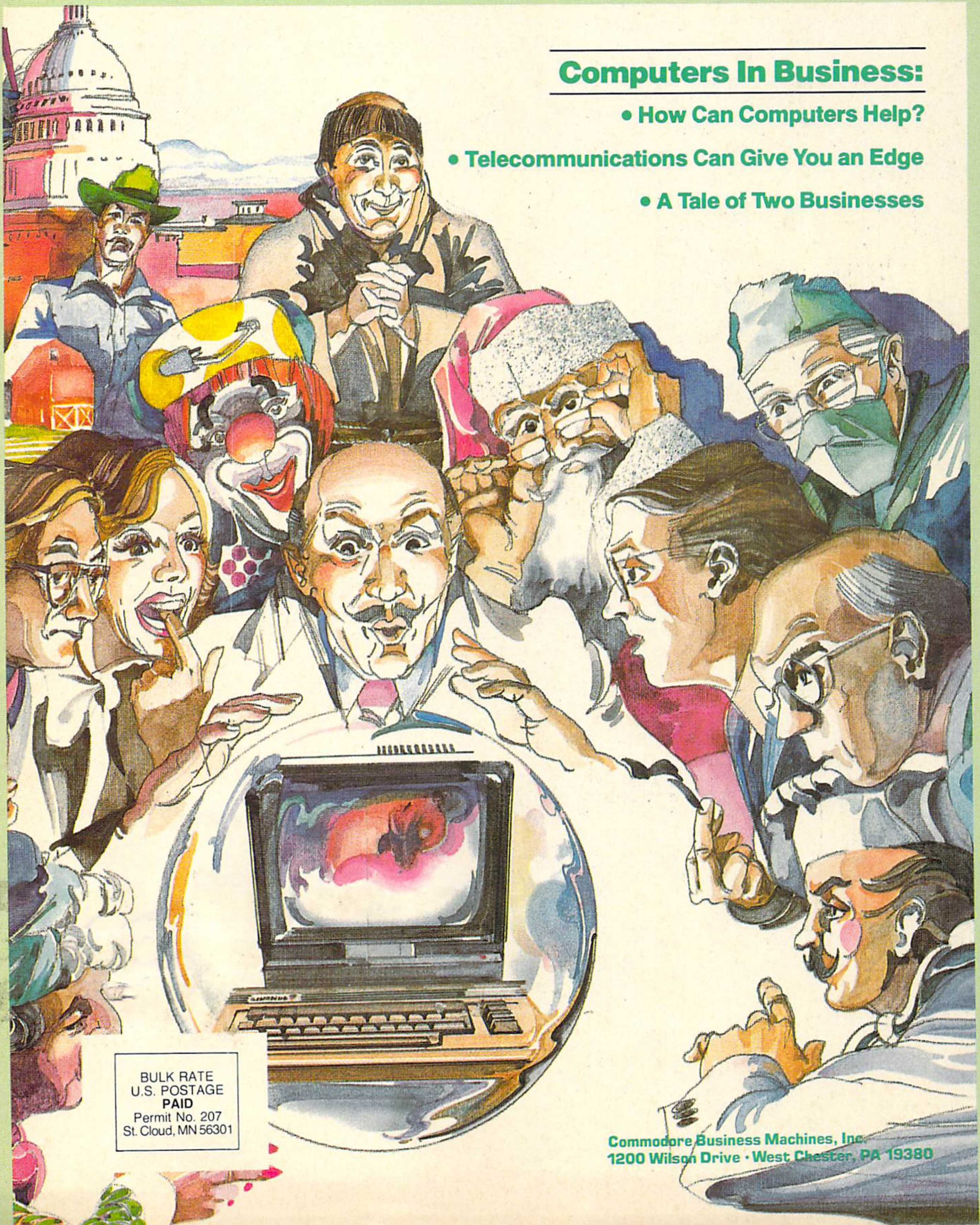
commodore
the microcomputer magazine

Volume 4, Number 4, Issue 25
$2.50 U.S.
$3.50 Canada



Computers In Business:

- How Can Computers Help?
- Telecommunications Can Give You an Edge
- A Tale of Two Businesses

BULK RATE
U.S. POSTAGE
PAID
Permit No. 207
St. Cloud, MN 56301

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

King of the mountain!

Workhorse solutions for tough questions.

When **Southern Solutions** acquired the exclusive marketing rights for the CMS Accounting System, the first (and the best) accounting system for the Commodore computer, we offered dealers who were dissatisfied with their current accounting software the opportunity to swap ... ours for anyone else's.

WOW! We were covered with the others ... MAS, BPI, EBS, etc ... all trading for CMS. We provide the only complete coverage of real software for Commodore computers:

THE PREMIER ... SYSTEM IV. Real accounting. More like a mini, yet priced for the Commodore. SuperMath™ gives precision to **$1 billion.** No one else comes close. General ledger, accounts receivable, accounts payable, payroll, inventory, mailing list. Plus important vertical products: oil accounting, pharmacy management, encumbrance accounting, church records and more.

THE STANDARD ... SYSTEM III. Similar to System IV but lower priced. G/L, A/R, A/P, P/R, mailing list.

Peripherals. Monitors, monitor cables, blank cassettes.

All software has FileGuard™. Never lose data files, EVEN IF YOU LOSE ELECTRICITY! Compatible with almost any computer, disk drive and printer combination. User-definable reports. Fast file access.

Sold only through professional computer dealers.

To become a **Southern Solutions** dealer, or for the name of your nearest retailer, call or write our General Manager, Bill Swingler.

Dealer Hotline: 1-800-527-4548

*Commodore 64 is a registered trademark of Commodore

OFFICIALLY APPROVED BY COMMODORE

Available in Canada through
Canadian Micro Distributors Ltd.
500 Steeles Avenue
Milton, Ontario, Canada L9T 3P7

Southern Solutions
P.O. Box P, McKinney, Texas 75069 - (214) 542-0278

Don't let price get in the way of owning a quality printer.

Adding a printer to your computer makes sense. But deciding which printer to add can be tricky. Do you settle for a printer with limited functions and an inexpensive price tag or buy a more versatile printer that costs more than your computer? Neither choice makes sense.

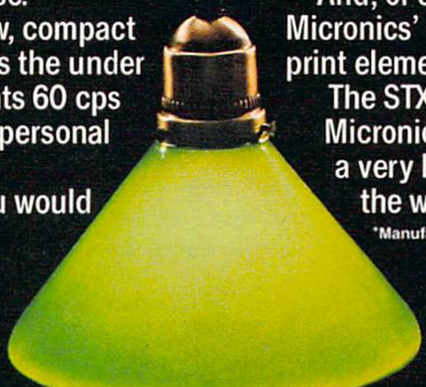
Here's a refreshing option—the new, compact STX-80 printer from Star Micronics. It's the under $200 printer that's whisper-quiet, prints 60 cps and is ready to run with most popular personal computers.

The STX-80 has deluxe features you would expect in higher priced models. It prints a full 80 columns of crisp, attractive characters with true descenders, foreign language characters and special symbols. It offers both finely detailed dot-addressable graphics and block graphics.

And, of course, the STX-80 comes with Star Micronics' 180 day warranty (90 days on the print element).

The STX-80 thermal printer from Star Micronics. It combines high performance with a very low price. So now, there is nothing in the way of owning a quality printer.

*Manufacturer's suggested retail price.



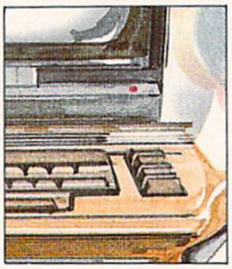
star™
micronics • inc
THE POWER BEHIND THE PRINTED WORD.
Computer Peripherals Division, 1120 Empire Central Place, Suite 216, Dallas, TX 75247 (214) 631-8560

The new STX-80 printer for only $199.*

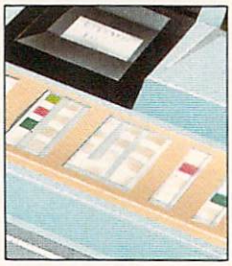
commodore
the microcomputer magazine

features

Volume 4, Number 4, Issue 25


Can Computers Help?

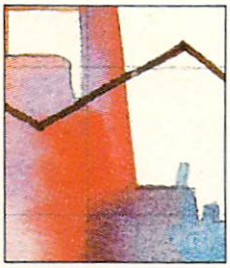

Two Businesses


Cash Register Programs

departments

commodore
the microcomputer magazine

Telecommunications



Simulation

staff

Publishing Manager
Neil Harris

Editor
Diane LeBold

Technical Editor
Jim Gracely

Staff Writers
Jeff Hand
April Koppenhaver
Walt Kutz
Stephen Murri
Mark Odgers
Michael Tomczyk

Contributing Writers
Robert W. Baker
Frank Covitz
Elizabeth Deal
Doris Dickenson
Don Hassler
John Heilborn
C.D. Lane
Larry Modrell
Alexandra Muller
Allen Patterson
Joe Rotello
Gregory Yob

Technical Staff
Paul Goheen
Pat McAllister
Cyndie Merten
Sarah Meyer
Dave Middleton
Tom Rizol, Jr.
David Scott

Circulation Manager
John O'Brien

Circulation Assistant
Kathy Reigel

Advertising Coordinator
Sharon Steinhofer

Graphic Design
Neumann Greenberg Schlenker

Cover Art
Robert Neumann

Printing
Volkmuth Printers

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

Commodore: The Microcomputer Magazine is published six times a year by the Computer Systems Division, Commodore Business Machines, Inc., 1200 Wilson Drive, West Chester, Pennsylvania 19380. Copyright 1983© by Commodore Electronics Ltd. All rights reserved. No material may be reprinted without permission. Volume IV, Number 4, Issue 25.

Subscription Information: U.S. subscriber rate is $15.00 per year. Canadian subscriber rate is $20.00 per year. Overseas $25.00. To order phone 800-345-8112 (In Pennsylvania 800-662-2444).

The Best is Yet to Come! Watch for These Issues!

Power/Play Fall: Our Adventure Game Special, featuring the sophisticated Zork games and Scott Adams' graphic adventures for the Commodore 64. We'll be out there with this exciting issue in mid-September.

Commodore Issue 26: Software, software and more software. Commodore's new Software Division is off to a terrific start. Find out what they're up to in this issue, on the stands in early October.

We Need Articles About Commodore CBM 8032/8096 Systems!

If you're using or programming our 8032 computers and have some information to share, let us know and we'll send you our Guidelines for Writers. Send your request to: Commodore Magazine, 1200 Wilson Drive, West Chester, PA 19380, Attention Guidelines for Writers.

Advertising Policy

Advertisements within this magazine are presented for the information of our readers. Acceptance of any ad does not constitute an endorsement by Commodore Business Machines. We stand by only those products manufactured by us. In addition, products displaying the "Commodore Approved" logo have been tested by Commodore.

If any product advertised herein fails to perform as advertised, or you are unable to resolve a problem with an advertiser, please bring the matter to our attention. Write to Commodore Publishing Group, 1200 Wilson Drive, West Chester, PA 19380.

DEVELOP-64

FIVE POWERFUL SOFTWARE DEVELOPMENT TOOLS

Plus The Exciting New Book

INSIDE THE COMMODORE 64™

THE BOOK

A complete clear explanation of machine language, Assembly language, Commodore 64 architecture, graphics, joystick and sound effect programming. Detailed step-by-step guide to the use of the development tools. How to combine BASIC and machine language, make auto-start cartridges, interface with the internal ROM-based programs of BASIC and the Kernal. Sample programs fully explained.

THE TOOLS

Assembler/Editor/Loader/Decoder/Monitor
Full-featured Assembler allows use of labels, comments and arithmetic expressions to create machine language programs. Create, save, modify Assembly language programs with the Editor. Load and link machine language modules with the Loader. Decode machine language back into assembly language for study or input to the Editor. Single-step program execution with the Monitor. Combines Assembler/Editor for maximum ease of use.

ALL FOR $54.95 PLUS $2.00 POSTAGE AND HANDLING Add $5.00 for disk version.

Send check, M.O., VISA/MC ($2.00 S.C.) or specify C.O.D. (add $3.00) to:

French Silk smoothware

P.O. Box 207, Cannon Falls, MN 55009

507-263-4821

Commodore 64™ is a registered TM of Commodore Business Machines Inc.

FOR YOUR COMMODORE WORDPROCESSING NEEDS INVEST IN THE BEST

WORDPRO PLUS. IN A CLASS BY ITSELF.

When choosing a Word Processor for your Commodore™ computer, there's no reason to settle for anything but the best — in a word. . . WordPro™

With over 30,000 happy clients churning out letters and documents all over the world, the WordPro Plus™ Series is unquestionably the #1 selling software package on Commodore computers! So when you choose WordPro, you know you're investing in a trial-tested program that's a real winner. And WordPro is NOW available for your Commodore 64™ computer— at prices starting as low as $89.95.

Designed for the user who has no computer or word processing experience whatsoever, WordPro Plus brings a new dimension to the term "user-friendly." More than just easy to use, WordPro will turn your Commodore computer into a sophisticated time saving word processing tool — loaded with the same inventory of features found in systems costing much, much more.

Our nationwide team of over 600 Professional Software/Commodore computer dealers will help you choose the WordPro Plus system that is best for your needs. Our full-service dealers have been set up to provide strong customer support. In addition to helping you choose the right system, many Professional Software dealers also offer WordPro Plus training and system installation.

Professional Software offers a complete spectrum of WordPro word processing software for Commodore computers ranging from the Commodore 64 to the more business oriented 8000/9000 series computers. And WordPro 4 Plus and 5 Plus also interact with our database management systems including InfoPro and The Administrator. So whatever your Word Processing needs, there's a WordPro system that's right for you.

Invest in the best. . .
WordPro Plus. In a class by itself.

Call us today for the name of the WordPro Plus dealer nearest you.

Professional Software Inc.

51 Fremont Street
Needham, MA 02194
(617) 444-5224
Telex: 951579

WordPro™ and WordPro Plus™ are trademarks of Professional Software Inc.
The WordPro Plus Series was designed and written by Steve Punter of Pro Micro Software Ltd.
Commodore™ and the Commodore 64™ are trademarks of Commodore Electronics, Inc.
Dealer inquiries invited.

editor's notes

For those readers who have had a hard time reading our dot matrix program listings, GOOD NEWS! With this issue we begin using a new system. First, listings are now run off on a letter-quality printer. That in itself should help a lot. But the best part is that, instead of marginally readable graphic characters, you'll find WORDS (very readable words) in brackets. Just press the keys indicated by the bracketed words. You'll get the appropriate character on your screen and much-improved results when you run the program. These are the translations:

```
[HOME] = CLR/HOME
[CLEAR] = SHIFT CLR/HOME
[DOWN] = CURSOR DOWN
[UP] = CURSOR UP
[RIGHT] = CURSOR RIGHT
[LEFT] = CURSOR LEFT
[RVS] = REVERSE ON
[RVOFF] = REVERSE OFF
```

If you're convinced we're a bunch of sadists for ever having run dot matrix in the first place, I'd like to offer an explanation. The only reliable program listing is one that comes right off a tape or disk we know runs. Before we got our hands on Jim Butterfield's translation program the only way to print a hot-off-the-disk listing and still get those graphic characters was to use a dot matrix printer. The alternative was to typeset the programs and draw each character in. Talk about mistakes—let me tell you, that would have been a disaster! So, we chose the lesser of the two evils.

Now on to computers in business—our theme this issue. I thought I'd say a few things about Commodore's new "B" series advanced business computers—another of our products that is simply going to blow away the competition. It's going to be very hard for anyone to come up with anything close for the money: a big 128K or 256K RAM, 80-column screen and the capability to run not just CP/M, but MS DOS and CC-CP/M86 as well. Not to mention the tilt-swivel monitor, classy looking case, comfortable keyboard, numeric keypad and built-in music synthesizer. All for a suggested retail price that makes the competition look pretty silly. If you find a better deal (through legitimate channels) let me know.

We got such a good response to the list of educational software we ran in the May issue (Volume 4, Number 2) we thought it would help our business users to run a list of business software in this issue. Finding really comprehensive lists of software for any given computer is a hard task, which Commodore has tried to make a little easier by publishing the *Commodore Software Encyclopedia.* The list of business software we're running in this issue, I have to admit, was taken (in extremely condensed form) from entries in the newest edition of the *Software*

Encyclopedia, which should be available at your Commodore dealer soon, if not right now. If you'd like more detailed explanations of the programs we've included in our chart in this issue, you'll find them in the *Software Encyclopedia.*

You'll also notice that we're looking for more articles about using and/or programming our CBM 8032 systems. Several 8032 users have asked us for more input, and we'd like to oblige. If you've got some information that would be of use to our CBM audience, write (or call) and we'll send you our Guidelines for Writers.

We'd also like to run occasional cartoons, so if you create your own, we'll be glad to take a look at them. Send us copies (not originals) for approval and if we like them we'll ask you for the originals so we can reproduce them.

Next issue we'll be featuring new developments from Commodore's Software Division. They're working on some very hot items, especially for the Commodore 64, that will amaze and delight you. See you then. **C**

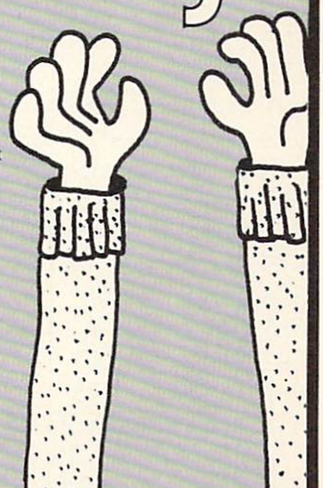
—Diane LeBold
Editor

tired of software that doesn't work?

reach up & turn over to...

new leaf!™

All New Leaf Inc. products are created with our special committment to quality . . . because we really care. Unlike most software, even novice computer users will be able to enjoy all of the benefits, without any of the hassles!



BIORHYTHM+

Classic Biorhythm with audio feedback & compatability comparisons. Plus, ours will give you the option of a printed chart.*

Cassette only **14.50**

C*A*R*S

(Car Analysis Reporting System) Graph & monitor operating efficiency & cost of ownership including car insurance, payments, taxes & license fees as well as repairs. It distinguishes between operating & ownership costs. Again, ours will give you the option of printed reports.*

Cassette version **24.50**
Disk version **27.50**

ESP›Calc

(Electronic Spreadsheet Planning Calculator)
Now you can calculate any spreadsheet easier! Rows and columns are limited only by your machine's memory. An excellent tool for planning and repetitive calculations. You can even make printed copies of your spreadsheets.*

Cassette version **43.50**
Disk version **47.50**

*On any Commodore® compatible printer.

Each of our programs has been tested and re-tested over and over again. Our national testing network eliminates any problems BEFORE they reach you. Simply type Load & New Leaf's™ unique Self-starting format will do the rest.

Mastercard and VISA customers, please add 3%. MasterCard VISA

new leaf inc.

Product ordered ______
☐ VIC-20® ☐ Commodore 64®
☐ Disk ☐ Cassette
Amount enclosed ______
Name ______
Address ______
City ____ State ____ Zip ____
Phone Number ( ) ______
MC or VISA # ______
Expiration date ______
Interbank # (MC) ______
Missouri residents add sales tax
120 Lynnhaven • Belleville, IL 62223

COMMODORE 64K

American Peripherals

GAMES

(on tape)

646 Pacacuda 19.95
650 Logger 19.95
651 Ape Craze 19.95
652 Centropod 19.95
653 Escape 19.95
641 Monopoly 19.95
642 Adventure #1 19.95
648 Galactic Encounter 9.
667 Yahtzee 14.95
671 Robot Blast 14.95
673 Moon Lander 14.95
676 Othello 14.95
686 Horserace-64 14.95
692 Snake 14.95
697 Football 14.95
819 Backgammon 24.95
822 Space Raider 19.95
846 Annihilator 19.95
842 Zwark 19.95
845 Grave Robbers 13.95
841 Pirate Inn Adv. 22.95
904 Shooting Gallery 14.95
816 Dog Fight 19.95
817 Mouse Maze 19.95
818 Ski Run 22.
820 Metro 22.
823 Sub Warfare 29.
838 Retroball 39.95 (cartridge)
839 Gridrunner 39.95 (cartridge)
825 Mine Field 13.
672 Dragster 14.95
662 Oregon Trail 14.95
679 3-D TicTacToe 14.95
655 Castle Advent. 14.95

EDUCATIONAL

(on tape)

644 Type Tutor 19.95
645 Assembly Language Tutor 14.95
687 Fractional Parts 14.95
902 Estimating Fractions 14.95
695 Tutor Math 14.95
870 Square Root Trainer 14.95
699 Counting Shapes 14.95
694 Money Addition 14.95
689 Math Dice 14.95
678 Speed Read 14.95
643 Maps and Capitals 19.95
645 Sprite Editor 19.95
904 Sound Synthesizer Tutor 19.
696 Diagramming Sentences 14.95
690 More/Less 14.95
688 Batting AVERAGES 14.95
802 TicTac Math 16.95
904 Balancing Equations 14.95
905 Missing Letter 14.95
864 Gradebook 15.
810 French 1-4 80.
811 Spanish 1-4 80.
807 English Invaders 16.95
809 Munchword 16.95
812 Puss IN Boot 20.
813 Word Factory 20.
660 Hang-Spell 14.95
905 Division Drill 14.95
906 Multiplic. Drill 14.95
907 Addition Drill 14.95
908 Subtraction Drill 14.95
910 Simon Says 14.95
911 Adding Fractions 14.95
912 Punctuation 14.95

EDUCATIONAL

Series on disk

Computer Science (30 programs) $350
HS Biology (70 programs) $500
HS Chemistry (40 programs) $450
HS Physics (60 programs) $475
HS SAT Drill (60 programs) $99.
Elem. Social Studies (18 pr.) $225
Elem. Science (18 programs) $225
Elem. Library Science (12 pr.) $170
Librarians Package (4 utilities) $110
3rd Grade Reading (20 lessons) $99.
4th Grade Reading (20 lessons) $99.
5th Grade Reading (20 lessons) $99.
6th Grade Reading (20 lessons) $99.
Spanish Teaching (12 lessons) $95.
PARTS OF SPEECH (9 lessons) $95.

BUSINESS

(all on disk)

WORD PRO 3 + 95.00
DATAMAN-64 data base program. 49.95
PERSONAL FILING SYSTEM (index card style) 19.95
HOME FINANCE 19.95
CYBER FARMER $195.
GA 1600 Accounting System 395.
PERSONAL TAX 80.
ACCOUNTS RECEIVABLE 22.
New York State Payroll 89.
MAILING LIST 24.
Manufacturing Inventory 59.
Stock Market Package 39.
Finance 16.95

Visa — Mastercharge — C.O.D.

ORDERING BLANK

To: American Peripherals
122 Bangor Street
Lindenhurst, NY 11757

Ship to: Name ____________________

Street ____________________

Town, State, ZIP ____________________

____

☐ Please send your complete 64K catalog.

ITEM	DESCRIPTION	PRICE

NY State Residents only add 7¼% tax ______

Shipping $1.50

(If COD, add 1.50) ______

TOTAL AMOUNT ______

If Canada or Mexico, add an additional $2.00

Commodore 64™ is a registered TM of Commodore Business Machines Inc.

letters

"Modern" Computer Languages Missing the Point?

To the Editors:

Even though I have taught computer science in various schools for ten years, I still seem to know little that is of use today. My field is applications programming and languages. I have often taught "introduction to programming" courses, showing students how to write programs in a variety of languages, including FORTRAN, BASIC, COBOL, ALGOL, LISP, SNOBOL and APL, and have taught machine language and assembly language, too. Now we have FORTH, C, COMAL, PILOT and LOGO—and so on, virtually *ad infinitum*. It's a bit like fashions in clothing.

As with languages, so with "programming style". There is a school of thought that, no matter how remarkable and powerful the program, if it is not written according to the style set up by Edsger W. Dijkstra of Burroughs, with no unconditional GOTO's, it may as well be burned—a recommendation Dijkstra has seriously advanced with respect to the language PL/I.

The point of this is to ask if we are not, in all of this concern about what is "modern and fashionable", rather missing the point of what our science is all about. Byzantine civilization sank into pedantry because it placed form and uniformity of style before everything else. I think the same thing is happening to computer science. It began as a brilliant labor-saving device, a practical disclosure of what is really going on in mathematical operations, but now is decaying into arguments about unconditional GOTO statements, linguistic subtleties and other matters totally divorced from usage. Along with this is the tendency to continually change operating systems so that no one can gain any confidence in his ability to use the machines for really significant applications.

I think this obsession with machines, with languages, with style, with fine points and with everything detracting from a transparent symbology which is servant to the solution of profound problems is a degeneration of the point of computer science. A generation of in-grown specialists who can interface bed pans and feather dusters with obscure operating systems is not my conception of competent computer scientists. I have had the opportunity of direct contact with the creators of these unfortunate fashions and am not impressed.

Let's get back to scientific, artistic and mathematical applications. The best kind of computer is one that you don't even know is there—one that becomes a part of you—not a temperamental pile of crap accompanied by twelve volumes of jargon.

Yes, I am a PET owner and user and I love your product. Don't be baited into the big computer mode. I use that stuff, too. I know. It is a stack of needless sophistry and mysticism. **C**

Sincerely,
Dr. George Robert Talbott
Chief Computer Scientist
Specialised Software
Wilmot, Wisconsin

Our PETs Have More Bytes Than Barks!

Mandy, the pet of Linda Martin Bilyeu from Watsonville, California, showed up at school one day to pose, flanked by Linda's other favorite classroom PETs.

Commodore's Computer Challenge Sparks High Interest at 1983 Olympics of the Mind

by Mark Odgers
Commodore Customer Support Representative

The Olympics of the Mind was created in 1978 by New Jersey educators Theodore Gourley and Samuel Micklus to foster the development of students' creative and intellectual abilities. Since then it has challenged thousands of students each year to solve problems that force them to think originally and creatively.

In this article Mark Odgers, who created the first computer problem for the Olympics on behalf of Commodore, explains the problem and presents the winning solutions.

The Olympics of the Mind World Finals were held on the campus of Central Michigan University in Mt. Pleasant, Michigan, on May 26 and 27, 1983. This annual event was the culmination of nine months of competition which challenged the creativity of students from kindergarten through grade 12 and for the first time included a computer problem for the youngsters to solve.

The computer problem was sponsored exclusively by Commodore. Commodore's sponsorship included designing the problem and supplying the equipment (twenty VIC 20 systems) at the World Finals so the students could compete. Commodore also sent three representatives to the Finals: Dan Kunz from the education department, Pat McAllister from software and myself. Our group administered and scored the computer event. In addition, we set up the equipment, provided technical assistance and also provided software for scoring the other Olympics of the Mind events. The scoring was done on two CBM 8032 systems using 8050 disk drives and 8023 printers.

The Problem

The problem, titled "Black Box", was designed to challenge the Olympians' creativity on a microcomputer. It called for the teams to create a program that would reproduce on the video screen this or a similar "balancing diamond" pattern:

Olympics of the Mind

```
      O
     O X
    O X O
   X X X O
  O O X X O
 O O O X X O
O O O X O O O
 O X X O O X
  O O X O X
   X X O X
    O O O
     X O
      X
      O
      O
     O X
    X X O
   X O X O
```

The most important component of each team's program was that it not only print the balancing diamond, but that it also be able to handle any random order of x's and o's. The teams saw only samples and received a totally new pattern of x's and o's on the day of the World Finals.

Seems fairly simple, doesn't it? Well, it would be if that were all there was to it. However, like all the other Olympics of the Minds problems, the problem had limitations written in that made it necessary for the teams to be creative in order to both solve the problem and receive a competitive score.

Specifically, the limitations were: 1) The program had to be in BASIC. 2) The only BASIC statements that could be used were PRINT, LET, DIM, INPUT, FOR... NEXT, READ, DATA, GOTO, GOSUB, RETURN and IF... THEN. 3) The only special characters allowed were plus sign, minus sign, asterisk, slash, equal sign, opening and closing parentheses, dollar sign, quotation marks, greater- and less-than symbols and commas. Colons were not allowed. 4) The only variable data in the program had to be input using the INPUT statement. The data could not be inserted into the program itself. The variable data was limited to six pairs of two characters each. (Note: Division I, grades K-5, was allowed 60 one-character inputs.) The only allowable data characters were A-Z and 0-9. There were no exceptions. If it was not specified in the limitations, it could not be used.

The limits written into the Black Box problem had three purposes: 1) To make the competition equal (no particular advantage could be gained by developing the program on different computers with different capabilities. 2) To assure a team's program would work on the computer provided by the tournament directors (VIC 20's). 3) Lastly, and most importantly, to test programming creativity and make it necessary to program the computer step-by-step without being able to take advantage of the special shortcut instructions built into the machine.

Scoring

Scoring was based on the following criteria. The lowest score wins.

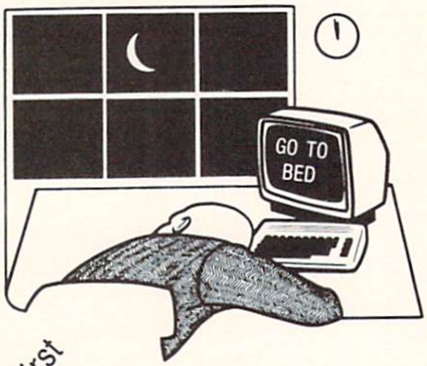
—Time of operating measured from starting signal to handing in result sheet and tape.	1 point per second
—Number of lines in program	10 points per line
—Characters not missing or wrong but out of format.	10 points for each character out of format
—Number of characters wrong.	5 points per wrong character
—A character is missing from pattern.	10 points for each missing character
—Using a BASIC statement which is not allowed (see limitations section).	250 points for each illegal statement
—Using a character in the program which is not allowed (see limitations section).	250 points for each illegal character
—Using a character in your inputs which is not allowed (see limitations section).	250 points for each illegal character

The Winners

Over 200 schools representing the United States and Canada participated in the overall World Finals competition. They had arrived at the World Finals by winning state and regional competitions. Of the 200 schools entered, 86 participated in the Commodore computer problem. As in all Olympics of the Mind events, the Olympians were classified into three divisions: Division I represented kindergarten through grade 5, Division II grades 6 through 9 and Division III grades 10 through 12.

The MIDNITE SOFTWARE GAZETTE / The PAPER

A Bi-Monthly Journal of Notes,
Reviews and Articles
Five Years of Service to the PET® Community

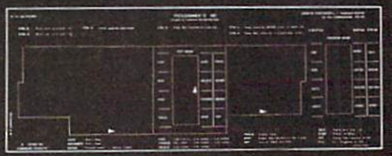


The First Independent U.S. Magazine for Users of Commodore Brand Computers

EDITORS: Jim and Ellen Strasma
$20 US / YEAR
Sample Issue free on request, from:
635 MAPLE, MT. ZION, IL 62549 USA
217/864-5320

NEW

LEROY'S CHEATSHEET™ KEYBOARD OVERLAYS FOR YOUR VIC-20™



AT LAST! The information you need, at your fingertips, without going back to the manual. No need to memorize commands. These durable plastic coated overlays are designed for use by first time users, amateur enthusiasts, and trained professionals. Each overlay contains program starting locations, function key labeling, commands and warnings in bold letters, and additional helpful aids in center cutout.

DON'T DELAY - MAIL TODAY

Please send me the following Leroy's Cheatsheet™ keyboard overlays

- ☐ Programmer's Aid[1]
- ☐ Vicmon[1]
- ☐ Super Expander[1]
- ☐ Vic Typewriter[1]
- ☐ Victerm 1[1]
- ☐ Graphic Printer (1515 & 1525)[1]
- ☐ UMI Wordcraft 20[2]
- ☐ HES Writer[3]
- ☐ HES Vic Forth[3]

ONLY $3.95 EACH

Send check or money order plus $2.00 (postage and handling)
PA residents add 6% sales tax.

Name ______
Address ______
City ______ State ______ Zip ______

(1) Products of Commodore Business Machines, Inc. (2) Product of United Microware Industries, Inc (3) Products of Human Engineered Software. VIC-20 is a Trademark of Commodore Business Machines, Inc.

CHEATSHEET PRODUCTS™ CM883
P.O. Box 8299 Pittsburgh PA. 15218

Congratulations to the fifteen winning entrants:

Division I: 41 teams participating

1st	Harry Spence School	Wisconsin	259 points
2nd	Canaan School	New Hampshire	
2nd (tie)	Cherry Hill School	New Jersey	
3rd	Weston School	Connecticut	
4th	Hoover School	Oregon	

Division II: 29 teams participating

1st	Alice Birney School	South Carolina	402 points
2nd	Canaan School	New Hampshire	
3rd	Jefferson School	Michigan	
4th	Shepard School	Washington, D.C.	
5th	Hodgdon	Maine	

Division III: 28 teams participating

1st	Revere High School	Ohio	380 points
2nd	Alexander Graham Bell H.S.	North Carolina	
3rd	Fairview High School	Colorado	
4th	Clover Hill High School	Virginia	
5th	Wayne Central High School	New York	

The winning solutions for Divisions I and III follow. C

Olympics of the Mind Winning Solutions

Compare your solutions to those of our Division I and Division III first-place programs.

Division I

```
1 DIM B$(18)
2 FOR J=1TO18
3 READ B$(J),X
4 FOR J1=1TOX
5 INPUT A$
6 B$(J)=B$(J)+A$+" "
7 NEXT J1
8 NEXT J
```

```
9 FOR J=1 TO 18
10 PRINT B$(J)
11 NEXT J
12 DATA"        ",1,"        ",2,"        ",3,"     ",4,"   ",5," ",6,"",7," ",
   6,"  ",5,"",4
13 DATA"    ",3,"      ",2,"        ",1,"         ",1,"         ",1,"
   ",2,"    ",3," ",4
```

Division III

```
10 DIM I(49),X(60),X$(60)
15 FOR K=1 TO 49
20 I(K)=K
30 NEXTK
40 FOR K=1 TO 6
50 INPUT A
110 FOR B=1 TO 7
115 Z=Z+1
120 IFA/2<>I(A/2)THENX(Z)=1
130 A=I(A/2)
140 NEXTB
145 Z=Z+3
150 NEXTK
160 FORK=1TO60
170 IFX(K)=1THEN185
180 X$(K)="O "
182 GOTO190
185 X$(K)="X "
190 NEXTK
200 PRINT"      "X$(1)"                        "X$(2)X$(3)
220 PRINT"     "X$(4)X$(5)X$(6)"                     "X$(7)X$(8)X$(9)X$(10)
240 PRINT"   "X$(11)X$(12)X$(13)X$(14)X$(15)
250 PRINT"  "X$(16)X$(17)X$(18)X$(19)X$(20)X$(21)
260 PRINT" "X$(22)X$(23)X$(24)X$(25)X$(26)X$(27)X$(28)
270 PRINT"  "X$(29)X$(30)X$(31)X$(32)X$(33)X$(34)
280 PRINT"   "X$(35)X$(36)X$(37)X$(38)X$(39)"     "X$(40)X$(41)X$(42)
X$(43)
300 PRINT"     "X$(44)X$(45)X$(46)"                   "X$(47)X$(48)
320 PRINT"       "X$(49)"                   "X$(50)"                "X$(
51)
350 PRINT"      "X$(52)X$(53)"                    "X$(54)X$(55)X$(56)
360 PRINT"    "X$(57)X$(58)X$(59)X$(60)
```

Advanced Bit-Mapped Graphics on the Commodore 64
Part 2

by Frank Covitz

Frank concludes his two-part series by showing you how to create a complete graphics language, using an assembler, that you can then use to program bit-mapped graphics on your Commodore 64. Part 1 appeared in* Commodore, *Issue 24.

In the last installment we discussed the essential features of bit-mapped graphics on the Commodore 64. We went over three steps in creating bit-mapped graphics using machine language. Now we get to step four—drawing the "best" straight line between two points. As I said last issue, the technique I'm going to use is not the easiest way, but it is one of the fastest, so it will be worth your time to try to understand it.

Since we are going to use a bit of algebra, it may be time for you to break out your first-year math book. First, consider a Cartesian coordinate system, with an X-axis and a Y-axis (Figure 3). Imagine a straight line going in any direction, but starting at the origin and ending somewhere. Our algorithm needs to start a sort of graphic "cursor" at the origin and figure out, in single pixel movements, how to "walk" it as closely as possible to that line, turning on pixels as we go.

Note that, since the procedure will always take steps from where the "cursor" is to where it is going, we really could have started our line anywhere. To be more specific, our "cursor" is simply the byte address and bit data that goes with it.

The routine we developed in Part 1, PXADDR, will give us this "cursor", if we start it off at the X,Y coordinate of the starting point for the line we want to draw. For bit-mapped graphics, PXADDR, is equivalent to a graphic MOVE command, since it gets us to X,Y without drawing. For the moment, we will leave aside the precise procedure for moving in pixel-sized steps, and just consider what types of moves to make to keep as close to the true line as possible.

Now comes the kicker. I claim that for any given line, only *two* types of elementary moves are needed to do that "walk". To see this more clearly, divide up the coordinate system into eight octants by drawing two 45 degree diagonal lines through the origin, and at right angles to each other (Figure

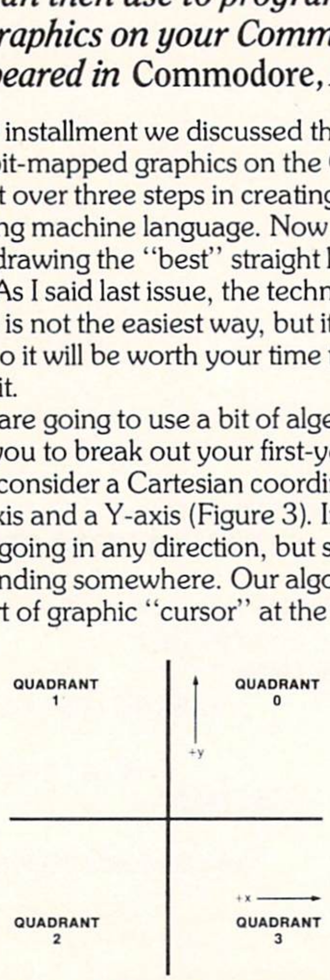


Figure 3. X,Y COORDINATE SYSTEM

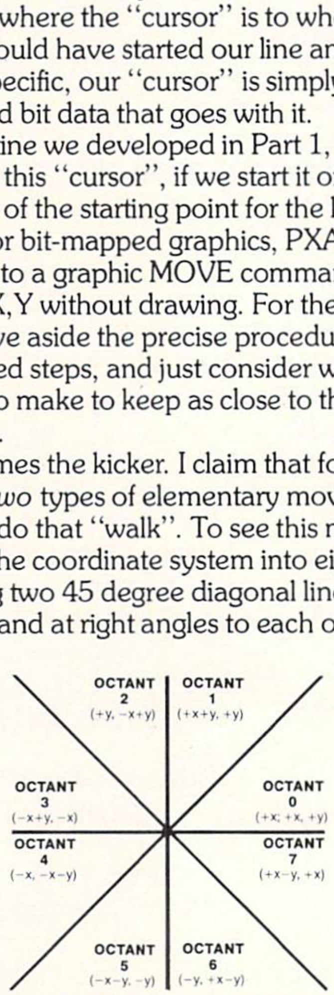


Figure 4. OCTANTS WITH ALLOWABLE MOVES

4). Just like the points of a compass, right? Any line has to fall entirely inside, or on, one of these octants. Let's number them 0-7 (just like the bits in a byte... hmm!), with octants 0 and 1 in the first quadrant, octants 2 and 3 in the second quadrant, etc.

If the line happens to fall in octant 0, our steps will consist of either 1 pixel movements to the right (+X direction) or a combination of 1 pixel right and one pixel up (+X and +Y). In octant 1, the moves will be either right and up (+X+Y) or just up (+Y).

Aha! If we just could figure out which octant our line is in, we would at least have restricted the possible elementary moves to just two types. It's not too difficult if you think about it. First of all, instead of considering the two endpoints—call them X1, Y1 and X2, Y2—separately, what we need are their differences. (Remember, we've already taken care of getting to the starting point by calling PXADDR using X1,Y1). So, we first do dX = X2−X1 and dY = Y2−Y1. Next, take the absolute value of dX and dY. If ABS(dX) is greater than ABS(dY) the line must be in octants 0,3,4, or 7, right?

We've now got two groups. If we're in the first group, is dX positive? If it is, the line must be in octant 0 or 7. Next, is dY positive? If it is, then we must be in octant 0. Three yes/no decisions are all we need. Here is a table of the three conditions needed to fix which octant the line must be in:

Octant	is ABS (dX)> ABS(dY)?	dX	dY	move type
0	+	+	+	+X,+X+Y
1	−	+	+	+X+Y,+Y
2	−	−	+	+Y,+Y−X
3	+	−	+	+Y−X,−X
4	+	−	−	−X,−X−Y
5	−	−	−	−X−Y,−Y
6	−	+	−	−Y,−Y+X
7	+	+	−	−Y+X,+X

Now for the algebra part. The equation for a straight line through the origin is just Y = mX, where m is the slope, OK? The slope, in turn, is just the ratio of the differences in the Y and X coordinate endpoints, i.e., m = dY/dX, and we can substitute this into the straight line equation to get Y = (dY/dX)*X. Multiply both sides by dX to get dX*Y = dY*X. Next subtract dX*Y from both sides to get 0 = dY*X − dX*Y. Any specific point X1,Y1 must satisfy this equation if it is on the line.

Now, suppose we see how far off we are if, starting from a point on the line, we move one unit in the +X direction. Since this new point is no longer exactly on the line (except if it is horizontal), the new term on the right-hand side will no longer exactly equal zero. Let the "error" be represented by the letter "e". So we have for this new point, e = dY*(X+1) − dX*Y.

Next, expand this to form e = dY*X + dY − dX*Y, and note that the right-hand side contains the term dY*X − dX*Y, which, by our previous equation, was exactly equal to zero. So, we are left with simply e = dY.

By exactly the same reasoning, if we made a unit step in the +Y direction from a point on the line, we would have an error e = −dX. This is nice because, as you can see, a step in the +X direction contributes a positive error and a step in the +Y direction

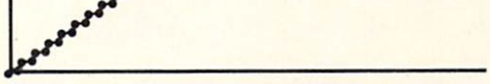


Figure 5. EXAMPLE OF THE STRAIGHT-LINE ALGORITHM. LINE GOES FROM (X,Y) TO (X+40, Y+20)

contributes a negative error. In other words, errors caused by stepping in the +X direction can be reduced by stepping in the +Y direction. We are done just after the total number of steps equals dX (since in octant 0 every move will have a +X move in it). This will be obvious if you think about it. We are now in a position to state the straight line algorithm (at least for a line in octant 0; see Figure 5):

Step 0. Start with X = 0, Y = 0, C = 0

Step 1. Move one step in the +X direction, and let e = e+dY

Step 2. If e is negative go to Step 3, else set e = e − dX and take one step in the +Y direction.

Step 3. Turn on the pixel.

Step 4. Let C = C+1

Step 5. If C < or = dX go to Step 1; else we're done.

By repeatedly checking the sign of e and taking the appropriate steps, we've managed to stay as close as possible to the desired line, without skipping any pixels, and by using just addition and subtraction. Except we left out one important step—we didn't say what the initial value of e should be. At first thought you might think it obvious that e should start at zero, since we started at the exact X,Y coordinate of the line's starting point. However, on reflection, we can see that this isn't quite right by considering the following case.

Imagine that we had a line in octant 0 that was nearly horizontal; in other words dY is very small compared to dX. If e started off at zero, it would always be positive after the first pass in Step 1, so that the first "move" would always be a +X+Y type. If dY were to equal +1 for example (a very nearly horizontal line), only one +X+Y step would be needed in drawing the entire line, and we certainly shouldn't take this +X+Y step right away. Rather, as I think you can see, the single +X+Y step should be taken at the middle of the line. This situation is correctly taken care of by starting with e = −dX/2, i.e., half the negative of dX.

Although it may not be clear exactly how to do it, I think you can see that it should be possible to set up the same type of algorithm for the other octants. It is just a matter of figuring out the sign of the correction terms, and which type of move is involved in Step 1, and I won't go through the logic of it—I'm sure you've had enough math so far. The algorithm itself, by the way, is not specific to the Commodore 64, and could be used for example in driving a digital plotter, or even in directing a robot to go from "here" to "there".

One more consideration must be taken care of before we can set down a program to implement the above straight line algorithm. What the straight line algorithm does is to decide on a step-by-step basis in what direction to make the next move. As we have seen, these moves can be any one of eight (the eight compass directions) and, for a given line, are one out of two possibilities.

We will now take care of how to implement those elementary moves. First of all, you should realize that of the eight really only four are necessary—namely up, down, left, and right—since the diagonal moves are made up of two of the latter. For example, to move northwest, we would move up then left. (Remember, the pixel won't actually be turned on until after the move is made).

Keep in mind how memory is organized in the bit-map mode (remember the cursor placements we did earlier). The right/left step is perhaps the easiest to figure out. BYTE is already set to the correct bit-map address and BIT contains the power of two representing the current pixel position at byte. To move right, we need to go to the next lower power of two, and to go left, we need to go to the next higher power of two, right? So, to go right, we would do BIT = BIT/2 and to go left, we would do BIT = 2*BIT, OK?

Now comes the fun part. What happens if BIT were equal to one and we wanted to go right? The pixel would sort of fall off the right edge and be lost forever unless we do something about it. This condition can be recognized by checking whether INT(BIT) = 0. The cure is simple; just add eight to the BYTE address, since we want to go to the same line of the next character cell, and set BIT = 128, i.e., the leftmost pixel in the new location. Conversely, if on trying to go left, BIT ends up > 128, we need to subtract eight from BYTE and set BIT = 1. The right/left subroutines are then, simply:

```
2000 REM MOVE 1 PIXEL RIGHT
2010 BIT = INT(BIT/2):IF BIT = 0 THEN BYTE = BYTE+8:BIT=128
2020 RETURN
2100 REM MOVE 1 PIXEL LEFT
2110 BIT = 2*BIT:IF BIT > 128 THEN BYTE = BYTE-8:BIT=1
2120 RETURN
```

It is just as simple or simpler in machine language: (Note that here I am using symbolic notation, in which BIT = $033F, BYTE = $FD and BYTE + 1 = $FE)

```
RIGHT LSR BIT     ;shift 1 bit to the right
      BCC RDONE   ;if carry clear, we're done
      ROR BIT     ;this sets BIT = $80 and clears the carry flag
      LDA BYTE    ;add 8 to BYTE
      ADC #8
      STA BYTE    ;take care of low part
      BCC RDONE   ;done if carry clear
      INC BYTE+1  ;else add 1 to high byte
RDONE RTS         ;we're done
LEFT  ASL BIT     ;shift 1 bit to the left
      BCC LDONE   ;if carry clear, we're done
```

```
       ROL BIT     ;this sets BIT = 1 and clears the carry flag
       LDA BYTE    ;get set to subtract
       SBC #7      ;this is like subtracting 8, since carry is clear
       STA BYTE    ;take care of low byte
       BCS LDONE   ;done if carry set
       DEC BYTE+1  ;else take care of high part
LDONE  RTS         ;we're done
```

The up/down routines are just a bit trickier. If we stay within a character cell, going up is equivalent to subtracting one from BYTE, and going down to adding one to BYTE. (Note that the value of BIT can not change as a result of an up or down move.) What would happen if we were already at the bottom line of a character cell and we moved down, or if we were at the top and moved up? The down move would take us to the top of the character cell just to the right of the current one, and the up move would take us to the bottom of the character cell just to the left of the current one—obviously not where we want to be.

The fix is simple: just add 313 to BYTE in the former case and subtract 313 from BYTE in the latter case. Where in the world did the 313 come from? Remember adding 320 to BYTE would move us an entire character cell down, which would be seven lines too far, so we just subtract seven from 320 to get 313, which gets us to the top line of the next lower character cell. The same kind of reasoning applies to moving up one line. So, here are the BASIC subroutines for moving up/down.

```
2200 REM MOVE UP ONE LINE
2210 IF BYTE AND 7 = 0 THEN BYTE = BYTE - 313:RETURN
2220 BYTE = BYTE - 1:RETURN
2300 REM MOVE DOWN ONE LINE
2310 IF BYTE AND 7 = 7 THEN BYTE = BYTE + 313:RETURN
2320 BYTE = BYTE + 1:RETURN
```

(The AND 7 in each case checks for our exception condition. If the result equals zero, it means we're on the top line of a character cell. If the result equals seven, it means we're on the bottom line.)

In machine language, these routines are:

```
UP     LDA BYTE  ;check for exception
       AND #$07  ;test the low bits
       BNE UP1   ;if not = 0 we're just going to subtract 1
       SEC       ;else we're going to subtract 313
       LDA BYTE
       SBC #$39  ;313 = $0139
       STA BYTE  ;take care of low byte
       LDA BYTE  ;take care of high byte
       SBC #$01
       STA BYTE
       JMP UPDONE ;we're done
UP1    SEC       ;subtract 1
       LDA BYTE
       SBC #$01
       STA BYTE
       LDA BYTE+1;take care of high byte
       SBC #$00
       STA BYTE+1
UPDONE RTS        ;we're done

DOWN   LDA BYTE  ;check for exception
       AND #$07  ;examine low bits
       CMP #$07  ;is result = 7?
```

```
       BNE DOWN1 ;no, we're just going to add 1
       CLC       ;else we're going to add 313
       LDA BYTE
       ADC #$39  ;since 313 = $0139
       STA BYTE
       LDA BYTE+1;take care of high byte
       ADC #$01
       STA BYTE+1
       JMP DDONE ;we're done
DOWN1  INC BYTE  ;add 1 to low byte
       BNE DDONE ;if result not = 0 then we're done
       INC BYTE+1;else adjust high byte
DDONE  RTS       ;we're done
```

Now we just have to take care of the four diagonal moves and we are done with this stage. Trivial, right, since the diagonal moves are just combinations of the appropriate pair of the right/left/up/down moves? So:

```
2400 REM MOVE 1 STEP TO UPPER RIGHT
2410 GOSUB 2200:GOSUB 2000:RETURN

2500 REM MOVE 1 STEP TO UPPER LEFT
2510 GOSUB 2200:GOSUB 2100:RETURN

2600 REM MOVE ONE STEP TO LOWER RIGHT
```

```
2610 GOSUB 2300:GOSUB 2000:RETURN


2700 REM MOVE ONE STEP TO LOWER LEFT
2710 GOSUB 2300:GOSUB 2100:RETURN
```

That wasn't too bad, in fact it was so simple that I'm not going to give the corresponding machine language routines here. (The entire assembly language source for the whole shootin' match is given later.) We are finally in a position to set down in BASIC the algorithm for step four of our outline.

```
100 REM THIS ROUTINE DRAWS A STRAIGHT LINE FROM THE CURRENT (X1,Y1)
110 REM GRAPHICS POSITION TO THE NEW ONE (X2,Y2)
120 REM I INDICATES THE OCTANT
130 REM C COUNTS THE MOVES
140 REM E IS THE ERROR ACCUMULATOR
150 IF X1<0 OR X1> 319 OR Y1<0 OR Y1>199THEN ?"ERROR":STOP
150 GOSUB 1000 : REM SET BYTE AND BIT FOR X1 Y1
160 REM ENTER HERE IF BYTE, BIT ALREADY SET
170 IF X2<0 OR X2>319 OR Y2<0 OR Y2>199 THEN ?"ERROR":STOP
180 DX=X2-X1:DY=Y2-Y1
190 X1=X2:Y1=Y2:REM X1, Y1 SET FOR NEXT TIME AROUND
200 I=0:C=0:IF DX<0 THEN DX=-DX:I=2
210 IF DY<0 THEN DY=-DY:I=I+4
220 IF DX-DY<0 THEN T=DX:DX=DY:DY=T:I=I+8:REM INTERCHANGE DX AND DY
230 E=-DX/2:REM NOW SET TO MOVE
```

```
240 GOTO 330:REM JUMP INTO MIDDLE OF DRAWING LOOP
250 REM MAIN DRAWING LOOP STARTS
260 N=I:E=E+DY
270 IF E<0 THEN 300
280 E=E-DX:N=N+1
290 REM MAKE MOVE BASED ON N
300 IF N<8 THEN ON N+1 GOSUB 1000,1100,1200,1300,1400,1500,1600,1700
310 IF N>7 THEN ON N-7 GOSUB 1800,1900,2000,2100,2200,2300,2400,2500
320 REM SET PIXEL ON
330 POKE BYTE,(PEEK(BYTE) OR BIT)
340 C=C+1
350 IF C<DX THEN 260:REM KEEP LOOPING
360 RETURN
```

The first part located our octant, and at the same time adjusted dX and dY to be what was needed (both positive and dX > dY) for the stepping algorithm. N then alternates between I and I + 1 as directed by the sign of E. The two massive ON N GOSUB NNNN's make the correct pair of moves for the specific octant.

Now comes the machine language version. To make things clearer, I will use the same variable names as in the BASIC version (remember these will refer to specific RAM addresses which are defined later in the assembly source), and the comments will also refer to the BASIC version.

```
;this routine assumes the X's and Y's are in range
;
;NOTE - DX,DY, and E are double byte signed numbers

LINE   SEC        ;DX=X2-X1
```

```
        LDA X2      ;take care of low byte
        SBC X1
        STA DX
        LDA X2+1    ;then take care of high byte
        SBC X1+1
        STA DX+1
;
        SEC         ;DY=Y2-Y1
        LDA Y2      ;take care of low byte
        SBC Y1
        STA DY
        LDA Y2+1    ;(these will normally be zero)
        SBC Y1+1    ;(but we need to make DY double-byte)
        STA DY+1
;
        LDA X2      ;X1=X2
        STA X1
        LDA X2+1
        STA X1+1
;
        LDA Y2      ;Y1=Y2
        STA Y1
        LDA Y2+1
```

```
        STA Y1+1
;
        LDA #$00  ;I=0
        STA I
        STA C     ;C=0
        STA C+1
;
        BIT DX+1  ;test sign of DX
        BPL LINE1 ;skip to next if DX>0
        LDA DX    ;IF DX<0 THEN DX=-DX
        JSR COMPL ;subroutine to negate
        STA DX
        LDA DX+1
        JSR COMPH ;negate the high byte
        STA DX+1  ;we now have DX=ABS(DX)
        LDA #$02  ;I=2
        STA I
;
LINE1   BIT DY+1  ;test sign of DY
        BPL LINE2 ;skip to next if DY>0
        LDA DY    ;IF DY<0 THEN DY=-DY
        JSR COMPL ;negate the low byte
```

```
        STA DY
        LDA DY+1
        JSR COMPH ;negate the high byte
        STA DY+1  ;we now have DY=ABS(DY)
        CLC       ;I=I+4
        LDA I
        ADC #$04
        STA I
;
LINE2   LDX DX    ;we're going to check the sign of DX-DY
        CPX DY    ;(at the same time we put DX into X-register)
        LDA DX+1  ;fetch DX+1
        TAY       ;hang on to DX+1 in Y register
        SBC DY+1  ;this is the way to do a double byte comparison
        BPL LINE3 ;skip to next if DX-DY is positive
        LDA DY    ;IF DX-DY<0 THEN T=DX:DX=DY
        STA DX
        LDA DY+1
        STA DX+1
        STX DY    ;(this is why we saved DX in X-register)
        STY DY+1  ;(and DX+1 in Y-register)
        CLC       ;I=I+8
        LDA I
```

```
        ADC #$08
        STA I
;
LINE3   LDA DX     ;E=-DX/2
        JSR COMPL  ;negate low byte of DX
        STA E
        LDA DX+1
        JSR COMPH  ;negate the high byte of DX
        STA E+1    ;we now have E=-DX
        SEC        ;(we're going to divide a negative number by 2)
        ROR E+1    ;rotate right is equivalent to dividing by 2
        ROR E      ;do low byte
;
        LDY #$00   ;we need Y=0 for the next step
        BEQ LINE6  ;JUMP INTO MIDDLE OF DRAWING LOOP
;
;the main drawing loop starts here
;
LINE4   LDX I      ;N=I (set octant pointer into X-register)
        CLC        ;E=E+DY
        LDA E
        ADC DY
```

```
        STA E
        LDA E+1
        ADC DY+1
        STA E+1
        BMI LINE5 ;IF E<0 THEN 300
        SEC       ;else E=E-DX
        LDA E
        SBC DX
        STA E
        LDA E+1
        SBC DX+1
        STA E+1
        INX       ;N=N+1
LINE5   JSR OUTPL ;this makes the correct move based on X-register
LINE6   LDA (BYTE),Y ;POKE BYTE,(PEEK(BYTE) OR BIT)
        ORA BIT
        STA (BYTE),Y
        INC C     ;C=C+1
        BNE LINE7 ;skip over next line unless result is zero
        INC C+1   ;(take care of high byte if necessary)
LINE7   LDA DX    ;IF C<DX THEN 360:REM KEEP LOOPING
        CMP C
        LDA DX+1
```

```
        SBC C+1    ;this is our double byte comparison again
        BCS LINE4 ;keep looping if this is true
        RTS        ;if we got here we're done
;
;finally comes the table of addresses
;note that because of the way JSR works
;we need address minus 1
;
;also note that an address needs two bytes, and that's why we had
to double the index
;
MOVTAB .WORD RIGHT-1 ;moves for octant 0
       .WORD UR-1
;
       .WORD LEFT-1  ;octant 3
       .WORD UL-1
;
       .WORD RIGHT-1 ;octant 7
       .WORD LR-1
;
       .WORD LEFT-1  ;octant 4
       .WORD LL-1
```

```
;
        .WORD UP-1     ;octant 1
        .WORD UR-1
;
        .WORD UP-1     ;octant 2
        .WORD UL-1
;
        .WORD DOWN-1   ;octant 6
        .WORD LR-1
;
        .WORD DOWN-1   ;octant 5
        .WORD LL-1
;
```

Believe it or not, we've just finished step four of our outline, and the end is in sight. The next step is only applicable to the machine language part, and involves a technique for linking BASIC to our machine language routines. The simplest way would be to POKE the appropriate numbers into RAM, and then SYS to the entry point of the machine language routine. But this is clumsy (I'm sure you're fed up with POKEs) and we're not going to do it.

Another way might be to use the USR command to pass a parameter, but our routines need two parameters (X and Y) so we won't do it that way either.

The most elegant way would be through the "wedge" (a routine called by BASIC to pick up consecutive characters from a BASIC program), and we could therefore create our own "reserved" words (like MOVE or DRAW) to call our routines. However, we won't do that for two reasons: 1) the wedge may already be in use (DOS and other program aids use it) and we could clobber it unknowingly. 2) A lot of checking via the wedge tends to slow down all of BASIC, which would defeat one of our main purposes.

So how are we going to do it already? I'll tell you—read on.

The compromise I've chosen is to use the SYS command, and we will use parts of the BASIC interpreter to fetch parameters which we will append to the SYS command. For example, suppose we've set a variable MV equal to the start of our MOVE routine. Our connection to machine language will be SYS(MV),X,Y—where the X and Y are anything normal BASIC can evaluate. That is, (we can leave out the parentheses around MV) it could be SYSMV,5,100 or SYSMV,SQR (5*Y),Z*(X+Y) or SYSMV,−Y,X as long as we keep in mind that the first number, whatever it evaluates to, will be interpreted as the X coordinate and the second as the Y coordinate. This is the way the real guys do subroutine calls in, for example, FORTRAN.

Note: the commas are necessary to keep the parameters separate and we will want a SYNTAX ERROR in line NNNN if they're not present. BASIC "sees" the SYSMV and goes there. Now our routine takes over by first calling the appropriate routine from the BASIC interpreter to check for the first comma (and takes care of SYNTAX ERROR if it's not there), then calls an expression evaluator sequence to evaluate the first parameter (which also aborts on finding an error condition), puts the result into RAM (the subroutine itself knows where to put the result), checks for the next comma, and finally gets the next parameter and executes the MV. The routines needed for the Commodore 64 are:

```
CHKCOM  =  $AEFD ;aborts  with  SYNTAX  ERROR  if  comma  not  next
non-space character
EVAEXP = $AD9E ;EVAluates EXPression in floating point form
FLTFIX = $B1AA ;converts the floating point result to fixed point
in the Y- and A- registers
ERRVEC = $0300 ;points to BASIC's error routine
```

That's all there is to it!! So let's create a little routine, which we can call whenever we need it:

```
and A (high byte). Parameter must be in the form ',<expression>'

GETVAL JSR CHKCOM  ;check for comma (aborts with SYNTAX ERROR if
comma absent)
       JSR EVAEXP  ;evaluates expression
       JMP FLTFIX  ;converts result of EVAEXP to fixed point in
Y,A and returns
;
;here is an example, which implements SYSMV,X,Y
;
MOVE   JSR GETVAL ;fetch X coordinate
       STY X2     ;save low byte
       STA X2+1   ;save high byte
```

```
        JSR GETVAL ;fetch Y coordinate
        STY Y2     ;save low byte
        STA Y2+1   ;save high byte
        JSR RNGCHK ;are X and Y values in range?
        JMP PXADDR ;set BYTE, BIT and return to normal BASIC
;
;here is RNGCHK, which makes sure X is in the range 0 to 319
;and Y within 0 to 199
;aborts with ILLEGAL QUANTITY ERROR if either X or Y are not in
range
;
RNGCHK  LDA X2     ;check X coordinate
        CMP #$40   ;320 dec. = $0140
        LDA X2+1   ;this is our double byte comparison again
        SBC #$01
        BCS RNGERR ;error if carry set
;
        LDA Y2     ;next chaeck Y coordinate
        CMP #$C8   ;200 dec. = $00C8
        LDA Y2+1
        SBC #$00
        BCS RNGERR ;error if carry set
```

```
        RTS          ;no error, return to calling routine
;
RNGERR LDX #$0E     ;this is the way to signal ILLEGAL QUANTITY
ERROR
        JMP (ERRVEC) ;abort through BASIC's error vector
;
```

We are now ready for the sixth and last step of our outline, namely, to provide a clean return back to normal BASIC. This is simply the inverse of what we did in step one, where we initialized the VIC chip for bit-mapped graphics. So we need to turn off bit-mapped mode, get back to bank 0, and restore the normal screen address. In BASIC, this is:

```
3000 POKE 53265,PEEK(53265) AND (255-32):REM TURN OFF BIT 5
3010 POKE 56576,PEEK(56576) OR 3:REM RESTORE BANK 0
3020 POKE 53272,PEEK(53272) AND 7 OR 16:REM RESTORE SCREEN ADRESS
3030 RETURN
```

In machine language:

```
RESTOR LDA $D011  ;VIC control register
AND #$DF    ;turn off bit 5
STA $D011   ;we're now in normal character mode
LDA $DD00   ;bank register
ORA #$03    ;turn on bits 0,1
```

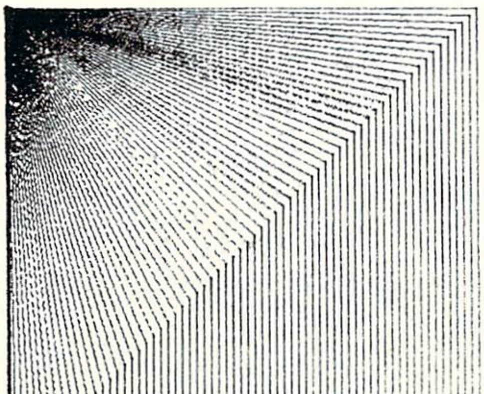
```
STA $DD00   ;VIC now sees addresses from 0 to $3FFF (bank 0)
LDA $D018   ;VIC memory register
AND #$07    ;clear bits 7-3
ORA #$10    ;turn on bit 4
STA $D018   ;screen memory is now at $0400-$07FF
RTS         ;we're done
```

So now that we have all the software resources we need for pixel setting and line drawing in high-resolution, how do we put it all together to give us something usable? Listing #1 gives the complete assembly source for the machine language part, which for the most part, follows exactly the routines I have discussed above. Any differences should be clarified by reading the comments. Those with assemblers will find it quite worthwhile to key in the source text, especially since the potential for expandability is large, and they will be in a very good position for possible future articles.

If you have a machine language monitor, you can key in and SAVE the hex code directly via S "HRSUPP",dn,6000,6331 where dn is 08 for disk, 01 for tape. Use the checksums in listing #2 with your own BASIC program to add up the bytes, and remember to use LOAD "HRSUPP",dn,1 where dn is 8 for disk, 1 for tape. Otherwise, use all of listing #2, which is done in BASIC, which has DATA in "hex", and includes checksums for each 128 bytes. As always for this kind of operation, SAVE your data *before* attempting to RUN.

A BASIC program in listing #3, HRTEST, gives

Continued on page 62


1

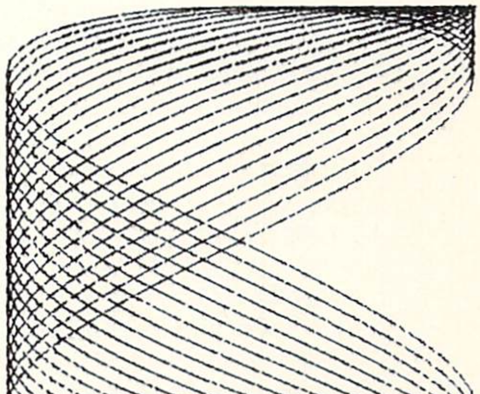

2

Figure 6. Results of the Routines in Listing #3.

Your computer can be your financial advisor, your accountant, your secretary and your file clerk. It will calculate your taxes, connect you right to Dow Jones, and bring you your evening (electronic) newspaper. All you have to do is pick your software carefully and choose a system that can expand as your business does.

HOW CAN COMPUTERS HELP YOUR BUSINESS?

By Diane LeBold

How *can* computers help your business? Those of you who have been using Commodore computers in your businesses already know the answer to that question. Computers save time, paper, file space and aggravation. Mainly they save time. And when you or your employees don't have to spend all that time struggling to keep up records or address envelopes or perform any of the other tedious, time consuming tasks involved in running a business, you can finally get to important things like soliciting new accounts or staying in closer contact with your existing clients or salespeople. Things that help you build up your business and increase your profits—instead of just staying even. Then pretty

The money you invest in a Commodore system can be more than paid back in the time you save and the aggravation you prevent. Which, of course, leaves you with more time and energy to devote to things like marketing, promotion, improving relations with customers and employees.

soon you find ways for your computer to help you do even these new tasks quicker, so you have time for... maybe even a day at the beach. (If you've been doing business by hand you've probably forgotten that people do take days off.)

Commodore computers are being used around the world in all kinds of businesses for all kinds of tasks. In past issues of *Commodore* we've talked about some of these businesses: a nursery (as in plants, not children) that uses a CBM system to enter and track orders, keep inventory and customer records, produce invoices and sales summaries and figure sales commissions; a moving and storage company that uses their CBM to maintain a warehouse control system and produce invoices and statements; a veterinarian who uses a Commodore system to keep records; an announcer on a radio talk show who screens calls using a VIC 20; a tie salesman who keeps all his accounts on a CBM. And in this issue you can find out about other business people who have streamlined their operations using Commodore equipment. This is just a tiny sampling of the many small-to-medium-sized businesses who have used Commodore computers to successfully cope with—and enhance—their growth.

But back to our original question. How can computers help *your* business? Think about this: could you manage your finances better if you could play around on a "what-if" spreadsheet that automatically changed all the affected numbers when one number changed? Without any tedious calculations on your part? What if your gross revenues in one sales area change? How would it affect your overall profit? What if you added five people to your payroll? Would you like to forecast sales and set sales goals? An electronic spreadsheet can help you do all that—and much more—so you can see exactly what your finances will do under various circumstances.

Could you take better care of your customers if you could enter one piece of data—for instance, a product code—and immediately get a list of all the customers who buy that product? Or could you use a list of all the customers who haven't made any purchases since a certain date—instantly and accurately, without having to shuffle through reams of paper files? How about a list of all the sales reps who have sold over $100,000 this quarter? A good data base manager can help you manipulate this kind of important data to your best advantage.

What about those contracts or form letters you have to send out time after time after time, each one just slightly different? Or the reports that undergo several revisions before you get them into final form? Or the labels you need every month to send out your latest updates to your clients? A good word processor can make these tasks so much easier you'll wonder how you ever got by with just a typewriter. (A note for our novices: because several companies make what we call "dedicated" word processors—that is, computers that have word processing software built in and can do only word processing and nothing else—many people think the term "word processor" refers to the hardware—the computer itself. This is not the case. A word processor is software, whether built in or loaded from disk or tape.)

Accounts receivable and payable, with or without the capability to produce invoices or write checks, that updates records immediately so you always know exactly where you stand. Payroll software that calculates deductions and keeps complete records on all employees. Inventory software that you can coordinate with order-entry software to keep your inventory records up-to-the-minute accurate. Specialized programs for contractors that estimate job costs based on the most up-to-date information entered in the system. Other specialized programs for real estate brokers, farmers, lawyers, doctors, designed to meet their unique needs. Retail software that keeps accurate track of what each of your sales people sell each

A computer is one employee who is terrific at boring, tedious, repetitive, time-consuming tasks like complex calculations and information filing and retrieval. So the logical place to start is with those kinds of tasks. (The ones you or your employees generally hate.)

day, calculates commissions, and coordinates with your inventory and payroll software as well.

By now you get the idea, I'm sure. The money you invest in a Commodore system can be more than paid back in the time you save and the aggravation you prevent. Which, of course, leaves you with more time and energy to devote to things like marketing, promotion, improving relations with customers and employees—and, as a result, helps increase market share, productivity and profits.

OK, you're convinced. Now all you need to decide is what kind of system to buy, or how to improve your existing system. When you're ready to make that decision, we suggest you work backwards. First sit down and make a list of all the things you would like your computer system to do—or do better, if you already have a system. Remember that a computer is one employee who is terrific at boring, tedious, repetitive, time-consuming tasks like complex calculations and information filing and retrieval. So the logical place to start is with those kinds of tasks. (The ones you or your employees generally hate.)

Next look at the chart at the end of this article. True, it's by no means the last word on what's available for Commodore systems, but it will give you a good sense of what some of the more popular products presently on the market can do. Under "Capabilities" find the jobs you want your computer to do. Then see which software packages do these jobs. You'll notice that many products—usually designed to be complete general business "systems"—do more than one job, while others are specialized. Very often specialized products made by the same company are compatible with each other. For instance, information in an order entry program may be able to be used in an inventory program produced by the same company. But not every manufacturer provides this cross-compatibility, so before you buy, make sure you check on which programs are compatible with each other. It's an important feature to consider.

Only after you decide which software packages suit your needs are you ready to start thinking seriously about which system to buy. (That's why I said the decision-making process is backwards.) Now you're ready to consider things like the cost and convenience of expanding the system to meet your future needs and the types of peripherals available. For instance, will you need a more expensive letter-quality printer so the copy looks like it was done on a regular typewriter? Or will dot matrix be sufficient? (Dot matrix copy is perfectly readable but looks "computerish"). Do you anticipate needing significantly more memory before too long? Will the number of rows and columns you can view on the screen continue to be sufficient in the future?

You should also think about other types of software and additional features you'd like to have, either just for the fun of it (like the Commodore 64's music synthesizer for instance) or for extended business benefits (like the capability to use a modem, so you can access huge telecommunications data bases to get the latest information on stocks, news, airline schedules and much more—see Walt Kutz's article in this issue for details). Then you can finally weigh cost/benefit ratios, narrow down your possibilities and make a purchase. Actually, if you've done the rest of your homework, this is the easy part.

There can be no doubt that a Commodore computer is a versatile tool. But, like any other tool, its real value and usefulness are often ultimately determined by the skill and good sense of its user. Your computer will not, as some people like to imply, perform miracles—at least not all by itself. But if you put your system together carefully and choose your software intelligently, you will be amazed at how easy formerly cumbersome tasks become.

Business Software for Commodore Computers

System: PET/CBM

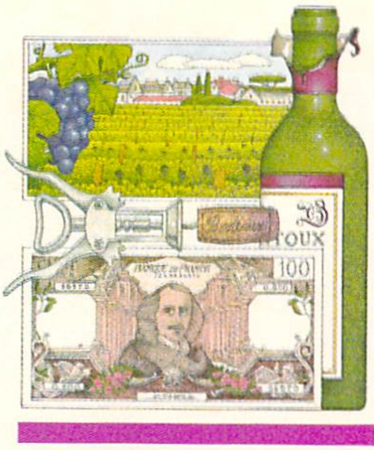
Available From	Program Name	Computer	Drive	Accts	Bus Mgmt	Data Mgmt	Financial	Farm	Inventory	Invoice Billing Order Entry	Legal	Mail List	Medical/Dental	Payroll	Real Estate	Retail	Specialized	Statistics	Tax	Word Processing
Commodore Dealers	OZZ	8032	8050		X															
	Dow Jones Portfolio Management	8032 4032	8050 4040			X	X													
	Freight Rating Information System	Contact Vendor				X														
	Medical Accounting System	8032	8050	X									X							
	Atlas	8032	8050						X											
	Titan	8032	8050							X										
	Information Retrieval & Management Aid	8032 4032	8050 4040			X												X		
	Legal Time Accounting	8032	8050	X							X									
Computer Marketing Services	Accounts Receivable Inventory System	8032 4032	8050 4040	X					X	X										
	Silicon Office	8096	8050 8250 9060			X														X
	Wordcraft 80	8032	8050 4040																	X
Management Accountability Group	MAGIS™	8032	8050	X	X	X				X				X					X	
	MAGIS™ Plus	8032	8050	X	X		X		X	X				X					X	
	Real Estate	8032	8050	X		X			X	X				X	X					
	The Contractor	8032	8050	X	X				X					X						
	Computerized Public Accounting	8032	8050	X			X							X						
Southern Solutions	General Ledger	8032	8050	X										X						
	Accounts Payable	8032	8050	X						X										
	Accounts Receivable	8032	8050	X						X										
	Payroll	8032	8050											X						
	Mailing List Manager	8032	8050									X								
	General Accounting System	8032	8050	X						X				X						
Info Designs	Order Entry System	8032 4032	8050 4040							X										
	Inventory Management System	8032 4032	8050 4040						X											
	Accounts Receivable/Billing	8032 4032	8050 4040	X						X										
	Accounts Payable/Checkwriting	8032 4032	8050 4040	X						X										
	General Ledger System	8032 4032	8050 4040	X																
	Payroll	8032 4032												X					X	

Available From	Program Name	Computer	Drive	Acctg	Bus Mgmt	Data Mgmt	Financial	Farm	Inventory	Invoice Billing Order Entry	Legal	Mail List	Medical/Dental	Payroll	Real Estate	Retail	Specialized	Statistics	Tax	Word Processing
Info Designs (continued)	Management/Client Billing System	8032 4032	8050 4040	X	X															
Professional Software	InfoPro™	8032 SuperPET	2040 4040 8050			X														
	The Administrator	8032	2040 4040 8050			X			X			X								
	Word Pro 1-Plus	8K	Datassette																	X
	Word Pro 2-Plus	16K 32K	Datassette																	X
	Word Pro 3-Plus	4032	4040																	X
	Word Pro 4-Plus	8032 SuperPET	8050																	X
	Word Pro 5-Plus	8096	8050																	X
	Word Pro-ML (Multi-Lingual)	8032	2040 4040 8050																	X
	Word Pro Mail List	8032	2040 4040 8050									X								
Jini Micro Systems	Jinsam	8032 4032	8050 4040			X	X											X		
Personal Software	Visi Calc™	8032 8096	4040 8050		X				X									X		
Canadian Micro-Distributors	The Manager	8032	8050			X														
Computer House Division	A/P-A/R Job Costing & Estim.	Contact Vendor		X						X										
	Accounting	Contact Vendor		X						X				X						
	Inventory	Contact Vendor							X											
	Mailing List	Contact Vendor										X								
	Legal Accounting	Contact Vendor		X						X	X									
	Real Estate Listing	Contact Vendor							X						X					
BPI Systems	General Ledger	4032 8032	8050	X			X											X	X	
	Accounts Receivable System	4032 8032	8050	X			X		X	X		X								
	Inventory Control System	4032 8032	8050			X			X										X	
	Job Cost System	4032 8032	8050		X					X								X		
	Payroll Systems	4032 8032	8050	X										X						
Cyberia, Inc.	Farmer's Workbook	4032 8032	4040 8050		X			X												
	Farrow-Filler	4032 8032	4040 8050		X			X	X											
	Cyber-Farmer	4032 8032	4040 8050	X	X		X	X	X											

Business Software for Commodore Computers

Available From	Program Name	Computer	Drive	Acctg	Bus Mgmt	Data Mgmt	Financial	Farm	Inventory	Invoice Billing Order Entry	Legal	Mail List	Medical/Dental	Payroll	Real Estate	Retail	Specialized	Statistics	Tax	Word Processing
Minisoft	General Ledger	8032	8050	X																
	Accounts Receivable	8032	8050	X						X		X								
	Accounts Payable	8032	8050	X								X								
	Payroll	8032	8050	X										X						
	Inventory	8032	8050	X					X											
	Job Cost	8032	8050	X	X															
	Mail List	8032	8050									X								
Computer System Sales	Chain Inventory	4032 8032	8050						X							X				
	Motorcycle Shop	4032 8032	4040 8050						X							X	X			
	Shoe Store	2001-32 4032 8032	2040 4040 8050						X							X	X			
	Hairdressing School	2001-32 4032 8032	2040 4040 8050		X												X			
	Photo Lab	2001-32 4032	4040 8050		X					X							X			
	Basic Inventory	2000 4000 8000	2040 4040 8050						X											
Clockwork Computers	CCI Restaurant Package	4000 8000	4040	X					X	X						X	X			
	CCI Retail Package	4000 8000	4040	X					X	X						X				
	CCI Retail & Light Mfrg.	4000 8000	4040	X					X	X						X				
INI™	Zipper™	8032	8050									X								
Mystic Software	Stock Brief	16K	2031 4040 8050			X	X													
Bits & Bytes	Billing Manager	4032 8032	4040 8050	X					X	X										
Transadental Software	Transadental File	4032 8032	Datassette	X		X							X					X		
	Dental Recall File	4032 8032	Datassette			X						X	X							
CFI	Tax Preparation System (1040)	Contact Vendor		X															X	
	Asert	8032	4040 8050			X														
Total Information Services	Accounts	Contact Vendor		X		X				X										
	Calendar	Contact Vendor			X															
Instant Software	Accounting Assistant	8K PET					X													
Mini Comp Systems	Inventory Control	4032 8032	2040						X											

Continued on page 57



A TALE OF TWO BUSINESSES

By Diane LeBold

These two very different businesses—one selling Avon Products and the other importing wines and liquors—have one thing in common. They improved their business dramatically when they began using a Commodore computer.

An Avon District Sales Manager "Revolutionizes" Her Business

Illustration—Jean Gardner

For two years in a row Marilyn Phillips' Avon sales district has had outstanding sales increases. Marilyn plans to continue to stay at the top—thanks to the Commodore system her husband brought home.

Marilyn Phillips

When Marilyn Phillips' husband brought home a PET computer in 1979 she thought he was crazy.

"His excuse was that it would revolutionize my business," Marilyn explains. "But neither one of us had ever done anything with computers before."

It wasn't too long, however, before Marilyn, a district sales manager for Avon Products, was using the computer to handle the enormous amount of paperwork involved in running her southern California sales district. As a result, she suddenly had more time to devote to planning her sales strategies and staying in close touch with her 400 sales representatives. This caused a substantial increase in sales volume. In fact, by the end of that year Marilyn's district had one of the highest volume increases in the country, placing in the top 10%—and winning Marilyn a trip to Monte Carlo to boot.

Marilyn points out that before her husband bought the computer, her district already had a high sales volume.

"It's easy to increase a *low* volume," she explains. "But to have a significant increase in an already high-volume area is very hard, especially considering the state of the economy in those years."

Marilyn has since purchased a CBM 8032 computer and an 8050 dual disk drive, but she continues to use the same software packages—a modified version of the *Jinsam* data base manager from Jini Micro Systems, and *VisiCalc*™, an electronic spreadsheet.

On the *Jinsam* data base she keeps a list of all her sales representatives, with their addresses and phone numbers. She has the list coded by length of service, groups (sales leaders, president's club, etc.), territory, census tract boundaries, net sales, number of customers served and number of brochures ordered. As a result she can run a list of representatives in any combination of categories. If she wants to do a specialized mailing, she can, for instance, produce labels for everyone with a two-year length of service who sold more than $500 and who ordered more than 100 brochures—or any other such combination.

Using *VisiCalc*, Marilyn does her sales forecasting for both the district as a whole and individual representatives. She then sets goals for each representative based on past sales records. This system has been very successful in helping increase sales, Marilyn says.

"I once did a forecast for a $50,000 campaign and sent out individualized postcards to the representatives telling each one what their share of the campaign was. We immediately had a wild increase in sales."

But she says she has to be judicious in how she applies her various strategies.

"I could do that kind of thing every time," she goes on, "but I think it would lose its impact. So I try other approaches, too."

Before she started using the Commodore system, Marilyn says she "went crazy" doing all her paperwork by hand. Now, even though she spends as much time at her work as she did before, she's accomplishing much more in the time she spends, getting things done that she simply did not have time for in her pre-computer days.

"Some people just love paperwork," she says, "but I'm not that kind of person. I'd rather get out and work with my representatives individually—and let the computer handle the nitty gritty for me."

"You have to do more than just rely on luck—or a good economy—if you're going to have *consistent* high volume," she elaborates. "I've been in the top 10% of volume increases for two years in a row, and I think the computer really helped me do that."

Eventually Marilyn hopes to be able to hook her computer into Avon's computer, so reports can be transmitted directly. This, she says, is "my dream. It would save a tremendous amount of time."

Not surprisingly the Phillips family computer has affected other areas of their lives. Marilyn says her husband is now head of data processing at his company, since he taught himself about computers using the Commodore system. And her daughter now does all her papers for school on the word processor. ("I put my typewriter away almost two years ago," Marilyn explains. "I think I keep it because I keep thinking I might need it to type an address on an envelope some day.") So, in Marilyn's words, the Commodore system her husband bought to "revolutionize her business" has also managed to revolutionize her family, as well.

Thank a VIC 20 For More Fine Bordeaux Wines

Their big computer system didn't have the flexibility to do the complex calculations this liquor importing business needed. They turned to a VIC 20, and were able to increase their Bordeaux imports about a hundredfold.

Let's get right to the point. In 1981 Michael Allen & Company, Inc., a wine and liquor importer in Lindenhurst, New York, was selling 20 or 30 cases a month of about 40 different classified Bordeaux wines (in addition to other wines and liquors, of course). Now they turn over about 2500 cases a month from a selection of about 350 different fine Bordeaux's. How did it happen? You're right. In 1981 they started using a computer—namely a VIC 20 with 16K expansion—to do the complex calculations needed to handle these particular wines, whose prices are very volatile.

The constant fluctuations in the prices of top quality Bordeaux wines combined with the unpredictability of the French franc, according to Marty Gilbert, executive vice president at Allen & Company, had previously made it next to impossible for the company to handle these wines in any quantity. Unlike their cousins from Burgundy, whose prices remain relatively stable and need to be updated only about once a year, the Bordeaux wines change prices almost as often as a bumble bee changes flowers on a sunny day.

"We couldn't get into the Bordeaux business before we created this program," Marty explains. "The calculations just took too much time. We were trapped."

Marty Gilbert wrote the specialized wine importing program himself, even though he has had no formal training in programming. What Marty's program does, in short, is take the price of the wine in French francs, convert it to dollars (based on the latest value of the franc), add ocean freight, duties, and taxes, and calculate a total New York-landed price—the total cost, in dollars, of getting the wine into the Allen & Company warehouse. It also calculates the in-store price for the retailers to

Marty Gilbert

whom Allen sells, also in dollars.

It then prints out an alphabetical list of all the chateaus within a region, showing the name of the wine, the vintage, the cost in francs, the New York-landed price in dollars and the retail price in dollars, with Allen's mark-up added on from a sliding scale built into the program. After the list is printed out, the retail version or "offering list" then goes out to their customers whenever there is a price change.

Of course, to get everything calculated right you have to enter the latest prices of all the wines that have changed and the current value of the French franc, but that's pretty easy compared to what you'd have to do if you were doing all the calculations by hand.

The program also has another interesting facet. At the beginning it asks for the actual value of the French franc at the time the company placed its order, and then for the actual value at the time they paid the winery. That's because, Marty explains, the company bills its customers and figures retail prices based on the value of the franc at the time the wine is ordered. But they calculate their New York-landed cost at the value of the franc when they actually make payment.

"It's confusing," Marty chuckles "but then it's a confusing issue. Without the computer it would be impossible."

The program, Marty says, is now being used by two other importers—one in California and one in New Hampshire—with great success. In Marty's own company, as a result of using the program, expensive Bordeaux wines now make up 20% of total business —up from 3% in 1981.

Marty has also written two other business programs for the VIC: one that produces a yearly gross profits report and one that calculates his company's state and city excise tax every month.

"It's a very complicated formula," he says of the monthly excise tax calculations. "Before, it took us twenty minutes for each of 500 items. It would take us four days to get it done by hand. Now it takes the VIC about an hour."

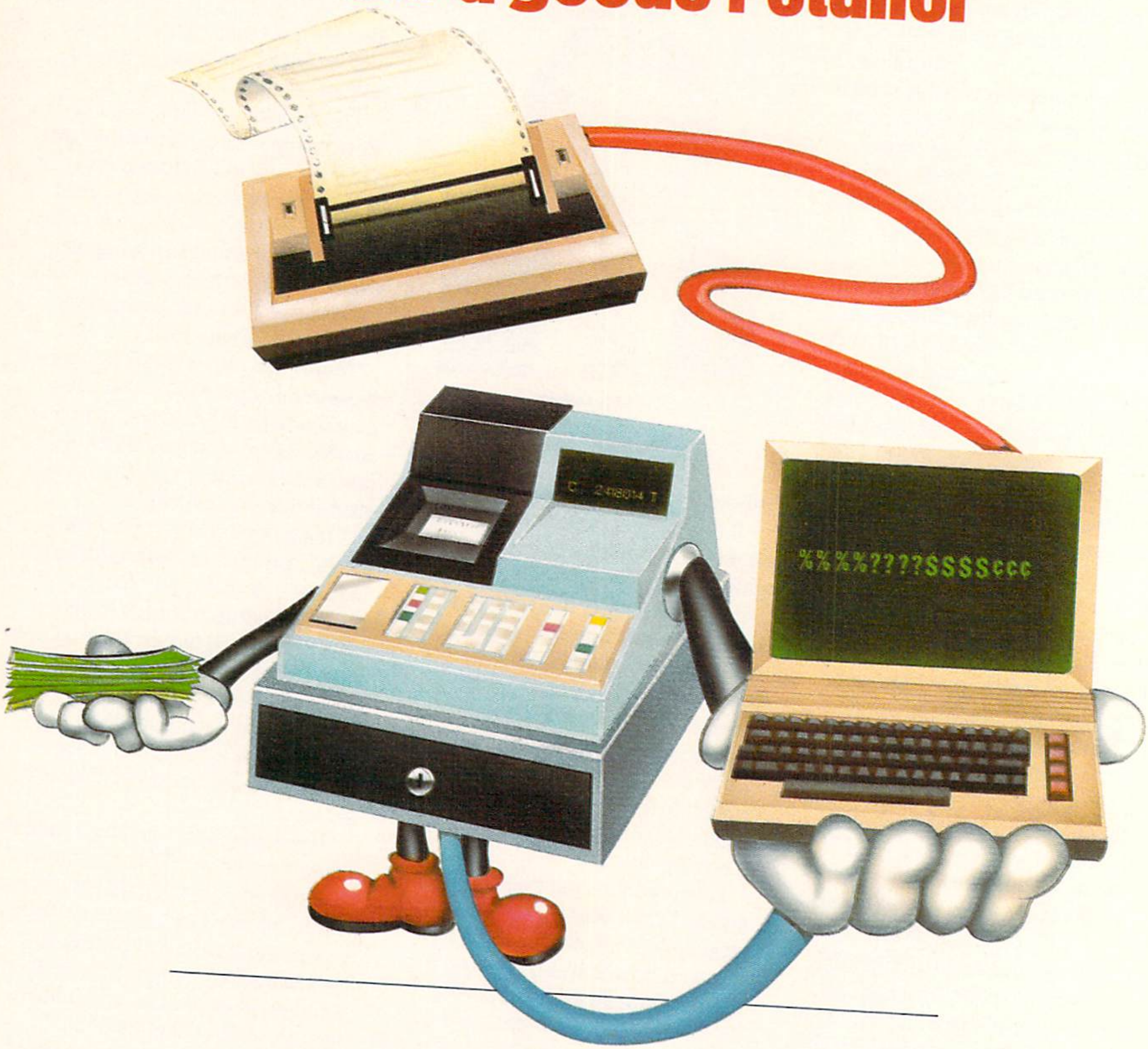
He is also in the process, he says, of writing a data manager for the Commodore 64, his newest love.

"I originally wrote it as a mailing list to handle our 750 customers," he goes on, "but then I saw how it could be used as a data manager. I'm going to use it at home, too, for things like keeping track of what movies are on which tapes for our video cassette deck."

From exquisite wines to video cassettes. Did I hear someone say computers are versatile? Come to think of it, I think Marty Gilbert said that, somewhere in our interview.

Continued on page 60

Cash register and computer programs for the hard goods retailer



Illustration—Carmen Console

By Donald E. Hassler
Fidelity Management Systems
Phoenix, AZ. 85012

Software that keeps track of retail sales, calculates sales people's commissions, and coordinates payroll and inventory helps this business run more smoothly.

The first full year I used my Commodore microcomputer with my own sales entry programs, it saved me $10,000!! That sounds like a lot. It saved time on audits from various agencies and, above all, it gave me a management tool that I had never had before, right here in the store.

The whole story started about 20 years ago. When National Cash Register (NCR) brought out its class five cash registers, it also provided a complete retail management program for those who wanted it. This program took all the cash register transactions and processed them by a mainframe computer, to produce a variety of management reports. We started using the NCR package way back in 1965. All the processing was done by NCR in Denver and it took forever (ten days or more) to get the reports back.

In 1971 one of our local banks took over the work and our turnaround time was improved. So we went along with it but never really got things organized as they should be. For instance, if we wanted any special reports or wanted to make even minor changes in the master records, action seemed to take two or three months. Even when it was working properly, our bookkeeping staff never seemed to understand all the reports. We made lots of costly, uncorrected mistakes. These were mostly input errors that went undetected for many months.

In 1978 I decided that we must have in-house data processing for the three stores that we were operating. So the search began.

This is lesson number one for all of you who want your own computer systems. If you are generating more than $750,000 and have more than five employees, it will *cost you* money not to have your own microcomputer. The lesson is: find someone who has the right software for your type of business!! Believe me, the software is out there. *(Editor's Note:* See our listing of business software in this issue)

So I located a computer expert in Tucson—Harry Goodkin—who had the exact programs that I needed. Harry had taken the basic data and reports from the NCR Retail Management System and was using them on his PDP-11 mini. Best of all, the system was already being run successfully by a retail jewelry chain in Tucson. The programs were written, tested, and running. And that is lesson number two. The lesson is: be sure that the software you choose has been thoroughly tested!!

The time between my original contact with Harry and the purchase was at least six months. After hearing the horror stories about others whose systems failed, I was not about to make a rash decision. We discussed many different systems and ways to process. The final choice was very fortunate. We purchased a Commodore 8032, an 8050 dual disk drive, and a 2022 printer. We also decided to buy the Business Enhancement Software (BEC) accounting programs. Harry would do the programming necessary to fit his PDP-11 programs to our system. The total price of the entire deal was *under* $6,000.

Mindy Feie, cashier, enters a sale. Later, the Sales Entry and Analysis System (Salent) will use this information to generate reports. General Manager John Courtney observes.

Since March of 1981 we have been using the system. It is the best boost our business can have. All the programs run well. We get full sales reports every Friday and two days after the close of every month. We could have them every day if we wanted. The balance sheet and income statement are finished by the tenth. There is *no* substitute for this speed of management communication. And this is lesson number three. The lesson is: if you have a system, use it for speed and accuracy. Don't expect it to immediately replace personnel.

Now you are probably thinking, "What's so great about a business accounting package? There are lots of those." You're right. The heart of our system is not the business accounting package. It's the sales entry and analysis system. We call it "Salent".

Salent takes every cash register transaction, either from the detail tape or from the actual invoices, and allows it to be input and listed on the computer. It then creates a datafile (or database) of these transactions for generating reports. And reports there are!!

The best part of the system is the "sales performance" module. There are MTD and YTD reports for each salesperson in each store, showing merchandise sold, non-merchandise sold, returns and number of transactions. By separating merchandise and non-merchandise there can be a separate commission paid. Technicians or other non-sales persons may also have a sales number if they are producing revenue.

Sales may be split between two salespersons. And sales may be split between several stores for the same sales number if one person works at, or is transferred to, a different location. Each store's sales total will match exactly the cash register daily totals.

Now here's the workhorse of this system. It's called a "sales edit list". This list is printed as the product of each day's store transactions. The sales edit list figures *must* equal the daily cash deposit. Just as a cash register with locking totals forces a balance, so does the sales edit list. The best part of it is that the list is in highly readable form. It's easy for a controller or auditor to locate errors or make adjustments.

The daily sales disks (one for each store) are posted periodically to a posting (or analysis) disk. This disk is the source of all the system reports. In addition to "sales performance" there is a "sales summary" (shows sales by class), "sales tax and non-merch report" and a "charge transactions list". All the charge transactions may be posted to BEC's accounts receivable module, if desired.

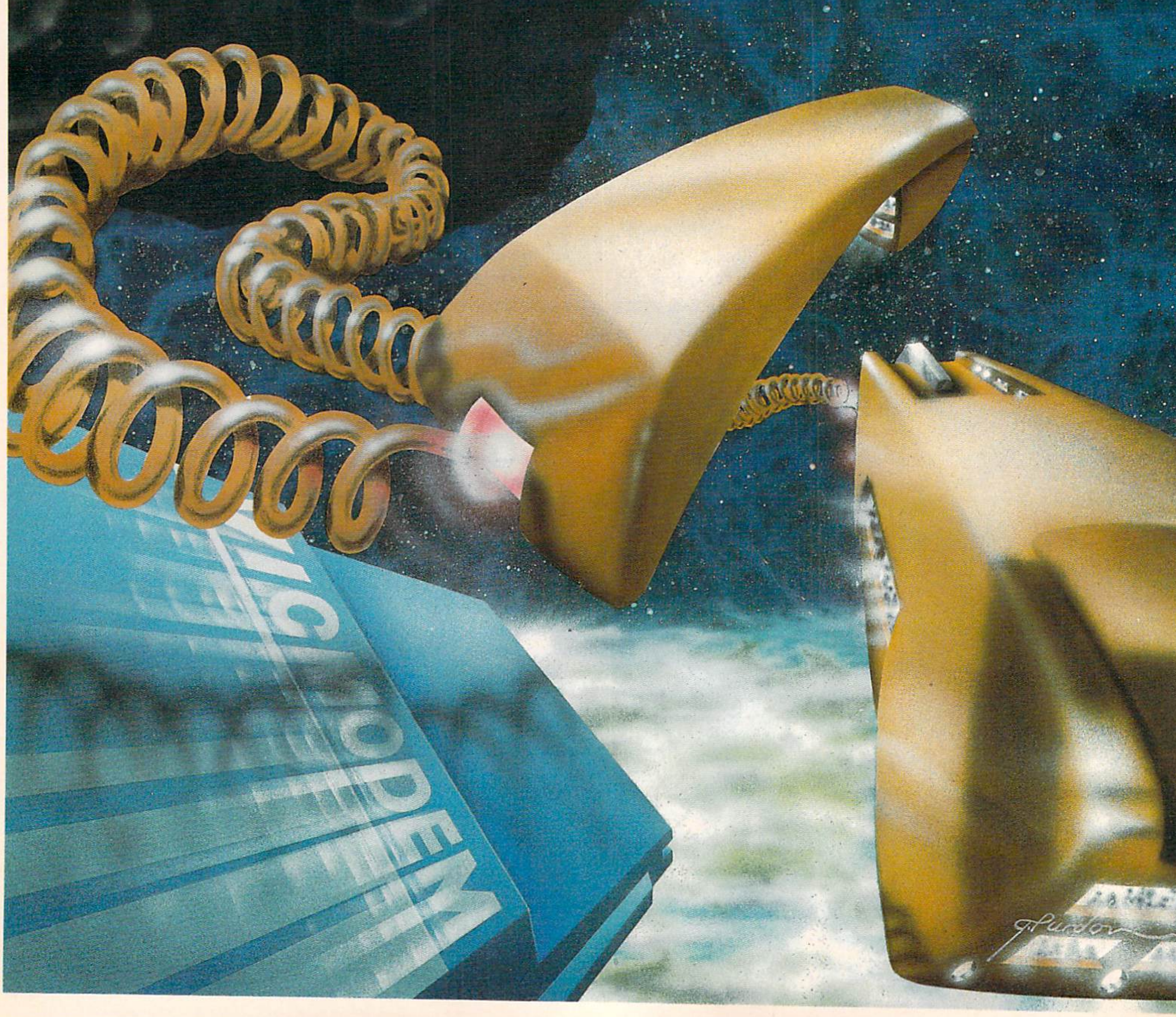
Bookkeeper Jeanne Reeves uses a CBM 8032 with a business accounting package for speed and accuracy.

That's the main story. The entire software package sells for $450. It supports up to nine store locations, 49 salespeople, 45 non-merchandise and 799 merchandise classifications. Combine it with a good database unit inventory system and you're set. If anyone would like further information please call me in Phoenix at 602-277-5711. Or read about the entire Business Enhancement Software system in the *Commodore Software Encyclopedia*. C

Telecommunications Gives Your Business an Important Edge

Using a modem you can access up-to-the-minute information that can help you tune your decision-making process and manage your business better.

By Walt Kutz
Business Computer Systems Product Manager



In today's business world, "next morning" information is no longer satisfactory. Today's business people must have up-to-the-minute data in order to gain an edge in the marketplace. Your computer, used with a modem, can provide this data by giving you access to huge national telecommunications networks, and thus increase management's ability to respond quickly when changes occur.

Information for the business community is stored in data bases that are accessed through telecommunications "time-sharing" systems. Some of the time-sharing systems available to the microcomputer user are CompuServe, Dow Jones Portfolio Management System, Dun & Bradstreet (Dunsprint), I.P. Sharp Associates, Inc. and The Source. I would like to explore just two of these in this article: Dun & Bradstreet and I.P. Sharp Associates.

Dun & Bradstreet's Dunsprint System

Dun & Bradstreet is one of a number of national business credit-reporting agencies. Their reports provide credit executives with objective, up-to-date payment information. This mutual exchange of information among credit executives is essential in today's business community. Computers now provide the most efficient, economical method for exchange of this information. For instance, Commodore's own national credit department is currently using the SuperPET and a Universal Data System 1200-baud modem to access Dun & Bradstreet's Dunsprint system. The major benefit has been a nearly 30% reduction in the cost of each report.

The information in each Dunsprint file is printed on a report specifically created to best display the information contained in that file. The format was designed by credit experts working directly with experienced technical personnel. Requests for reports are contained in the central file and are highly confidential. Elaborate procedures to assure information security are in effect at all times and access and exposure to credit files, equipment and programs are strictly controlled. The files are available only to qualified users who have a security code or password and a special account number.

I.P. Sharp Associates, Inc.

I.P. Sharp Associates is a private Canadian software and computer time-sharing company, founded in 1964. Users of the I.P. Sharp system have access to a growing list of publicly available data bases that are of interest to a variety of industries. The public data bases are grouped into five major categories: economics, finance, aviation, energy and insurance. These public data bases generally contain historical-numeric data called "time series" data. The number of time series contained in each data base varies from several hundred to several million, with the total number available exceeding twenty million.

With access to this type of data base the potential number of reports you can obtain is staggering. As an example, in the areas of economics and finance, over 28,000 monthly, quarterly and annual time series reports are available in the International Financial Statistics data base compiled by the International Monetary Fund for over 170 countries and country groupings. In addition, aggregate data for the world and over fifty selected regions is provided in this data base. Categories covered include exchange rates, international liquidity, banking, interest rates, prices and production, commodities, national accounts, government spending and international transactions. Annual series date back to 1948, quarterly to 1957 and monthly to 1965.

For those organizations associated with the aviation industry, the ICAO (International Civil Aviation Organization) data base provides international airline traffic statistics for over 600 airlines and 300 airports. The data is collected by the ICAO and is updated yearly, typically in October of the following year. Other segments of the I.P. Sharp aviation data base include Form 41 Data Base, ER586 Data Base, OAG and T6 Charter Data Base.

The energy data base includes such information as Quarterly Oil Statistics, API Weekly Statistical Bulletin, Liquified Petroleum Gas Report, Fuel Oil by Sulfur Content and much more. The insurance data base includes an actuarial data base containing primitive mortality information on insured lives, annuitants and the general population taken from over 200 tables published by regulatory actuarial bodies.

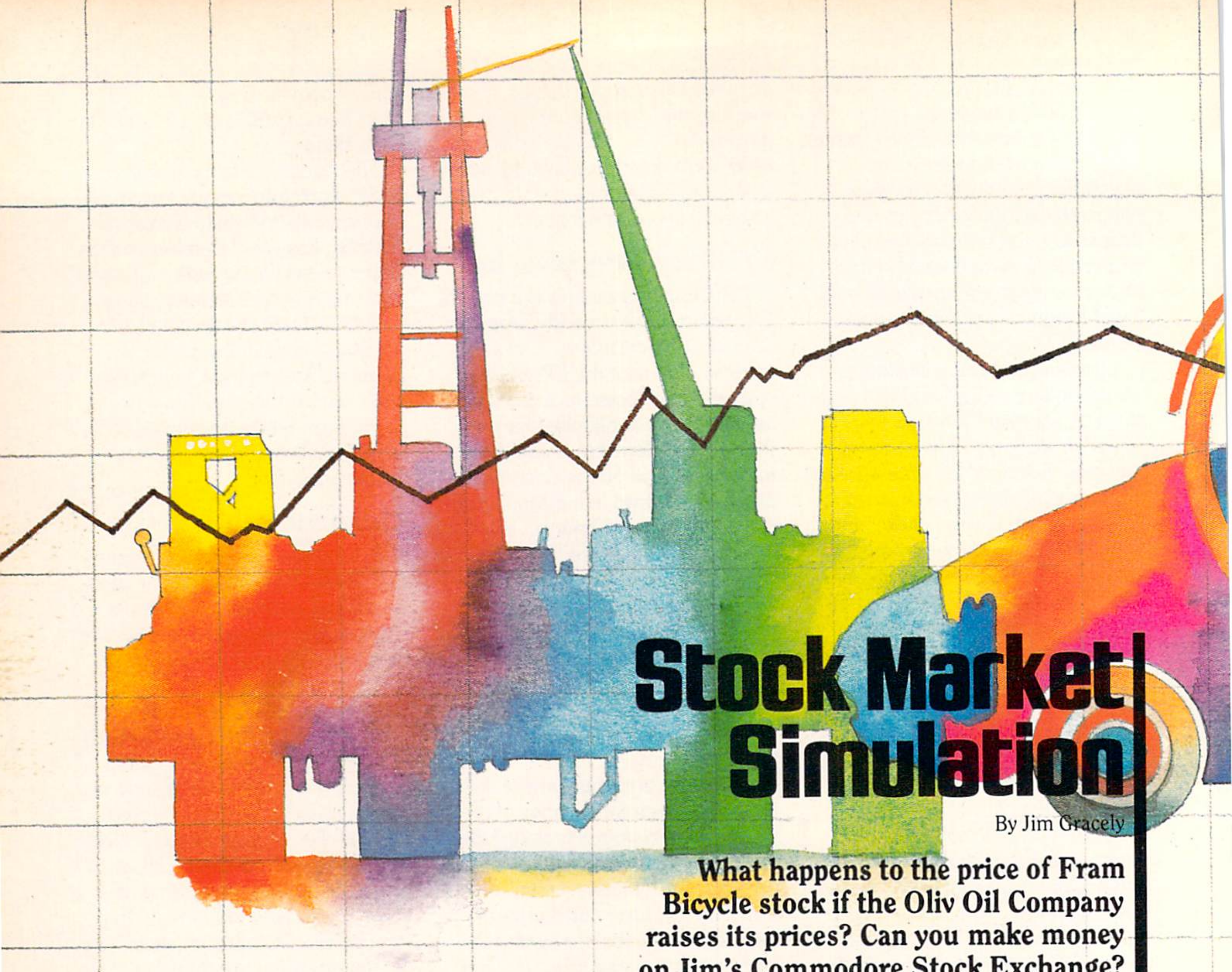
Electronic Mail

In addition to accessing these many data bases using their computer and modem, businesses can also gain access to another service they will undoubtedly find very valuable—electronic mail. Electronic mail is a medium of communication the likes of which the world has not seen before. Comparing it to the telephone or telex is missing the point. Its real strength lies in its ability to provide managers with all the information they need about everything that is happening everywhere—the direction in which other members of management are thinking and blow-by-blow accounts of decision-making processes—all without the need for a telephone or interminable meetings.

The electronic mailbox is a means of communication between people, not places. So the code assigned to an individual is the "address" to which a message is sent. The electronic mailbox is, therefore, completely removed from geography, so users can access mail from wherever they happen to be at the time.

The information in this article is far from inclusive. In fact, it shows just a tiny fragment of what is available to businesses using telecommunications time-sharing systems. But you can undoubtedly see that even the few services I've mentioned here are of enormous use to many different types of businesses. How about yours? C

Illustration—Greg Purdon



Stock Market Simulation

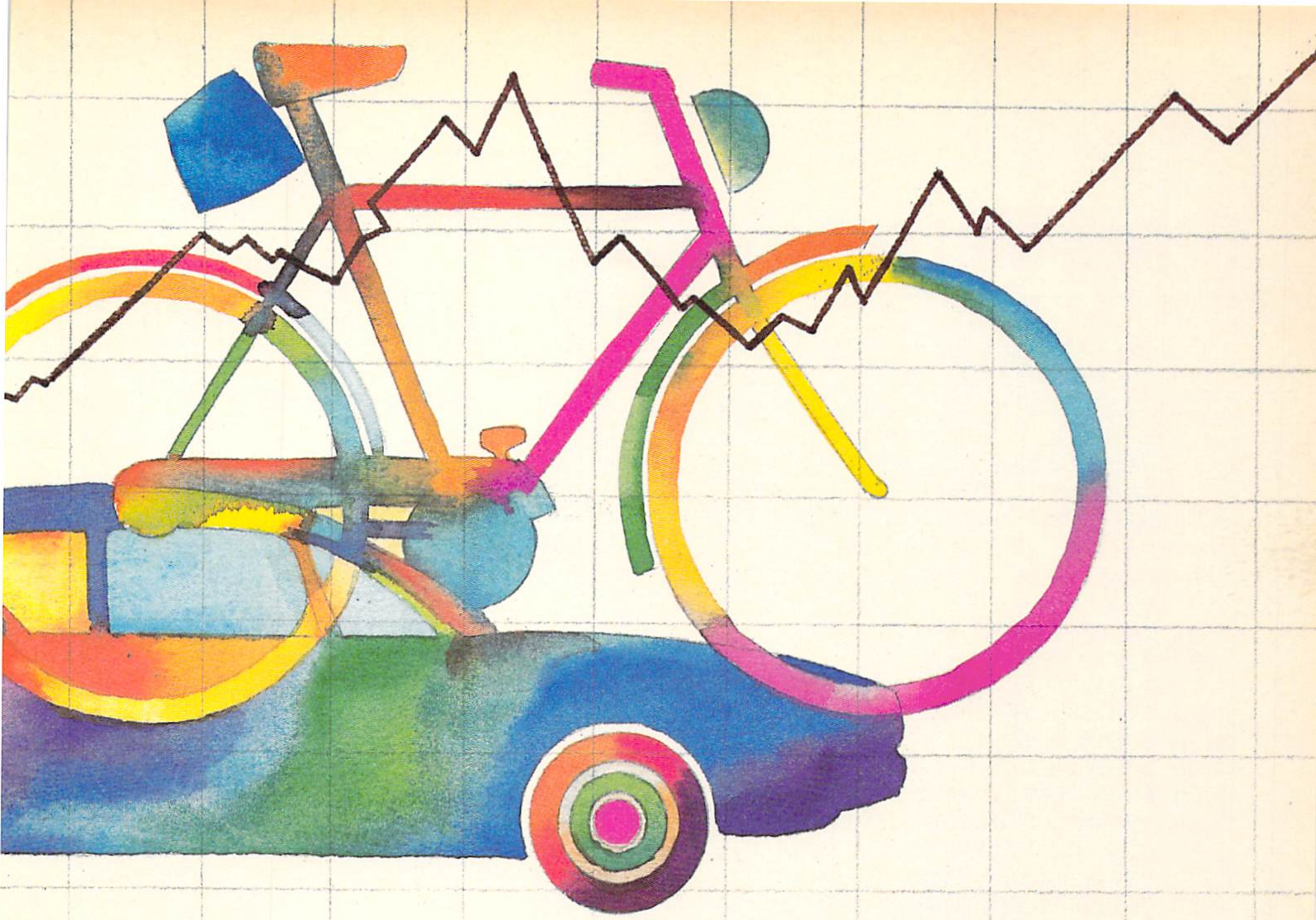
By Jim Gracely

What happens to the price of Fram Bicycle stock if the Oliv Oil Company raises its prices? Can you make money on Jim's Commodore Stock Exchange?

How can you have a computerized business special without a stock market simulation? You can't. So here's my version of the popular simulation. You get $5000 and 52 weeks to make your millions playing the stock market. I have set out to change the one most annoying feature of the simulations that I've seen. The amount that each stock changed in price and the direction of that change is always randomly generated. Now how can they call that a simulation? If the stock market actually changed randomly, it would be like a big lottery with people taking random chances on random changes. In real life there are economic principles which guide the changes of the various stocks. I have incorporated just a little of that into this program to allow a more realistic simulation.

There are five companies competing in this program with shares of each selling for $50 at the beginning of the program. The relationships between the companies and the price of their stock is relatively simple. There are two oil companies; when one goes up the other goes down (makes sense). There are two car companies; when the oil shares go up the car shares go down (who wants to buy a new car when gasoline is $1.50 a gallon?). The last company is a bike manufacturer; when the net change of oil and car shares is up, the bike shares go down (the more that people are driving, the less they are biking).

To create these interdependencies, I used a random seeding method. This means that the first change and direction (for oil company 1) is randomly generated and this is used as a seed for the change of the second oil company. The sum of the first two changes is used as a seed for the total of the second two changes (the car companies). Then the sum of the first four changes is used as a seed for the last change (the bike manufacturer). One of the interesting side effects of using this method is that the size and randomness of the changes decrease through the five stocks. If you want to take a chance on the "big score" put all your money on the first stock.



Illustration—Jack Freas

You've got even odds on making a lot or losing a lot. If you want to be more of a conservative, put your money on the last stock. Your money is relatively stable here. Don't count on making a killing, but if you lose some money it won't be much. Play the field any way you want and see how good your market instincts are.

Subroutines perform each of the major calculations, inputs, and displays of the program. There are subroutines to display each of the three program screens: Stock Market Screen at 400, Portfolio Screen at 721 and Broker's Window at 800. The main calculation subroutine is in lines 8 through 170. This subroutine calculates the weekly changes for each of the stocks. Line 195 performs some housekeeping by jumping to subroutines that round off numbers to the correct number of decimal places, check for high and low values of each stock and update the current value of any previously purchased shares. Lines 900 to 960 are the subroutine for buying and selling stocks. The subroutine beginning at 200 ends the program after 52 weeks.

The following list shows all of the variables in the program with their uses. The subscripted variables each have five subscripts, one for each company.

N$( ) = Names of companies
A$ ( ) = Abbreviated names of companies
ST ( ) = Current price of stock
L ( ) = Lowest price of stock
H ( ) = Highest price of stock
N ( ) = Number of shares owned
P ( ) = Purchased value of shares
C ( ) = Current value of shares
CS = Cash on hand
TT = Total assets
W = Week number
D1-D5 = Weekly change of each stock
DT = Sum of D1 and D2
BC = Background color
N$,Z$,A$,R$,B$,A,B,X = Input statement and miscellaneous variables

Stock Market Simulation

```
1 REM ***STOCK MARKET SIMULATION***
2 REM ***WRITTEN BY  JIM GRACELY***
3 N$(1)="OLIV OIL":N$(2)="BODY OIL":N$(3)="ODOM MOTORS":N$
  (4)="MILLI MOTORS"
4 N$(5)="FRAM BIKES":A$(1)="O. OIL":A$(2)="B. OIL"
  :A$(3)="O. MOTORS"
5 A$(4)="M. MOTORS":A$(5)="F. BIKES"
6 FOR X=1 TO 5:ST(X)=50:L(X)=50:H(X)=50:NEXT:W=0
  :CS=5000:BC=53281:POKE BC-1,0
7 GOTO 704
8 REM ***CALCULATIONS***
9 REM ***FIRST TWO***
10 D1=RND(1)*10
20 D1=INT(D1*10)/10
30 S=SGN((RND(1)*6)-3)
40 D1=D1*S
50 ST(1)=ST(1)+D1
55 IF ST(1)<0 THEN ST(1)=ST(1)-D1
60 D2=-(INT(D1*RND(1)*10)/10)
70 ST(2)=ST(2)+D2
75 IF ST(2)<0 THEN ST(2)=ST(2)-D2
80 REM ***SECOND TWO***
90 DT=D1+D2
100 D3=-DT/(RND(1)+.50)
120 D3=INT(D3*10)/10
130 D4=INT(RND(1)*D3*10)/10
140 ST(3)=ST(3)+D4
145 IF ST(3)<0 THEN ST(3)=ST(3)-D4
150 ST(4)=ST(4)+(D3-D4)
155 IF ST(4)<0 THEN ST(4)=ST(4)-(D3+D4)
157 REM ***LAST ONE***
160 D5=INT((DT+D3)*RND(1)*20)/10
165 ST(5)=ST(5)+D5
170 IF ST(5)<0 THEN ST(5)=ST(5)-D5
195 GOSUB 510:GOSUB 310:GOSUB 610:GOSUB 405
199 IF W<52 THEN W=W+1:RETURN
200 REM ***ENDING***
210 PRINT"[DOWN]AFTER 52 WEEKS (1 YEAR) THIS IS HOW THE"
220 PRINT"STOCKMARKET STANDS[DOWN3]"
230 PRINT"PRESS THE SPACE BAR TO SEE"
235 PRINT"YOUR FINAL TOTALS"
240 GET Z$:IF Z$=""THEN 240
250 IF Z$<>" "THEN 240
260 F=1:GOSUB 722
265 TT=INT(((TT+CS)-5000)*100)/100
270 IF TT>=0 THEN T$="MADE"
280 IF TT<0 THEN TT=-TT:T$="LOST"
290 PRINT"[DOWN2]HOPE YOU HAD FUN!"
```

```
295 PRINT"YOU "T$" $"TT" !!"
297 GET A$:IF A$=""THEN 297
298 POKE BC-1,14:POKE BC,6:PRINT CHR$(154)CHR$(147):END
300 REM ***LOWEST/HIGHEST CHECK***
310 FOR X=1 TO 5
320 IF ST(X)<L(X)THEN L(X)=ST(X)
330 IF ST(X)>H(X)THEN H(X)=ST(X)
340 NEXT:RETURN
400 REM ***STOCK MARKET SCREEN***
405 POKE BC,12:PRINT CHR$(5)
410 PRINT"[CLEAR,RVS,SPACE9]***STOCK MARKET***              ";
412 FOR X=1 TO 40:PRINT"[RVS]-[RVOFF]";:NEXT
413 PRINT"[RVS]WEEK-->[RVOFF]"W
415 PRINT"[DOWN2,RVS]STOCK[RVOFF]"," [RVS]LOW[RVOFF]
    ","[RVS]HIGH[RVOFF]","[RVS]PRESENT[RVOFF]"
420 FOR X=1 TO 5
430 PRINT"[DOWN]"A$(X),L(X),H(X),ST(X)
440 NEXT:RETURN
500 REM ***CONTROL DECIMAL PORTION***
510 FOR X=1 TO 5
520 ST(X)=INT(ST(X)*10)/10
530 P(X)=INT(P(X)*100)/100
540 NEXT:RETURN
600 REM ***UPDATE CURRENT VALUE***
610 FOR X=1 TO 5
620 C(X)=ST(X)*N(X)
630 NEXT:RETURN
700 REM ***START OF MAIN ROUTINE***
704 GOSUB 10:R$=""
705 PRINT"[DOWN2]DO YOU WANT TO VIEW YOUR PORTFOLIO (Y/N)":INPUT R$
710 IF LEFT$(R$,1)="N"THEN 704
720 IF LEFT$(R$,1)<>"Y"THEN PRINT"[UP5]";:GOTO 705
721 REM ***PORTFOLIO SCREEN***
722 POKE BC,14:PRINT CHR$(31)
725 PRINT"[CLEAR,RVS,SPACE13]PORTFOLIO                  ";
726 FOR X=1 TO 40:PRINT"[RVS]+[RVOFF]";:NEXT
730 PRINT"[DOWN3,RVS]STOCK[RVOFF]","[RVS]SHARES[RVOFF]","
    [RVS]PURCH[RVOFF]","[RVS]CURRENT[RVOFF]"
735 PRINT,,"[RVS]VALUE[RVOFF]","[RVS]VALUE[RVOFF]"
740 TT=0:PRINT"[DOWN]":FOR X=1 TO 5
750 PRINT A$(X),N(X),P(X),C(X)
755 TT=TT+C(X)
760 NEXT
770 PRINT"[DOWN]CASH $",,,CS
775 FOR X=1 TO 38:PRINT"@";:NEXT
780 PRINT:PRINT"TOTAL $",,,TT+CS
785 IF F=1 THEN RETURN
```

```
790 PRINT"[DOWN2]WOULD YOU LIKE TO MAKE ANY CHANGES (Y/N)"
    :R$="":INPUT R$
795 IF LEFT$(R$,1)="N"THEN 704
797 IF LEFT$(R$,1)<>"Y"THEN PRINT"[UP5]";:GOTO 790
800 REM ***BROKER'S WINDOW***
803 POKE BC,1:PRINT CHR$(144)
805 PRINT"[CLEAR,RVS,SPACE11]BROKER'S OFFICE                   [RVOFF]";
807 FOR X=1 TO 40:PRINT"[RVS]*[RVOFF]";:NEXT
810 PRINT:PRINT"[DOWN3,SPACE3,RVS]STOCK[RVOFF]",,"
    [RVS]SHARES[RVOFF]",
   "[RVS]PRICE[RVOFF,DOWN2]"
820 FOR X=1 TO 5
830 PRINT"[RVS]"X"[RVOFF]"N$(X),N(X),ST(X)
840 NEXT
850 PRINT"[DOWN2]YOU HAVE $"CS" ON HAND"
860 PRINT"[DOWN]WHICH STOCK (1-5) DO YOU WANT":PRINT
    "TO CHANGE (0 TO EXIT)"
   :INPUT A$
870 A=VAL(A$):IF A<0 OR A>5 THEN PRINT"[UP4]";:GOTO 860
880 IF A=0 THEN 704
890 PRINT"[UP3]YOU HAVE ENOUGH MONEY TO BUY "
892 N$=STR$(INT(CS/ST(A)))
895 PRINT"       "N$" SHARES      "
900 REM ***BUYING AND SELLING***
910 PRINT"ENTER NUMBER OR SHARES THAT YOU WISH TO BUY
    (+)/SELL(-)":INPUT B$
920 B=VAL(B$):IF B>INT(CS/ST(A))OR B<(-N(A))THEN PRINT
    "[UP3]";:GOTO 910
930 IF B<0 THEN P(A)=P(A)+(P(A)/N(A))*B
940 IF B>0 THEN P(A)=P(A)+ST(A)*B
950 N(A)=N(A)+B:CS=CS-ST(A)*B:CS=INT(CS*100)/100
960 GOTO 805
```

C

Computers Help Your Business

Business Software for Commodore Computers

(Continued from page 40)

Available From	Program Name	Computer	Drive	Acctg	Bus Mgmt	Data Mgmt	Financial	Farm	Inventory	Invoice Billing Order Entry	Legal	Mail List	Medical/Dental	Payroll	Real Estate	Retail	Specialized	Statistics	Tax	Word Processing
Micro Computer Industries, Ltd.	Create-A-Base	8032	4040 8050			X														
Creative Equipment	Master List	8032 8096 SuperPET	8050 8250		X	X			X			X								
Sof-Tec	Subdivision Analysis	Contact Vendor		X	X		X								X				X	
Micro Software Systems	Maxi-Calc	8032 4032					X													
	Finance-Calc	8032 4016					X													
Channel Data Systems	Omnfile	2001	2040 or Datassette			X			X			X								
	General Ledger	4016	2040 or Datassette	X																
Delta Software	Payroll Systems	8032	4040 8050	X										X						
Business Enhancement Compuservice	Accounting III & IV—BEC	Any CBM	8050	X	X				X	X		X		X					X	
Briley Software	Business Researcher	16 or 32K			X															
AB Computers	Flex File II	4032 8032	4040 8050			X														
	Paper Mate	Contact Vendor																		X
Dr. William A.C. Schmidt	Stock Market Decisions	8K	Datassette			X	X													
Impact	Partrac	8032	8050			X			X	X										
United Software of America	Request	8032	8050			X														
Data Max Software	Mailman	8032	4040 8050									X								
AQR Products	Ticker Tape Info. Processing	Ticker Tape Interface Cable				X	X													
Connecticut microComputer	CmC Word Processor	Contact Vendor																		X
Cognitive Products	Textcase II	8KPET																		X
Optimized Data Systems	Word Processor	8KPET																		X

System: Commodore 64

Available From	Program Name	Computer	Drive	Acctg	Bus Mgmt	Data Mgmt	Financial	Farm	Inventory	Invoice Billing Order Entry	Legal	Mail List	Medical/Dental	Payroll	Real Estate	Retail	Specialized	Statistics	Tax	Word Processing
Commodore Dealers	Easy Calc 64	64	1541 or Datassette		X															
	Easy Plot 64	64	1541 or Datassette		X															
	Easy Finance 64	64	1541 or Datassette				X													
	Easy Schedule 64	64	1541 or Datassette		X															
	Easy File 64	64	1541 or Datassette			X														
	Easy Script 64	64	1541 or Datassette																	X

Business Software for Commodore Computers

Available From	Program Name	Computer	Drive	Acctg	Bus Mgmt	Data Mgmt	Financial	Farm	Inventory	Invoice Billing Order Entry	Legal	Mail List	Medical/Dental	Payroll	Real Estate	Retail	Specialized	Statistics	Tax	Word Processing
Commodore Dealers (continued)	Word Machine	64	1541 or Datassette																	X
	Name Machine	64	1541 or Datassette									X								
	Easy Mail 64	64	1541 or Datassette									X								
	General Ledger	64	1541	X																
	Receivable/Billing	64	1541	X						X										
	Accounts Payable/ Checkwriting	64	1541	X						X										
	Payroll	64	1541											X					X	
	Inventory Management	64	1541			X			X											
Powerbyte	The Billing Solver	64	1541			X														
	Cash Flow Model	64	1541			X														
	Predictor-Linear Regression	64	1541		X															
	Depreciator	64	1541				X													
	Statistics Sadistics	64	1541															X		
	Taxman	64	1541	X															X	
	Net Worth Statement	64	1541				X													
	Investment Analyst	64	1541				X													
	Stock Ticker Tape	64	1541			X	X													
	Super Broker	64	1541			X	X													
	Profit Sharing Plan	64	1541		X															
	Lease/Buy?	64	1541		X															
	Syndicator	64	1541		X		X													
	Order Tracker	64	1541	X						X										
	The Bidder—My Profit Margin	64	1541		X															
	Business Calendar	64	1541		X															
	Client Tickler	64	1541			X														
TOTL. Software	TOTL. Time Manager	64	1541		X															
	Research Assistant		1541			X														
RAK Electronics	Sales/Expense	64	1541 or Datassette	X		X														

Available From	Program Name	Computer	Drive	Acctg	Bus Mgmt	Data Mgmt	Financial	Farm	Inventory	Invoice Billing Order Entry	Legal	Mail List	Medical/Dental	Payroll	Real Estate	Retail	Specialized	Statistics	Tax	Word Processing
Abacus Software	Quick Chart	64	1541 or Datassette			X														
Cyberia, Inc.	Cyber-Farmer	64	1541	X	X	X	X	X	X											

System: VIC 20

Available From	Program Name	Computer	Drive	Acctg	Bus Mgmt	Data Mgmt	Financial	Farm	Inventory	Invoice Billing Order Entry	Legal	Mail List	Medical/Dental	Payroll	Real Estate	Retail	Specialized	Statistics	Tax	Word Processing
Commodore Dealers	VIC File	VIC w/ 16K expansion	1541			X			X											
	VIC Writer	VIC w/ 8K expansion	1541																	X
	Simplicalc	VIC w/ 8K expansion	1541				X													
TOTL. Software	TOTL. Time Manager	VIC w/ 8K expansion	Datassette		X															
	Research Assistant	VIC w/ 8K expansion	Datassette			X														
	TOTL. Label											X								

This is a sample of what Marty's program prints out after he enters all his data.

OFFER

SHIPPER: LOUIS BERNARD. BORDEAUX

FF= .15

VINT	WINE	FOB FF	FOB $	NY LANDED	IN STORE $
79	CHATEAU PHELAN SEGUR	589.00	90.11	94.79	106.29
80	CHATEAU PHELAN SEGUR	326.00	51.45	56.13	62.90
81	CHATEAU PHELAN SEGUR	450.00	69.68	74.36	83.36
PAUILLAC					
76	CHATEAU CROIZET BAGES	698.00	106.13	110.81	122.02
77	CHATEAU CROIZET BAGES	388.00	60.56	65.24	73.13
79	CHATEAU CROIZET BAGES	543.00	83.35	88.03	98.70
81	CHATEAU CROIZET BAGES	465.00	71.88	76.56	85.83
75	CHATEAU DUHART MILON	1473.00	220.06	224.74	242.98
79	CHATEAU DUHART MILON	853.00	128.92	133.60	147.13
80	CHATEAU DUHART MILON	527.00	81.00	85.68	96.06
81	CHATEAU DUHART MILON	651.00	99.23	103.91	114.40
70	CHATEAU FORTS LATOUR	2170.00	322.52	327.20	353.81
73	CHATEAU FORTS LATOUR	853.00	128.92	133.60	147.13
74	CHATEAU FORTS LATOUR	698.00	106.13	110.81	122.02
76	CHATEAU FORTS LATOUR	853.00	128.92	133.60	147.13
74	CHATEAU GRAND PUY LACOSTE	589.00	90.11	94.79	106.29
75	CHATEAU GRAND PUY LACOSTE	1163.00	174.49	179.17	195.52
78	CHATEAU GRAND PUY LACOSTE	1008.00	151.70	156.38	170.64
79	CHATEAU GRAND PUY LACOSTE	853.00	128.92	133.60	147.13
80	CHATEAU GRAND PUY LACOSTE	450.00	69.68	74.36	83.36
81	CHATEAU GRAND PUY LACOSTE	713.00	108.34	113.02	124.45
76	CHATEAU HAUT BATAILLEY	775.00	117.45	122.13	134.49
78	CHATEAU HAUT BATAILLEY	853.00	128.92	133.60	147.13
79	CHATEAU HAUT BATAILLEY	651.00	99.23	103.91	114.40
80	CHATEAU HAUT BATAILLEY	419.00	65.12	69.80	78.24
81	CHATEAU HAUT BATAILLEY	620.00	94.67	99.35	109.38
69	CHATEAU LAFITE ROTHSCHILD	2480.00	368.09	372.77	403.10
71	CHATEAU LAFITE ROTHSCHILD	5890.00	869.36	874.04	936.37
75	CHATEAU LAFITE ROTHSCHILD	6200.00	914.93	919.61	985.19
76	CHATEAU LAFITE ROTHSCHILD	4340.00	641.51	646.19	692.24
78	CHATEAU LAFITE ROTHSCHILD	5425.00	801.00	805.68	863.13
79	CHATEAU LAFITE ROTHSCHILD	3565.00	527.58	532.26	570.18
80	CHATEAU LAFITE ROTHSCHILD	2325.00	345.30	349.98	378.45
81	CHATEAU LAFITE ROTHSCHILD	3255.00	482.01	486.69	526.32
67	CHATEAU LATOUR	3875.00	573.15	577.83	619.01
70	CHATEAU LATOUR	5890.00	869.36	874.04	936.37
71	CHATEAU LATOUR	4340.00	641.51	646.19	692.24
73	CHATEAU LATOUR	1938,00	288.41	293.09	316.92
74	CHATEAU LATOUR	2325.00	345.30	349.98	378.45
76	CHATEAU LATOUR	3255.00	482.01	486.69	526.32
79	CHATEAU LATOUR	3255.00	482.01	486.69	526.32
80	CHATEAU LATOUR	1628.00	242.84	247.52	267.63
81	CHATEAU LATOUR	2868.00	425.12	429.80	464.79
75	CHATEAU LYNCH BAGES	1628.00	242.84	247.52	267.63
76	CHATEAU LYNCH BAGES	1085.00	163.02	167.70	183.00
77	CHATEAU LYNCH BAGES	512.00	78.79	83.47	93.59
78	CHATEAU LYNCH BAGES	1054.00	158.47	163.15	178.03
79	CHATEAU LYNCH BAGES	775.00	117.45	122.13	134.49

OFFER

SHIPPER: LOUIS BERNARD, BORDEAUX

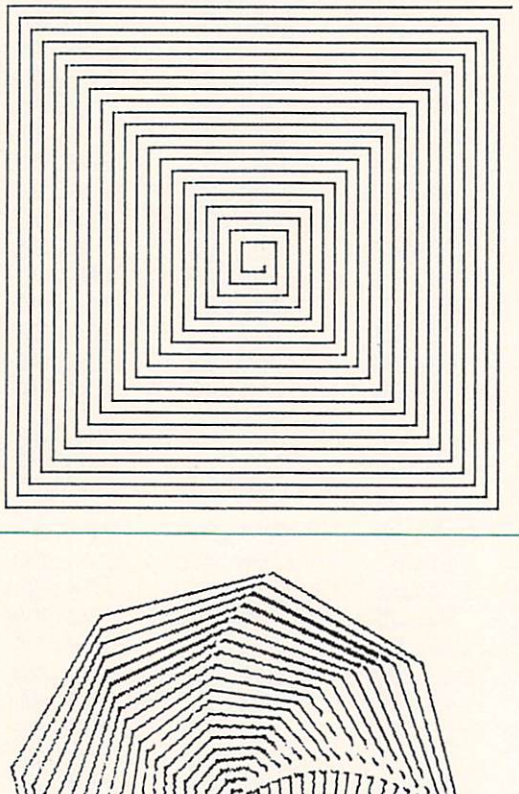
FF= .15

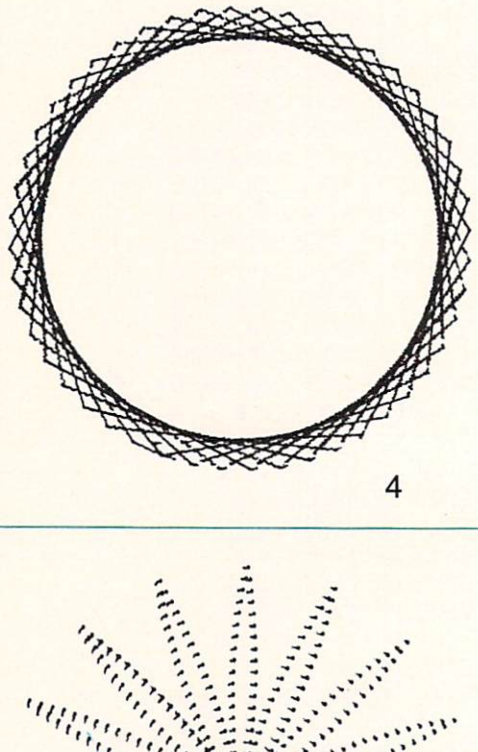
VINT	WINE	FOB FF	FOB $	NY LANDED	IN STORE $
80	CHATEAU LYNCH BAGES	512.00	78.79	83.47	93.59
81	CHATEAU LYNCH BAGES	729.00	110.69	115.37	127.04
75	CHATEAU MOUTON ROTHSCHILD	5890.00	869.36	874.04	936.37
76	CHATEAU MOUTON ROTHSCHILD	4650.00	687.08	691.76	741.07
78	CHATEAU MOUTON ROTHSCHILD	3720.00	550.37	555.05	594.59
79	CHATEAU MOUTON ROTHSCHILD	2713.00	402.34	407.02	440.14
80	CHATEAU MOUTON ROTHSCHILD	1628.00	242.84	247.52	267.63
81	CHATEAU MOUTON ROTHSCHILD	2790.00	413.66	418.34	452.39
73	CHATEAU PICHON BARON	713.00	108.34	113.02	124.45
75	CHATEAU PICHON BARON	1628.00	242.84	247.52	267.63
76	CHATEAU PICHON BARON	930.00	140.24	144.92	159.60
77	CHATEAU PICHON BARON	527.00	81.00	85.68	96.06
78	CHATEAU PICHON BARON	930.00	140.24	144.92	159.60
79	CHATEAU PICHON BARON	682.00	103.78	108.46	119.43
80	CHATEAU PICHON BARON	543.00	83.35	88.03	98.70
81	CHATEAU PICHON BARON	678.00	103.19	107.87	118.78
71	CHATEAU PICHON LALANDE	1860.00	276.95	281.63	304.52
74	CHATEAU PICHON LALANDE	620.00	94.67	99.35	109.38
75	CHATEAU PICHON LALANDE	1938.00	288.41	293.09	316.92
76	CHATEAU PICHON LALANDE	1318.00	197.27	201.95	218.34
77	CHATEAU PICHON LALANDE	589.00	90.11	94.79	106.29
78	CHATEAU PICHON LALANDE	1938.00	288.41	293.09	316.92
79	CHATEAU PICHON LALANDE	1194.00	179.05	183.73	200.50
80	CHATEAU PICHON LALANDE	527.00	81.00	85.68	96.06
81	CHATEAU PICHON LALANDE	930.00	140.24	144.92	159.60
ST. JULIEN					
67	CHATEAU BEYCHEVELLE	1395.00	208.59	213.27	230.58
70	CHATEAU BEYCHEVELLE	2480.00	368.09	372.77	403.10
74	CHATEAU BEYCHEVELLE	713.00	108.34	113.02	124.45
75	CHATEAU BEYCHEVELLE	2325.00	345.30	349.98	378.45
76	CHATEAU BEYCHEVELLE	1240.00	185.81	190.49	207.88
79	CHATEAU BEYCHEVELLE	899.00	135.68	140.36	154.58
80	CHATEAU BEYCHEVELLE	543.00	83.35	88.03	98.70
81	CHATEAU BEYCHEVELLE	837.00	126.57	131.25	144.54
76	CHATEAU BRENAIRE DUCRU	930.00	140.24	144.92	159.60
78	CHATEAU BRENAIRE DUCRU	930.00	140.24	144.92	159.60
79	CHATEAU BRENAIRE DUCRU	775.00	117.45	122.13	134.49
80	CHATEAU BRENAIRE DUCRU	527.00	81.00	85.68	96.06
81	CHATEAU BRENAIRE DUCRU	682.00	103.78	108.46	119.43
67	CHATEAU DUCRU BEAUCAILLOU	1783.00	265.63	270.31	292.27
70	CHATEAU DUCRU BEAUCAILLOU	3100.00	459.23	463.91	501.68
71	CHATEAU DUCRU BEAUCAILLOU	2248.00	333.98	338.66	366.21
73	CHATEAU DUCRU BEAUCAILLOU	1085.00	163.02	167.70	183.00
74	CHATEAU DUCRU BEAUCAILLOU	930.00	140.24	144.92	159.60
75	CHATEAU DUCRU BEAUCAILLOU	2325.00	345.30	349.98	378.45
76	CHATEAU DUCRU BEAUCAILLOU	1318.00	197.27	201.95	218.34
78	CHATEAU DUCRU BEAUCAILLOU	1783.00	265.63	270.31	292.27
79	CHATEAU DUCRU BEAUCAILLOU	1163.00	174.49	179.17	195.52
80	CHATEAU DUCRU BEAUCAILLOU	620.00	94.67	99.35	109.38

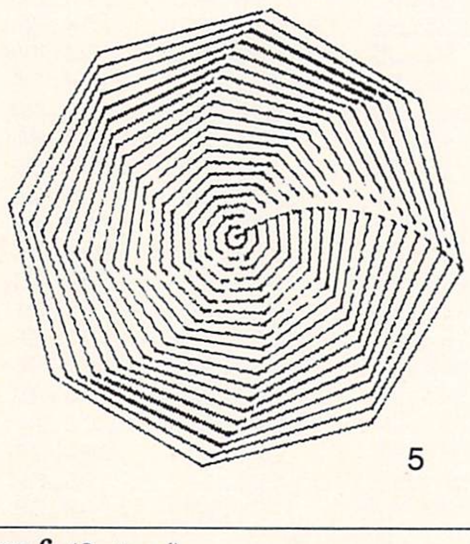
C

Advanced Bit Mapped Graphics (Continued from page 33)

several (not very imaginative, I'm afraid) examples of using the high-resolution routines. Each example is less than ten BASIC lines long, and a drawing of each is included in Figure 6 (which incidentally were done with a digital plotter driven by our straight line algorithm!). In every case BASIC is the speed-limiter. When you create your own programs, lines 10-100 from the example program should always be present as a preamble, but that line 10 (written for disk users, but I believe simply adaptable to tape) should be deleted after the first RUN. Tape users may be better off loading the machine code portion prior to loading HRTEST. Note also, because of my use of mnemonics to access the several routines, the variable names IN, RS, CL, DR, PX, and MV are reserved and


3


4


5

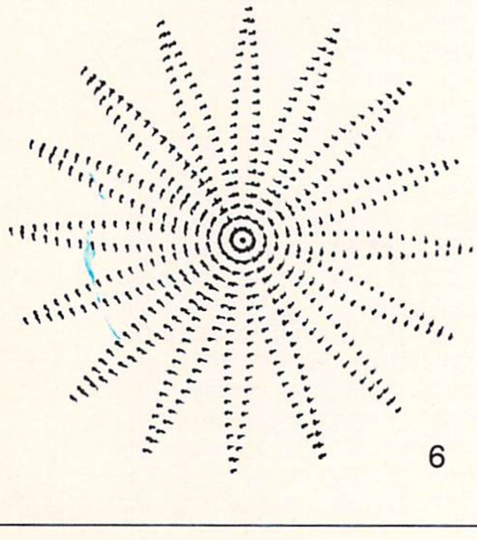
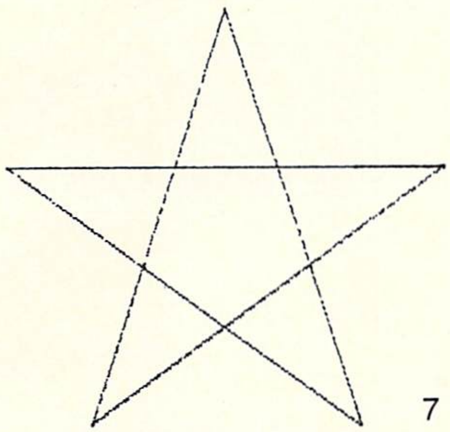

6

Figure 6. (Continued)

can *not* be used elsewhere.

The way the entry points are defined (by way of fixed jump vectors), it will not be necessary to change your programs, even if major modifications were to be made in the assembly source, as I may do in possible future articles on graphics. Even just staying within the high-resolution mode (no sprites yet), there are a number of topics that could be covered, such as high-speed circle and arc drawing, split-screen effects, colors, high-res character and shape sets, vector graphics, smooth X,Y scrolling of landscapes, animation techniques, graphic aids such as light pen input of "rubber band" lines, 3-D techniques with hidden line removal, or graphic fill! All of these and more are possible on the Commodore 64. C


7

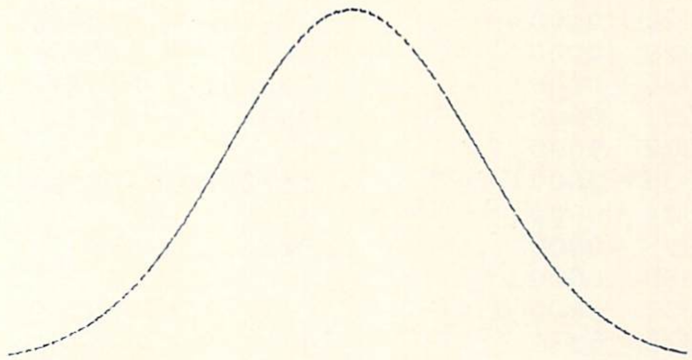

8

Figure 6. (Continued)

Listing #1: Complete Assembly Source Code

```
LINE# LOC    CODE          LINE

00002  0000                ;
00003  0000                ;** HRSUPP/64 **
00004  0000                ;
00005  0000                ORIGIN = $6000
00006  0000                ;
00007  0000                ;** EQUATES **
00008  0000                ;
00009  0000                ;SYSTEM ROUTINES
00010  0000                ;
00011  0000                ERROR   = $A437          ;PRINT ERROR MESSAGE
```

```
LINE# LOC    CODE         LINE

      0000                EVAEXP = $AD9E          ;EVALUATE EXPRESSION
      0000                CHKCOM = $AEFD          ;CHECK FOR COMMA
      0000                FLTFIX = $B1AA          ;CONVERT TO FIXED IN Y
                                                  (LOW) AND A (HIGH)
      0000                ;
      0000                ;VECTORS
      0000                ;
      0000                ERRVEC = $0300          ;ERROR ROUTINE
      0000                WARMV  = $0302          ;BASIC WARM START
      0000                ;
      0000                ;HI-RES STUFF
      0000                ;
      0000                VIC    = $D000          ;ADDRESS OF VIC CHIP
      0000                HRCTRL = VIC+17         ;MODE CONTROL
      0000                HRMREG = VIC+24         ;MEMORY CONTROL
      0000                ;
      0000                SCREEN = $0400          ;1K SCREEN
      0000                SCREND = SCREEN+999     ;LAST SCREEN LOC'N
      0000                BASE   = $2000          ;START OF 8K BYT
      0000                HRLAST = BASE+7999      ;LAST LOC'N
      0000                RAM    = $033C          ;USE CASSETTE BUFFER
      0000                ;
      0000                ;**ZERO PAGE**
      0000                ;
      0000                BYT    = $FD            ;BYT POINTER
      0000                ;
      0000                       *=RAM
      033C                ;
      033C                X1     *=*+2            ;X COORDINATE (0 - 319)
      033E                X2     *=*+2
      0340                Y1     *=*+2            ;Y COORDINATE (0 - 199)
      0342                Y2     *=*+2
      0344                BITNO  *=*+1            ;ON BIT IS PIXEL
      0345                DELTX  *=*+2            ;X2-X1
      0347                DELTY  *=*+2            ;Y2-Y1
      0349                E      *=*+2
      034B                T      *=*+2
      034D                C      *=*+2
      034F                I      *=*+1            ;DIRECTION POINTER
      0350                TEMP   *=*+2
      0352                ERVEC  *=*+2            ;HOLDS SYSTEM ERROR
                                                  VECTOR
      0354                ;
      0354                ;CONSTANTS
      0354                ;
      0354                XMAX=320
      0354                YMAX=200
      0354                COLS=40                 ;NUMBER OF COLUMNS/ROW
```

```
LOC    CODE        LINE
0354               COLOR=$50                    ;FOREGROUND/BACKGROUND
                                                = BLACK/GREEN
0354               ;
0354                      *=ORIGIN
6000               ;
6000               ;JUMP TABLE FOR COVENIENT ENTRY POINTS
6000               ;
6000   4C 90 62    JINIT  JMP HRINIT            ;INITIALIZE
6003   4C BD 62    JREST  JMP HRREST            ;RESTORE
6006   4C 5B 62    JCLR   JMP CLRHR             ;CLEAR SCREEN
6009   4C 89 60    JDRAW  JMP VECPLT            ;DRAW STRAIGHT LINE
600C   4C EC 62    JSETPX JMP SETPIX            ;TURN ON PIXEL
600F               ;
600F               ;FALL THROUGH TO MOVE ROUTINE
600F               ;
600F               ;HRADDR - GIVEN X-COORD (2 BYTES)
600F               ;AND Y-COORD (1 BYTE)
600F               ;CALCULATE BYT ADDRESS AND BITNO
600F               ;
600F               ;CLOBBERS X, LEAVES Y=0
600F               ;
600F               ;ENTER HERE IF FROM BASIC
600F               ;
600F   20 E2 62    HRMOVE JSR GETVAL            ;GET X1
6012   8C 3C 03           STY X1
6015   8C 3E 03           STY X2                ;FOR RNGCHK
6018   8D 3D 03           STA X1+1
601B   8D 3F 03           STA X2+1
601E   20 E2 62           JSR GETVAL            ;GET Y1
6021   8C 40 03           STY Y1
6024   8C 42 03           STY Y2
6027   8D 43 03           STA Y2+1
602A   20 C6 61           JSR RNGCHK
602D               ;
602D               ;ENTER HERE IF X1, Y1 ARE SET
602D               ;
602D   A9 00       HRADDR LDA #0                ;SET HIGH BYTE TO ZERO
602F   85 FE              STA BYT+1
6031   38                 SEC                   ;FORM 199-Y1
6032   A9 C7              LDA #YMAX-1
6034   ED 40 03           SBC Y1
6037   48                 PHA                   ;SAVE RESULT ON STACK
6038   29 F8              AND #$F8              ;FORM ROW #
603A   0A                 ASL A                 ;MULT BY 2
603B   26 FE              ROL BYT+1
603D   0A                 ASL A                 ;MULT BY 4
603E   26 FE              ROL BYT+1
6040   0A                 ASL A                 ;MULT BY 8
6041   26 FE              ROL BYT+1
```

```
LOC   CODE              LINE

6043  48                PHA                 ;SAVE ON STACK
6044  8D 50 03          STA TEMP            ;AND IN TEMP
6047  A5 FE             LDA BYT+1
6049  8D 51 03          STA TEMP+1          ;TEMP HAS 8*Y
604C  68                PLA                 ;RESTORE A
604D  0A                ASL A               ;MULT BY 16
604E  26 FE             ROL BYT+1
6050  0A                ASL A               ;MULT BY 32
6051  26 FE             ROL BYT+1           ;(CARRY STILL CLEAR)
6053  6D 50 03          ADC TEMP            ;FORM 32+8 = 40*
6056  85 FD             STA BYT             ;INTO BYT
6058  A5 FE             LDA BYT+1
605A  6D 51 03          ADC TEMP+1
605D  85 FE             STA BYT+1
605F  AD 3C 03          LDA X1              ;NOW ADD CHAR
6062  29 F8             AND #$F8
6064  65 FD             ADC BYT
6066  85 FD             STA BYT
6068  AD 3D 03          LDA X1+1
606B  65 FE             ADC BYT+1
606D  85 FE             STA BYT+1           ;(CARRY STILL CLEAR)
606F  68                PLA                 ;NOW ADD LINE
6070  29 07             AND #7              ;BY MASKING HIGH BITS
6072  65 FD             ADC BYT
6074  85 FD             STA BYT
6076  A5 FE             LDA BYT+1           ;FINISH BY ADDING BASE
6078  69 20             ADC #>BASE
607A  85 FE             STA BYT+1
607C  AD 3C 03          LDA X1              ;SET BITNO
607F  29 07             AND #7              ;IS LOW 3 BITS
6081  AA                TAX                 ;AND INDEX TO TABLE
6082  BD 29 63          LDA MSKTB,X
6085  8D 44 03          STA BITNO
6088  60                RTS                 ;BYT AND BITNO NOW SET
6089                ;
6089                ;*** FASTPLOT ***
6089                ;
6089                ;GRAPHIC SUBROUTINE FOR LINE DRAWING
6089                ;ON 320*200 HI-RES MEMORY
6089                ;
6089                ;ORIGINALLY WRITTEN AS VECTOR GENERATOR
6089                ;FOR HOUSTON INSTRUMENT HIPLOT
6089                ;DIGITAL INCREMENTAL PLOTTER
6089                ;
6089                ;MORE EFFICIENT ALGORITHM BY W. MCWORTER
6089                ;IN BYTE MAY 1981, P14
6089                ;
6089                ;RE-WRITTEN FOR MTU VISIBLE MEMORY (TM)
6089                ;BY F. COVITZ, AUG. 1981
```

```
LOC   CODE       LINE
6089             ;REVISED NOV. 1982 FOR CBM-64
6089             ;
6089             ;
6089             ;*****************************
6089             ;* TYPO IN ORIGINAL LETTER    *
6089             ;*                            *
6089             ;* IT READS; A$="RQVWPS...." *
6089             ;*                  *         *
6089             ;*SHOULD BE; A$="RQVWRS...." *
6089             ;*                            *
6089             ;*****************************
6089             ;
6089             ;COME IN WITH X1,Y1 AND X2,Y2
6089             ;AND FIRST PIXEL SET
6089             ;I.E. BYT,BYT+1, AND BITNO ARE SET
6089             ;VIA CALL TO PIXADR
6089             ;ROUTINE DRAWS BEST STRAIGHT LINE
6089             ;LEAVES WITH X1_X2,Y1_Y2
6089             ;
6089             ;LEAVES WITH Y=0, X CLOBBERED
6089             ;ROUTINE CHECKS FOR OVERFLOW
6089             ;
6089             ;**VECPLT**
6089             ;
6089             ;ENTER HERE FROM BASIC
6089             ;
6089  20 E2 62   VECPLT JSR GETVAL      ;GET X-COORD
608C  8C 3E 03          STY X2
608F  8D 3F 03          STA X2+1
6092  20 E2 62          JSR GETVAL      ;GET Y-COORD
6095  8C 42 03          STY Y2
6098  8D 43 03          STA Y2+1
609B             ;
609B             ;ENTER HERE IF X2, Y2 ALREADY SET
609B             ;
609B  20 C6 61   VECPL1 JSR RNGCHK      ;CHECK X2,Y2 OVERFLOW
609E  38                SEC             ;FORM DELTX (SIGNED)
609F  AD 3E 03          LDA X2
60A2  ED 3C 03          SBC X1
60A5  8D 45 03          STA DELTX
60A8  AD 3F 03          LDA X2+1
60AB  ED 3D 03          SBC X1+1
60AE  8D 46 03          STA DELTX+1
60B1  38                SEC             ;FORM DELTY (SIGNED)
60B2  AD 42 03          LDA Y2
60B5  ED 40 03          SBC Y1
60B8  8D 47 03          STA DELTY
60BB  AD 43 03          LDA Y2+1
60BE  ED 41 03          SBC Y1+1
```

```
LOC    CODE        LINE

60C1                ;
60C1   8D 48 03              STA DELTY+1
60C4   AD 3E 03              LDA X2          ;X1,Y1_X2,Y2
60C7   8D 3C 03              STA X1
60CA   AD 3F 03              LDA X2+1
60CD   8D 3D 03              STA X1+1
60D0   AD 42 03              LDA Y2
60D3   8D 40 03              STA Y1
60D6   AD 43 03              LDA Y2+1
60D9   8D 41 03              STA Y1+1
60DC                ;
60DC                ;NOW HAVE DELTX,DELTY (SIGNED)
60DC                ;
60DC                ;** MOVE **
60DC                ;
60DC                ;GIVEN DELTX, DELTY
60DC                ;DRAW/MOVE THE BEST STRAIGHT LINE
60DC                ;
60DC   A9 00        MOVE     LDA #0          ;DETERMINE OCTANT
60DE   8D 4F 03              STA I
60E1   2C 46 03              BIT DELTX+1     ;CHECK DELTX < 0
60E4   10 17                 BPL MV1
60E6   AD 45 03              LDA DELTX       ;CHANGE SIGN
60E9   20 E7 61              JSR COMPL
60EC   8D 45 03              STA DELTX
60EF   AD 46 03              LDA DELTX+1
60F2   20 E8 61              JSR COMPH
60F5   8D 46 03              STA DELTX+1
60F8   A9 02                 LDA #2
60FA   8D 4F 03              STA I
60FD   2C 48 03     MV1      BIT DELTY+1     ;CHECK DELTY < 0
6100   10 1B                 BPL MV2
6102   AD 47 03              LDA DELTY
6105   20 E7 61              JSR COMPL
6108   8D 47 03              STA DELTY
610B   AD 48 03              LDA DELTY+1
610E   20 E8 61              JSR COMPH
6111   8D 48 03              STA DELTY+1
6114   18                    CLC
6115   AD 4F 03              LDA I
6118   69 04                 ADC #4
611A   8D 4F 03              STA I
611D   AE 45 03     MV2      LDX DELTX       ;CHECK DELTX-DELTY
6120   EC 47 03              CPX DELTY       ;SET CARRY FOR LOW BYTE
6123   AD 46 03              LDA DELTX+1     ;NOW HIGH BYTE
6126   A8                    TAY             ;SET Y = DELTX
6127   ED 48 03              SBC DELTY+1
612A   10 1B                 BPL MV3
612C   AD 47 03              LDA DELTY       ;INTERCHANGE DELTX,Y
```

```
LOC     CODE          LINE

612F    8D 45 03             STA DELTX
6132    AD 48 03             LDA DELTY+1
6135    8D 46 03             STA DELTX+1
6138    8E 47 03             STX DELTY
613B    8C 48 03             STY DELTY+1
613E    18                   CLC
613F    AD 4F 03             LDA I
6142    69 08                ADC #8
6144    8D 4F 03             STA I
6147    AD 45 03      MV3    LDA DELTX       ;FORM E=-DELTX/2
614A    20 E7 61             JSR COMPL
614D    8D 49 03             STA E
6150    AD 46 03             LDA DELTX+1
6153    20 E8 61             JSR COMPH
6156    8D 4A 03             STA E+1
6159    38                   SEC             ;CHECK FOR NEGATIVE
615A    30 01                BMI MV4
615C    18                   CLC
615D    6E 4A 03      MV4    ROR E+1         ;DIVIDE BY 2
6160    6E 49 03             ROR E
6163    A0 00                LDY #0          ;SET Y=0
6165    8C 4D 03             STY C           ;SET COUNTER TO ZERO
6168    8C 4E 03             STY C+1
616B    F0 37                BEQ MV7         ;ABSOLUTE BRANCH
616D                  ;
616D                  ;** MAIN DRAWING LOOP **
616D                  ;
616D    AE 4F 03      MV5    LDX I           ;GET DIRECTION IN X
6170    18                   CLC             ;FORM E=E+DELTY
6171    AD 49 03             LDA E
6174    6D 47 03             ADC DELTY
6177    8D 49 03             STA E           ;FIRST LOW BYTE
617A    AD 4A 03             LDA E+1
617D    6D 48 03             ADC DELTY+1
6180    8D 4A 03             STA E+1
6183    30 14                BMI MV6
6185    38                   SEC             ;FORM E=E-DELTX
6186    AD 49 03             LDA E
6189    ED 45 03             SBC DELTX
618C    8D 49 03             STA E
618F    AD 4A 03             LDA E+1
6192    ED 46 03             SBC DELTX+1
6195    8D 4A 03             STA E+1
6198    E8                   INX             ;X BUMPED UP ONE
6199    20 BA 61      MV6    JSR OUTPLT      ;OUTPUT ONE MOVE
619C    EE 4D 03             INC C           ;BUMP COUNTER UP 1
619F    D0 03                BNE MV7
61A1    EE 4E 03             INC C+1
61A4                  ;
```

```
LOC    CODE       LINE
61A4              ;ENTER HERE ON 1ST PASS
61A4              ;
61A4   B1 FD      MV7    LDA (BYT),Y    ;TURN ON A POINT
61A6   0D 44 03          ORA BITNO
61A9   91 FD             STA (BYT),Y
61AB   AD 4D 03          LDA C          ;DONE WHEN C > = DELTX
61AE   CD 45 03          CMP DELTX
61B1   AD 4E 03          LDA C+1
61B4   ED 46 03          SBC DELTX+1
61B7   90 B4             BCC MV5
61B9   60                RTS            ;DONE
61BA              ;
61BA              ;** OUTPLT **
61BA              ;
61BA              ;OUTPUT AN ELEMENTARY MOVE
61BA              ;
61BA   8A         OUTPLT TXA
61BB   0A                ASL A          ;MULT BY TWO TO GET INDEX
61BC   AA                TAX
61BD   BD 0A 63          LDA MOVTAB+1,X ;GET THE VECTOR
61C0   48                PHA            ;HIGH BYTE ON STACK
61C1   BD 09 63          LDA MOVTAB,X
61C4   48                PHA            ;LOW BYTE ON STACK
61C5   60                RTS            ;DO COMPUTED JUMP
61C6              ;
61C6              ;RETURN VIA RTS TO JSR OUTPLT(1)
61C6              ;
61C6              ;**RNGCHK**
61C6              ;
61C6              ;CHECK X2, Y2 FOR OVERFLOW
61C6              ;RETURN TO CALLING PROGRAM ON OVERFLOW
61C6              ;
61C6   AD 3E 03   RNGCHK LDA X2         ;CHECK X2, LOW
61C9   C9 40             CMP #<XMAX
61CB   AD 3F 03          LDA X2+1       ;CHECK HIGH BYTE
61CE   E9 01             SBC #>XMAX
61D0   B0 0C             BCS RNG2       ;X2 > XMAX, SO ABORT
61D2   AD 42 03   RNG1   LDA Y2         ;CHECK Y2, LOW
61D5   C9 C8             CMP #<YMAX
61D7   AD 43 03          LDA Y2+1       ;CHECK HIGH BYTE
61DA   E9 00             SBC #>YMAX
61DC   90 08             BCC RNG3       ;Y2 < YMAX, SO OK
61DE   20 BD 62   RNG2   JSR HRREST     ;RESTORE NORMAL
61E1   A2 0E             LDX #14        ;ILLEGAL QUANTITY ERROR
61E3   6C 00 03          JMP (ERRVEC)   ;FUNNEL THROUGH ERROR
                                         ROUTINE
61E6   60         RNG3   RTS
61E7              ;
61E7              ;COMPL,H
```

```
LOC    CODE          LINE

61E7                 ;
61E7                 ;FORM COMPLEMENT OF SIGNED NUMBER
61E7                 ;1ST ENTER AT COMPL FOR LOW BYTE
61E7                 ;THEN ENTER AT COMPH FOR HIGH BYTE
61E7                 ;
61E7                 ;ANSWER IN A
61E7                 ;
61E7   38            COMPL  SEC              ;FOR LOW BYTE
61E8   49 FF         COMPH  EOR #$FF         ;COMPLEMENT
61EA   69 00                ADC #0           ;ADD CARRY STATE
61EC   60                   RTS
61ED                 ;
61ED   20 32 62 LL          JSR LEFT         ;GO LEFT AND
                                              FALL THROUGH TO DOWN
61F0   A5 FD        DOWN    LDA BYT
61F2   29 07                AND #7           ;EXAM LOWEST 3 BITS
61F4   49 07                EOR #7           ;FLIP THEM
61F6   F0 08                BEQ DN2          ;ORIGINAL BYTE WAS
                                              XXXX111
61F8   E6 FD                INC BYT          ;ELSE JUST BUMP BY 1
61FA   D0 11                BNE DN3
61FC   E6 FE                INC BYT+1
61FE   D0 0D                BNE DN3          ;BRANCH ALWAYS
6200   18            DN2    CLC              ;ADD 320-7
6201   A5 FD                LDA BYT
6203   69 39                ADC #<313
6205   85 FD                STA BYT
6207   A5 FE                LDA BYT+1
6209   69 01                ADC #>313
620B   85 FE                STA BYT+1
620D   60            DN3    RTS
620E                 ;
620E   20 48 62      UR     JSR RIGHT        ;FIRST RIGHT THEN
                                              FALL THROUGH TO UP
6211   A5 FD         UP     LDA BYT
6213   29 07                AND #7           ;CHECK LOW BITS
6215   D0 0F                BNE UP1          ;IF BYTE WAS NOT XXXXX000
6217   38                   SEC              ;ELSE SUBTRACT 320-7
6218   A5 FD                LDA BYT
621A   E9 39                SBC #<313
621C   85 FD                STA BYT
621E   A5 FE                LDA BYT+1
6220   E9 01                SBC #>313
6222   85 FE                STA BYT+1
6224   D0 08                BNE UP3          ;BRANCH ALWAYS
6226   A5 FD         UP1    LDA BYT          ;DECREMENT BY 1
6228   D0 02                BNE UP2
622A   C6 FE                DEC BYT+1
```

```
LOC    CODE        LINE

622C   C6 FD       UP2     DEC BYT
622E   60          UP3     RTS
622F               ;
622F   20 11 62    UL      JSR UP            ;1ST UP THEN FALL
                                              THROUGH TO LEFT
6232   0E 44 03    LEFT    ASL BITNO         ;GO 1 PIXEL LEFT
6235   90 0D               BCC LF2           ;NO CORRECTION ON
                                              CARRY CLEAR
6237   2E 44 03            ROL BITNO         ;SET BITNO=1 AND
                                              CLEAR CARRY
623A   A5 FD               LDA BYT
623C   E9 07               SBC #7            ;(-8 SINCE CARRY IS CLEAR)
623E   85 FD               STA BYT
6240   B0 02               BCS LF2
6242   C6 FE               DEC BYT+1
6244   60          LF2     RTS
6245               ;
6245   20 F0 61    LR      JSR DOWN          ;1ST DOWN THEN FALL
                                              THROUGH TO RIGHT
6248   4E 44 03    RIGHT   LSR BITNO         ;GO 1 PIXEL RIGHT
624B   90 0D               BCC RGT1
624D   6E 44 03            ROR BITNO         ;SET BITNO=$80 AND
                                              CLEAR CARRY
6250   A5 FD               LDA BYT
6252   69 08               ADC #8            ;ONE CELL RIGHT
6254   85 FD               STA BYT
6256   90 02               BCC RGT1
6258   E6 FE               INC BYT+1
625A   60          RGT1    RTS
625B               ;
625B               ;CLRHR
625B               ;
625B               ;CLEARS EXACTLY 8000 BYTES
625B               ;LEAVES Y=X=0
625B               ;
625B   A9 3F       CLRHR   LDA #>HRLAST
625D   85 FE               STA BYT+1         ;INIT. POINTER TO
                                              LAST PAGE
625F   A9 00               LDA #0
6261   85 FD               STA BYT
6263   A8                  TAY
6264   85 FD               STA BYT
6266   91 FD               STA (BYT),Y       ;THIS ONE DONE
                                              SEPARATELY
6268   A0 3F               LDY #<HRLAST      ;START AT BASE+$1F3F
626A   A2 20               LDX #$20          ;X KEEPS
                                              TRACK OF PAGES
626C   91 FD       CLRHR1  STA (BYT),Y       ;PUT IN 0'S
626E   88                  DEY
```

```
LOC    CODE          LINE

626F   D0 FB                BNE CLRHR1
6271   C6 FE                DEC BYT+1
6273   CA                   DEX
6274   D0 F6                BNE CLRHR1        ;DO 32 PAGES
6276   60                   RTS
6277                 ;
6277                 ;SETCOL
6277                 ;
6277                 ;SET FOREGROUND/BACKGROUND COLOR
6277                 ;
6277   A9 50         SETCOL LDA #COLOR        ;IN 2 NYBBLES
6279   A2 00         SETCL0 LDX #0
627B   9D 00 04      SETCL1 STA SCREEN,X      ;DO 4 PAGES
627E   9D 00 05             STA SCREEN+$0100,X
6281   9D 00 06             STA SCREEN+$0200,X
6284   E8                   INX
6285   D0 F4                BNE SETCL1
6287   A2 E8                LDX #<SCREND+1    ;DO LAST PAGE
6289   9D FF 06      SETCL2 STA SCREEN+$02FF,X
628C   CA                   DEX
628D   D0 FA                BNE SETCL2
628F   60                   RTS
6290                 ;
6290                 ;HRINIT - SETS UP HI-RES
6290                 ;
6290   AD 11 D0      HRINIT LDA HRCTRL        ;HI-RES MODE
6293   09 20                ORA #$20          ;TURN ON BIT 5
6295   8D 11 D0             STA HRCTRL
6298   AD 18 D0             LDA HRMREG        ;BYT AT $2000
629B   09 08                ORA #$08          ;TURN ON BIT 3
629D   8D 18 D0             STA HRMREG
62A0   20 77 62             JSR SETCOL        ;FORCE BLACK AND GREEN
62A3   20 5B 62             JSR CLRHR         ;FORCE TO ALL ZEROES
62A6   AD 00 03             LDA ERRVEC        ;REMEMBER SYSTEM
                                               ERROR VECTOR
62A9   8D 52 03             STA ERVEC
62AC   AD 01 03             LDA ERRVEC+1
62AF   8D 53 03             STA ERVEC+1
62B2   A9 F9                LDA #<ABRT        ;SET UP NEW
                                               ERROR RECOVERY
62B4   8D 00 03             STA ERRVEC
62B7   A9 62                LDA #>ABRT
62B9   8D 01 03             STA ERRVEC+1
62BC   60                   RTS
62BD                 ;
62BD                 ;HRREST - RESTORE NORMAL MODE
62BD                 ;
62BD   20 5B 62      HRREST JSR CLRHR         ;CLEAR HI-RES
62C0   AD 11 D0             LDA HRCTRL        ;MODE REGISTER
```

```
LOC   CODE       LINE

62C3  29 DF             AND #%11011111   ;TURN OFF BIT 5
62C5  8D 11 D0          STA HRCTRL
62C8  AD 18 D0          LDA HRMREG       ;MEMORY REGISTER
62CB  29 F7             AND #%11110111   ;TURN OFF BIT 3
62CD  8D 18 D0          STA HRMREG
62D0  A9 20             LDA #' '         ;FILL SCREEN
                                          WITH SPACES
62D2  20 79 62          JSR SETCL0
62D5  AD 52 03          LDA ERVEC        ;RESTORE SYSTEM
                                          ERROR VECTOR
62D8  8D 00 03          STA ERRVEC
62DB  AD 53 03          LDA ERVEC+1
62DE  8D 01 03          STA ERRVEC+1
62E1  60                RTS
62E2             ;
62E2             ;GETVAL - GET PARAMETER
62E2             ;
62E2  20 FD AE   GETVAL JSR CHKCOM       ;CHECK FOR COMMA
62E5  20 9E AD          JSR EVAEXP       ;EVALUATE EXPRESSION
62E8  20 AA B1          JSR FLTFIX       ;CONVERT TO INTEGER
                                          IN Y AND A
62EB  60                RTS
62EC             ;
62EC             ;ENTER HERE IF FROM BASIC
62EC             ;
62EC  20 0F 60   SETPIX JSR HRMOVE
62EF             ;
62EF             ;ENTER HERE IF X1,Y1 ALREADY SET
62EF             ;
62EF  A0 00      STPIX0 LDY #0
62F1  B1 FD             LDA (BYT),Y
62F3  0D 44 03          ORA BITNO
62F6  91 FD             STA (BYT),Y
62F8  60                RTS
62F9             ;
62F9             ;ERROR RECOVERY
62F9             ;
62F9  48         ABRT   PHA              ;SAVE REGS
62FA  8A                TXA
62FB  48                PHA
62FC  98                TYA
62FD  48                PHA
62FE  20 BD 62          JSR HRREST       ;RESTORE TO NORMAL
6301  68                PLA              ;RESTORE REGS
6302  A8                TAY
6303  68                PLA
6304  AA                TAX
6305  68                PLA
6306  6C 00 03          JMP (ERRVEC)     ;ERROR MESSAGE
```

```
LINE# LOC    CODE          LINE

00529  6309                ;
00530  6309   47 62        MOVTAB .WORD RIGHT-1
00531  630B   0D 62               .WORD UR-1
00532  630D   31 62               .WORD LEFT-1
00533  630F   2E 62               .WORD UL-1
00534  6311   47 62               .WORD RIGHT-1
00535  6313   44 62               .WORD LR-1
00536  6315   31 62               .WORD LEFT-1
00537  6317   EC 61               .WORD LL-1
00538  6319   10 62               .WORD UP-1
00539  631B   0D 62               .WORD UR-1
00540  631D   10 62               .WORD UP-1
00541  631F   2E 62               .WORD UL-1
00542  6321   EF 61               .WORD DOWN-1
00543  6323   44 62               .WORD LR-1
00544  6325   EF 61               .WORD DOWN-1
00545  6327   EC 61               .WORD LL-1
00546  6329                ;
00547  6329   80           MSKTB  .BYTE $80,$40,$20,$10
00547  632A   40
00547  632B   20
00547  632C   10
00548  632D   08                  .BYTE $08,$04,$02,$01
00548  632E   04
00548  632F   02
00548  6330   01
00549  6331                ;
00550  6331                .END


ERRORS = 00000


SYMBOL TABLE

SYMBOL VALUE
 ABRT     62F9   BASE     2000   BITNO    0344   BYT      00FD
 C        034D   CHKCOM   AEFD   CLRHR    625B   CLRHR1   626C
 COLOR    0050   COLS     0028   COMPH    61E8   COMPL    61E7
 DELTX    0345   DELTY    0347   DN2      6200   DN3      620D
 DOWN     61F0   E        0349   ERROR    A437   ERRVEC   0300
 ERVEC    0352   EVAEXP   AD9E   FLTFIX   B1AA   GETVAL   62E2
 HRADDR   602D   HRCTRL   D011   HRINIT   6290   HRLAST   3F3F
 HRMOVE   600F   HRMREG   D018   HRREST   62BD   I        034F
 JCLR     6006   JDRAW    6009   JINIT    6000   JREST    6003
 JSETPX   600C   LEFT     6232   LF2      6244   LL       61ED
 LR       6245   MOVE     60DC   MOVTAB   6309   MSKTB    6329
 MV1      60FD   MV2      611D   MV3      6147   MV4      615D
 MV5      616D   MV6      6199   MV7      61A4   ORIGIN   6000
```

```
OUTPLT   61BA   RAM      033C   RGT1     625A   RIGHT    6248
RNG1     61D2   RNG2     61DE   RNG3     61E6   RNGCHK   61C5
SCREEN   0400   SCREND   07E7   SETCL0   6279   SETCL1   627B
SETCL2   6289   SETCOL   6277   SETPIX   62EC   STPIX0   62EF
T        034B   TEMP     0350   UL       622F   UP       6211
UP1      6226   UP2      622C   UP3      622E   UR       620E
VECPL1   609B   VECPLT   6089   VIC      D000   WARMV    0302
X1       033C   X2       033E   XMAX     0140   Y1       0340
Y2       0342   YMAX     00C8
```

Listing #2: BASIC Loader

```
1000 AD=6*16↑3:Z=0:W=1:T=2:C(0)=W:C(1)=16:FS=47:FE=58:F8=48:SF=64
     :FF=55
1010 CT=0:CH=0:E=0:PRINT"WORKING"
1020 FOR I=0 TO 5:CT=CT+CH:CH=0:FOR J=0 TO 127
1030 READ A$:GOSUB2000:POKEAD,D:AD=AD+1:CH=CH+D
1040 PRINT".";:NEXTJ
1050 READ N:PRINT:PRINT"CHECKSUM"I"IS";CH;",SHOULD BE";N
1060 IF N<>CH THEN E=1
1070 NEXT I
1080 CT=CT+CH:CH=0:FOR I=0 TO 48
1090 READ A$:GOSUB2000:POKEAD,D:AD=AD+1:CH=CH+D
1100 PRINT".";:NEXTI
1110 READ N:PRINT:PRINT"CHECKSUM 6 IS";CH;",SHOULD BE";N
1120 IF N<>CH THEN E=1
1130 PRINT:CT=CT+CH:READ N:IF CT=N AND E=0 THEN PRINT"HRSUPP NOW
     LOADED":END
1140 PRINT"CHECKSUM ERROR":END
1150 :
2000 D=Z:FORL=ZTOW:B$=MID$(A$,T-L,W):B=ASC(B$)
2010 IFB>FSANDB<FETHENB=B-F8
2020 IFB>SFTHENB=B-FF
2030 D=D+B*C(L):NEXTL:RETURN
2040 :
6000 DATA 4C,90,62,4C,BD,62,4C,5B
6010 DATA 62,4C,89,60,4C,EC,62,20
6020 DATA E2,62,8C,3C,03,8C,3E,03
6030 DATA 8D,3D,03,8D,3F,03,20,E2
6040 DATA 62,8C,40,03,8C,42,03,8D
6050 DATA 43,03,20,C6,61,A9,00,85
6060 DATA FE,38,A9,C7,ED,40,03,48
6070 DATA 29,F8,0A,26,FE,0A,26,FE
6080 DATA 0A,26,FE,48,8D,50,03,A5
6090 DATA FE,8D,51,03,68,0A,26,FE
6100 DATA 0A,26,FE,6D,50,03,85,FD
6110 DATA A5,FE,6D,51,03,85,FE,AD
6120 DATA 3C,03,29,F8,65,FD,85,FD
6130 DATA AD,3D,03,65,FE,85,FE,68
6140 DATA 29,07,65,FD,85,FD,A5,FE
6150 DATA 69,20,85,FE,AD,3C,03,29
6160 :
6170 DATA 14283:REM CHECKSUM 0
6180 :
6190 DATA 07,AA,BD,29,63,8D,44,03
6200 DATA 60,20,E2,62,8C,3E,03,8D
6210 DATA 3F,03,20,E2,62,8C,42,03
6220 DATA 8D,43,03,20,C6,61,38,AD
6230 DATA 3E,03,ED,3C,03,8D,45,03
6240 DATA AD,3F,03,ED,3D,03,8D,46
6250 DATA 03,38,AD,42,03,ED,40,03
```

```
6260 DATA 8D,47,03,AD,43,03,ED,41
6270 DATA 03,8D,48,03,AD,3E,03,8D
6280 DATA 3C,03,AD,3F,03,8D,3D,03
6290 DATA AD,42,03,8D,40,03,AD,43
6300 DATA 03,8D,41,03,A9,00,8D,4F
6310 DATA 03,2C,46,03,10,17,AD,45
6320 DATA 03,20,E7,61,8D,45,03,AD
6330 DATA 46,03,20,E8,61,8D,46,03
6340 DATA A9,02,8D,4F,03,2C,48,03
6350 :
6360 DATA 10315:REM CHECKSUM 1
6370 :
6380 DATA 10,1B,AD,47,03,20,E7,61
6390 DATA 8D,47,03,AD,48,03,20,E8
6400 DATA 61,8D,48,03,18,AD,4F,03
6410 DATA 69,04,8D,4F,03,AE,45,03
6420 DATA EC,47,03,AD,46,03,A8,ED
6430 DATA 48,03,10,1B,AD,47,03,8D
6440 DATA 45,03,AD,48,03,8D,46,03
6450 DATA 8E,47,03,8C,48,03,18,AD
6460 DATA 4F,03,69,08,8D,4F,03,AD
6470 DATA 45,03,20,E7,61,8D,49,03
6480 DATA AD,46,03,20,E8,61,8D,4A
6490 DATA 03,38,30,01,18,6E,4A,03
6500 DATA 6E,49,03,A0,00,8C,4D,03
6510 DATA 8C,4E,03,F0,37,AE,4F,03
6520 DATA 18,AD,49,03,6D,47,03,8D
6530 DATA 49,03,AD,4A,03,6D,48,03
6540 :
6550 DATA 10002:REM CHECKSUM 2
6560 :
6570 DATA 8D,4A,03,30,14,38,AD,49
6580 DATA 03,ED,45,03,8D,49,03,AD
6590 DATA 4A,03,ED,46,03,8D,4A,03
6600 DATA E8,20,BA,61,EE,4D,03,D0
6610 DATA 03,EE,4E,03,B1,FD,0D,44
6620 DATA 03,91,FD,AD,4D,03,CD,45
6630 DATA 03,AD,4E,03,ED,46,03,90
6640 DATA B4,60,8A,0A,AA,BD,0A,63
6650 DATA 48,BD,09,63,48,60,AD,3E
6660 DATA 03,C9,40,AD,3F,03,E9,01
6670 DATA B0,0C,AD,42,03,C9,C8,AD
6680 DATA 43,03,E9,00,90,08,20,BD
6690 DATA 62,A2,0E,6C,00,03,60,38
6700 DATA 49,FF,69,00,60,20,32,62
6710 DATA A5,FD,29,07,49,07,F0,08
6720 DATA E6,FD,D0,11,E6,FE,D0,0D
6730 :
6740 DATA 12978:REM CHECKSUM 3
6750 :
6760 DATA 18,A5,FD,69,39,85,FD,A5
6770 DATA FE,69,01,85,FE,60,20,48
6780 DATA 62,A5,FD,29,07,D0,0F,38
6790 DATA A5,FD,E9,39,85,FD,A5,FE
6800 DATA E9,01,85,FE,D0,08,A5,FD
6810 DATA D0,02,C6,FE,C6,FD,60,20
6820 DATA 11,62,0E,44,03,90,0D,2E
6830 DATA 44,03,A5,FD,E9,07,85,FD
6840 DATA B0,02,C6,FE,60,20,F0,61
6850 DATA 4E,44,03,90,0D,6E,44,03
6860 DATA A5,FD,69,08,85,FD,90,02
6870 DATA E6,FE,60,A9,3F,85,FE,A9
6880 DATA 00,85,FD,A8,85,FD,91,FD
6890 DATA A0,3F,A2,20,91,FD,88,D0
6900 DATA FB,C6,FE,CA,D0,F6,60,A9
6910 DATA 50,A2,00,9D,00,04,9D,00
6920 :
6930 DATA 17166:REM CHECKSUM 4
6940 :
6950 DATA 05,9D,00,06,E8,D0,F4,A2
6960 DATA E8,9D,FF,06,CA,D0,FA,60
6970 DATA AD,11,D0,09,20,8D,11,D0
6980 DATA AD,18,D0,09,08,8D,18,D0
6990 DATA 20,77,62,20,5B,62,AD,00
7000 DATA 03,8D,52,03,AD,01,03,8D
7010 DATA 53,03,A9,F9,8D,00,03,A9
7020 DATA 62,8D,01,03,60,20,5B,62
7030 DATA AD,11,D0,29,DF,8D,11,D0
7040 DATA AD,18,D0,29,F7,8D,18,D0
7050 DATA A9,20,20,79,62,AD,52,03
7060 DATA 8D,00,03,AD,53,03,8D,01
7070 DATA 03,60,20,FD,AE,20,9E,AD
7080 DATA 20,AA,B1,60,20,0F,60,A0
7090 DATA 00,B1,FD,0D,44,03,91,FD
7100 DATA 60,48,8A,48,98,48,20,BD
7110 :
7120 DATA 13370:REM CHECKSUM 5
7130 :
7140 DATA 62,68,A8,68,AA,68,6C,00
7150 DATA 03,47,62,0D,62,31,62,2E
7160 DATA 62,47,62,44,62,31,62,EC
7170 DATA 61,10,62,0D,62,10,62,2E
7180 DATA 62,EF,61,44,62,EF,61,EC
7190 DATA 61,80,40,20,10,08,04,02
7200 DATA 01
7210 :
7220 DATA 4154:REM CHECKSUM 6
7230 :
7240 DATA 82268:REM TOTAL CHECKSUM
```

Listing #3: Demonstration Program

```
10 IF A=0 THEN A=1:LOAD"HRSUPP"
   ,8,1
20 BA=6*16↑3:REM BASE ADDRESS
30 IN=BA
40 RS=BA+3
50 CL=BA+6
60 DR=BA+9
70 PX=BA+12
80 MV=BA+15
90 SYS(IN)
100 S=3:SYS(MV),S,S:FOR I=S TO
    195 STEP S
110 X1=S:Y1=X1:X2=X1:Y2=Y1+I
120 X3=X2+I:Y3=Y2:X4=X3:Y4=Y3-I
130 SYSDR,X2,198
140 SYSDR,X3,Y3
150 SYSDR,X4,Y4
160 SYSDR,X1,Y1
170 NEXT I
180 GET A$:IF A$<>"C" THEN 180
200 R=80:XC=160:YC=100:A=π/180:
    S=5
210 SYS(CL)
220 FOR AN = 0 TO π/1.99 STEP
    π/20
230 SYSMV,XC+R*SIN(AN),YC+R*S
    IN(AN)
240 FOR I=S TO 360 STEP S
250 SYSDR,XC+R*SIN(2*I*A+AN),
    YC+R*SIN(I*A+AN)
260 NEXT I,AN
270 GET A$:IF A$<>"C" THEN 270
300 SYS(CL)
310 D=4:E=2:X=XC:Y=YC
320 SYSMV,X,Y
330 FOR I=0 TO 20
340 D=D+E:Y=Y+D:SYSDR,X,Y
350 D=D+E:X=X+D:SYSDR,X,Y
360 D=D+E:Y=Y-D:SYSDR,X,Y
370 D=D+E:X=X-D:SYSDR,X,Y
380 NEXT I
390 GET A$:IF A$<>"C" THEN 390
400 SYSCL:S=π/3
410 FOR T=0 TO S STEP S/8
420 SYSMV,XC+R*COS(T),YC+R*SIN(T)
430 FOR I=S TO 2*π STEP S
440 SYSDR,XC+R*COS(I+T),YC+R*
    SIN(I+T)
450 NEXT I,T
460 GET A$:IF A$<>"C" THEN 460
500 SYSCL:S=π/4:D=R/20
510 FOR T=0 TO S STEP S/20
520 SYSMV,XC+R*COS(T),YC+R*SIN(T)
530 FOR I=S TO 2*π STEP S
540 SYSDR,XC+R*COS(I+T),YC+R*SI
    N(I+T)
550 NEXT I
560 R=R-D:NEXT T
580 GET A$:IF A$<>"C" THEN 580
600 SYSCL:R=80:S=π/8:D=R/20
610 FOR T=0 TO S STEP S/40
620 SYSPX,XC+R*COS(T),YC+R*SIN(T)
630 FOR I=S TO 2*π STEP S
640 SYSPX,XC+R*COS(I+T),YC+R*SI
    N(I+T)
650 NEXT I
660 R=R-D:NEXT T
680 GET A$:IF A$<>"C" THEN 680
700 SYSCL:R=80:S=2*π/5:A=π/10
710 FOR I=0 TO 4
720 T=A+I*S
730 X(I)=XC+R*COS(T):Y(I)=YC+R
    *SIN(T)
740 NEXT I
750 SYSMV,X(0),Y(0)
760 SYSDR,X(2),Y(2):SYSDR,X(4),
    Y(4)
770 SYSDR,X(1),Y(1):SYSDR,X(3),
    Y(3)
780 SYSDR,X(0),Y(0)
790 GET A$:IF A$<>"C" THEN 760
800 SYSCL:A=160:B=A/2:SYSMV,0,A
    *EXP(-4)
810 FOR X=4 TO 2*A-1 STEP 4
820 SYSDR,X,A*EXP(-((X-A)/B)↑2)
830 NEXT X
880 GET A$:IF A$<>"C" THEN 880
9999 SYS(RS)
```

A Graphics Language for the 64

The graphics program presented in this article is actually a graphics language. The demonstration program (listing #3) is one example of how to use this graphic language. There are seven commands that can be used with this language. They are Initialize, Reset, Clear, Pixel, Move, Draw and Color. Lines 20-80 of the demonstration program set the SYS values for each of these commands (except Color which would be BA+631). The following list explains each command and gives an example of its use.

NOTE: The following examples assume that lines 20-80 of the demonstration program have been used to set up your program.

Initialize—This command initializes the graphic language. This must be used before any other commands can be used.

Syntax—SYS(IN)

Reset—This command turns off the graphics language and will return the program to BASIC. This should be used at the end of your program to return the cursor and READY prompt.

Syntax—SYS(RS)

Clear—This command will clear the high resolution screen. The color displayed is the background color (see the Color command).

Syntax—SYS(CL)

Pixel—This command will turn on one point (or pixel) at the specified X and Y coordinate.

Syntax—SYS(PX),X,Y
Example—SYS(PX),50,120 would turn on the pixel 50 places to the right of and 120 places above the lower lefthand corner of the screen.

Move—This command moves the pixel pointer to the specified X and Y location. No pixels are turned on. This command is used to set the first X,Y point of a line to be drawn.

Syntax—SYS(MV),X,Y
Example—SYS(MV),50,120 would put the pixel pointer at the same location as in the Pixel example, however the pixel would not be turned on.

Draw—This command will draw a line between the current pixel pointer (set by either a Pixel or Move command) to the specified X and Y coordinates.

Syntax—SYS(DR),X,Y
Example—SYS(DR),100,150 would draw a line from the current pixel pointer position (X=50, Y=120 if the Move command example was used) to X=100, Y=150.

Color—This command will change the background and pixel colors displayed on the screen. The color number associated with this command is formed by an upper nibble for the pixel color and a lower nibble for the background color. Page 61 of the Commodore 64 user's guide has a chart with the number value for each color. The color number is defined as 16*(the pixel color #)+(the background color #).

Syntax—POKE(CR+1),(color number):SYS(CR)
Example—POKE(CR+1),33:SYS(CR) would set the pixel color as red (16*2=32) and the background color as white (33−32=1). **C**

Jim Gracely

Stop Gambling. Start Winning. Now.

It's a fact. You will beat the dealer if you play Blackjack correctly. In Las Vegas. In Atlantic City. In dozens of foreign countries throughout the world.

They haven't changed the rules. Even multiple-deck games pose no problem if you play properly. You can win just as easily in 1983 as you could in 1961 when the first Blackjack strategies were created.

This ad is your cue to join the small group of Blackjack players who are no longer gambling. Become a strategy player and win. Consistently.

The Obstacle

Despite the wild claims made by the Blackjack system charlatans, it is not possible to learn an effective strategy overnight. Learning an effective strategy takes time and discipline. If learning a strategy were easy, everyone would be making a living playing Blackjack. As it stands, less than one percent play well enough to make money.

The Solution

BLACKJACK TEACHER simulates, in precise detail, the events that transpire in actual casino play. The display screen depicts the top view of a Blackjack table. You interact with the program just as you would an actual game. Computer controlled players occupy adjacent seats. All events occur in real-time.

BLACKJACK TEACHER teaches seven different strategies of varying complexity and accuracy. This spectrum of strategies allows you to select a strategy that suits your needs.

BLACKJACK TEACHER monitors your betting and strategy decisions (hit/stand/double/split/insurance). If your decisions are incorrect within the guidelines of your strategy, the system will display error messages showing you the correct decisions.

BLACKJACK TEACHER is the result of over ten years of Blackjack research. The strategies encompassed by the system were developed using computers. The more complex strategies are among the most powerful ever devised.

Complete documentation is included which tells you everything you need to know to become an expert strategy player.

The SOTA Story

SOTA Enterprises has consistently produced nothing less than the highest quality software. When you buy software from SOTA, we do our utmost to make sure you get your money's worth.

ATTENTION VIC 20 USERS

A new version of BLACKJACK TEACHER is now available for the VIC 20. Although not as comprehensive as the original 32K program, the VIC 20 version does teach Basic Strategy – a must for the Blackjack strategy beginner!

FILL OUT AND MAIL TODAY!

Name ______________________

Address ______________________

City ______________________

State ____________ **Zip** ____________

Make Check or Money Order Payable to:

SOTA Enterprises, Inc.
833 Garfield Ave, Suite 101
South Pasadena, CA 91030

Check Box

☐ VIC 20	($19.95)
☐ COMMODORE 64	($49.95)
☐ PET (32K)	($49.95)
☐ PET 2001 (32K)	($49.95)
☐ CBM 4032	($49.95)
☐ CBM 8032 (w/screen cruncher)	($49.95)
☐ SUPER PET	($49.95)

Include $2.50 Postage and Handling • California Residents add 6½% Sales Tax

Color Me Purple... Or Red... Or Green...

by Doris Dickenson

Some activities to teach children to manipulate color on the Commodore 64, from a fourth-grade teacher who won her 64 in an essay contest—and then had to figure out how to use it. Doris' articles have appeared in several past issues of Commodore.

When we replaced the black and white T.V. monitor for our Commodore 64 with a new color monitor, we opened up a whole new area of exploration for my fourth-grade students. We added some language arts activity to the color capabilities of our computer, and also some practice in programming. Since many children of 9 or 10 seem to be interested in the visual aspects of computers, rather than the mechanics of programming on their own, the replacement created a great deal of renewed interest among the students.

Working independently with the classroom manuals that I created for them in our Computer Corner, the students were soon involved in drawing reverse color bars with the color keys. It wasn't long before they began creating their own color patterns. There were almost as many different combinations of designs and colors as there were students using the computer. (Editor's Note: Doris' instruction manual for children, "You and Your Computer", appeared in five parts in the last three issues of *Commodore.*)

As an introduction to using computer commands to control colors, I put up a chart showing the POKE code and number listings for different available border and background colors. (See the Commodore 64 users' manual, page 61.) When you do this, list color 3 (cyan) as light green-blue. It is more understandable to the students.

Activity 1: Type POKE 53280, ___;
POKE 53281, ___
RETURN

Use any numbers from the chart in place of the dashes. Once the color is changed, use cursor up and cursor right to replace the color numbers with other color numbers. You can come up with all sorts of interesting combinations, but watch out when you change the background to 14, light blue, which is the normal printing color. Your printing will seem to disappear unless you change the color of the printing with a color key before you put in the light blue background.

Activity 2: Using some of the language arts ideas from our reading, we selected some figures of speech that contained color words, then chose some colors to suit the single expressions. A little simple programming combined these into one program. (See program that follows for Activity 2.)

Activity 3: How many more "color expressions" can you find? Add these into the program.

Activity 4: Do some research into song titles with color words. You might want to start with "Red River Valley", "Greensleeves" or "Blue-Tail Fly". Use the program in the previous activity to help you make up your own program of song titles.

Activity 5: This short and simple program puts a familiar poem into color. (See program that follows for Activity 5.)

Activity 6: Try typing PRINT CHR$(20) for lines 10, 50, 110, 160, 210, and 270.

Activity 7: Change the border and screen colors in lines 60, 100, 150, and 200 by using different POKE color numbers. (See chart or users' manual, page 61)

Activity 8: Think of other ways you can put words and color together. Try to write them into a program. **C**

Program for Activity 2

```
5 REM COLOR WORDS
10 PRINTCHR$(147)
20 PRINTCHR$(5)
30 POKE53280,0:POKE53281,0
40 PRINT"BLACK AS PITCH"
50 FORX=1TO1000:NEXT
60 PRINTCHR$(147)
70 PRINTCHR$(144)
80 POKE53280,2:POKE53281,2
90 PRINT"RED AS A BEET"
100 FORX=1TO1000:NEXT
110 PRINTCHR$(147)
120 POKE53280,4:POKE53281,4
130 PRINT"PURPLE WITH RAGE"
140 FORX=1TO1000:NEXT
150 PRINTCHR$(147)
160 POKE53280,14:POKE53281,14
170 PRINT"FEELING BLUE"
180 FORX=1TO1000:NEXT
190 PRINTCHR$(147)
200 POKE53280,5:POKE53281,5
210 PRINT"GREEN WITH ENVY"
220 FORX=1TO1000:NEXT
230 PRINTCHR$(147)
240 POKE53280,1:POKE53281,1
250 PRINT"WHITE AS A SHEET"
260 FORX=1TO1000:NEXT
270 PRINTCHR$(147)
280 PRINTCHR$(154)
290 POKE53280,14:POKE53281,6
```

This clears the screen. (line 10)

This timing loop keeps it on the screen long enough to read it (line 50)

Program for Activity 5

```
5 REM COLOR POEM
10 PRINTCHR$(147)
15 PRINTCHR$(144)
20 POKE53280,1:POKE53281,1
35 PRINT:PRINT:PRINT:PRINT:PRINT
40 PRINTTAB(10)"AN OLD FAVORITE"
45 GOSUB500
50 PRINTCHR$(147)
60 POKE53280,4:POKE53281,2
70 PRINT:PRINT:PRINT:PRINT:PRINT
80 PRINTTAB(10)"ROSES ARE RED"
90 GOSUB500
100 POKE53280,4:POKE53281,6
110 PRINTCHR$(147)
120 PRINT:PRINT:PRINT:PRINT:PRINT
130 PRINTTAB(10)"VIOLETS ARE BLUE"
140 GOSUB500
150 POKE53280,8:POKE53281,7
160 PRINTCHR$(147)
```

Type PRINT "{CLR HOME}" by pressing SHIFT and CLR HOME. It will print as ♥ (line 10)

This moves your printing down the screen (line 35)

The TAB moves your printing in from the edge. (line 40)

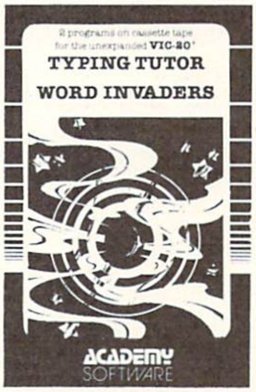
Instead of repeating the timing loop, the computer can go down to one *subroutine* when you want a delay. (line 45)

```
170 PRINT:PRINT:PRINT:PRINT:PRINT
180 PRINTTAB(10)"SUGAR IS SWEET"
190 GOSUB500
200 POKE53280,5:POKE53281,13
210 PRINTCHR$(147)
220 PRINT:PRINT:PRINT:PRINT:PRINT
230 PRINTTAB(10)"AND SO ARE YOU"
240 GOSUB500
260 POKE53280,14:POKE53281,6
270 PRINTCHR$(147)
280 PRINTCHR$(154)
290 END
500 FORT=1TO1000      <-- Subroutine timing loop
510 NEXTT
520 RETURN            <-- Go back to the regular program.
```

Press RUN/STOP and RESTORE to get back to a normal printing color mode.

FOR THE VIC 20™

JOIN THE COMPUTER REVOLUTION WITH A MASTERY OF THE KEYBOARD!



In the age of the computer, *everyone* from the school child to the Chairman of the Board should be at home at the computer keyboard. Soon there will be a computer terminal on every desk and in every home. Learn how to use it right ...and have some fun at the same time!

Rated THE BEST educational program for the VIC 20™ by Creative Computing Magazine

TYPING TUTOR PLUS WORD INVADERS — $21.95

(2 programs on one cassette tape for the unexpanded VIC 20™)

Typing Tutor plus Word Invaders makes learning the keyboard easy and fun! Typing Tutor teaches the keyboard in easy steps. Word Invaders makes typing practice an entertaining game. Highly praised by customers:

"Typing Tutor is great!", "Fantastic", "Excellent", High quality", "A source of great joy and learning for our children", "Even my little sister likes it", "Word Invaders is sensational!"

Customer comment says it all . . .

". . . and it was everything you advertised it would be. In three weeks, my 13 year old son, who had never typed before, was typing 35 w.p.m. I had improved my typing speed 15 w.p.m. and my husband was able to keep up with his college typing class by practicing at home."

FOR THE COMMODORE 64™

SPECIAL VERSION OF TYPING TUTOR PLUS WORD INVADERS NOW AVAILABLE FOR THE COMMODORE 64™... $21.95 *(Tape)* $24.95 *(Disk)*

All of the features of the VIC 20™ Version and more

SPECIAL!

SPRITE DESIGNER

by Dr. Lee T. Hill

$16.95 *(Tape)* $21.95 *(Disk)*

Create and then transform sprites automatically. We have the other sprite making programs, but this is the one *we use* to make sprites. The automatic transformations are great!

VISA — Shipping and handling $1.00 per order. California residents add 6% sales tax. VISA and Mastercard orders must include full name as shown on card, card number, and expiration date. Free catalog sent with order and on request. — MasterCard

ACADEMY SOFTWARE

P.O. Box 9403, San Rafael, CA 94912 (415) 499-0850

Programmers: Write to our New Program Manager concerning any exceptional VIC 20TM or Commodore 64TM game or other program you have developed.

Computer Programs Teach Fifth Graders Elementary Economics

by Larry Modrell

A VIC 20 does "payroll" and runs the "bank" in this Oregon classroom, where fifth graders get direct experience in economic realities.

Barbara Kroeker and I teach fifth grade students at the Elizabeth Page Elementary School in Springfield, Oregon. Over the past few years, we have devised an "economics" program to motivate students to excell in their academics, which we initiate during the last three months of each school year. The entire system is based on the conversion of the students' grades into a monetary value and the use of the resulting "paycheck" as it would be used in the real world.

On Thursday afternoons, the VIC 20 converts each student's grades for the week into a monetary value, and calculates how much tax the student must pay. We have written a program for the VIC to handle this chore, and it is amazing how much time the computer has saved us.

A payroll slip is then made out for each student. The students must then deduct their income tax, utility bills, rent on their desks and any fines that may have been levied over the past week. They use their basic math skills to accomplish this, and must be accurate in adding all these items together and subtracting the total from their gross earnings to get their net earnings.

When the students have completed this task, they go to one of the two computer operators in each of our classrooms to verify the accuracy of their net earnings. We have written another program to do this job. (Before we had the computers, we did this entirely with calculators, which was very time consuming.) After they check them, the computer operators sign the payroll slips they find to be correct.

Because the job of computer operator is a fairly responsible position, the operators had to write letters of application and interview for the jobs. They run the programs on our Commodore 64's, which have replaced the PET and VIC 20 that used to do these calculations.

After their figures are verified by the computer operator, students take their payroll slips to the bank (also run by students), and cash them in for "money"—actually play money printed with students' pictures on each denomination. If they wish they can put money into a savings account and earn ten percent interest per week. Our VIC 20's are tapped again to handle this task, managing the entire savings department at the bank, including calculating the interest earned on each account.

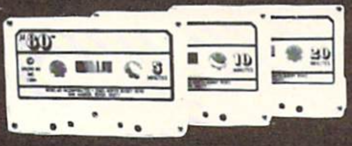
Students who wish to deposit money are given an account number and all transactions from that time are stored on tape and updated by our VIC computers. If a student deposits or withdraws money, the computer automatically adds or subtracts the amount and gives an instant printout of the new balance on the screen. Students can also print a hard copy of all updated accounts on our VIC printer.

If they do not wish to save all their "money" the students can also choose to spend it in our weekly stores. Students operate the stores and learn to make correct change when items are purchased. We have a toy store, a candy store, book store, car lot (Hot Wheels), bakery and others. I also get the opportunity to play auctioneer once a week and auction off items to the highest bidder.

The computers have enhanced the success of our economics program and have added a new dimension to our economics simulation that helps motivate and educate the children. It's true that parents are enthused and supportive, and the P.T.A. donates items to be sold in our stores. But, most of all, the students are learning that the computer can be used as a basic tool, for much more than the usual drill-and-practice routines they are generally exposed to in school. **C**

COMPUTER CASSETTES

100% Error-Free • Fully Guaranteed



LENGTH	12 PACK	24 PACK
C-05	79¢	69¢
C-10	89¢	79¢
C-20	99¢	89¢
Boxes	26¢	21¢

UPS $3.00 Pkg. $18.00 Case

C-10's 39¢
(Min. 500 Case Lot)
w/labels ADD 4¢,
w/boxes ADD 13¢

FOR ORDERS ONLY
1-800-528-6050
Extension 3005

MICRO-80™ INC.
2665-P Busby Road
Oak Harbor, WA 98277

A Giant Step for the computerist

THE PROMQUEEN

Opens up the world of modern electronics. Now – a complete microdevelopment system in a cartridge using the Commodore VIC-20. You get HEXKIT 1.Ø for general purpose 8 bit microprocessor software development, a 4K ROM emulator for testing program in circuits under development plus an EPROM programmer for making hard copy of programs. All-in-one cartridge with 100 page tutorial manual. **$199.00**

Arbutus Total Soft, Inc., 4202 Meridian, Suite 214, Bellingham, WA 98226. Phone 800-426-1253, in Washington 206-733-0404
Distributed in Canada by **IBC/Distribution Canada**, 4047 Cambie St. Vancouver, BC V5Z 2X9. Phone 604-879-7812

DEVELOP-20

FIVE POWERFUL SOFTWARE DEVELOPMENT TOOLS

Plus The Exciting New Book

INSIDE THE VIC

THE BOOK

A complete clear explanation of machine language, Assembly language, VIC 20 architecture, graphics, joystick and sound effect programming. Detailed step-by-step guide to the use of the development tools. How to combine BASIC and machine language, make auto-start cartridges, interface with the internal ROM-based programs of BASIC and the Kernal. Sample programs fully explained.

THE TOOLS

Assembler/Editor/Loader/Decoder/Monitor
Full-featured Assembler allows use of labels, comments and arithmetic expressions to create machine language programs. Create, save, modify Assembly language programs with the Editor. Load and link machine language modules with the Loader. Decode machine language back into assembly language for study or input to the Editor. Single-step program execution with the Monitor. Extended features combines Assembler/Editor for maximum ease of use.

ALL FOR $49.95 PLUS $2.00 POSTAGE AND HANDLING Standard version runs on any system with Datasette (5K and up) Add $5.00 for disk version, $5.00 for extended features (minimum 8K) Send check, M.O., VISA/MC ($2.00 S.C.) or specify C.O.D. (add $3.00) to:

French Silk smooth ware

P.O. Box 207, Cannon Falls, MN 55009

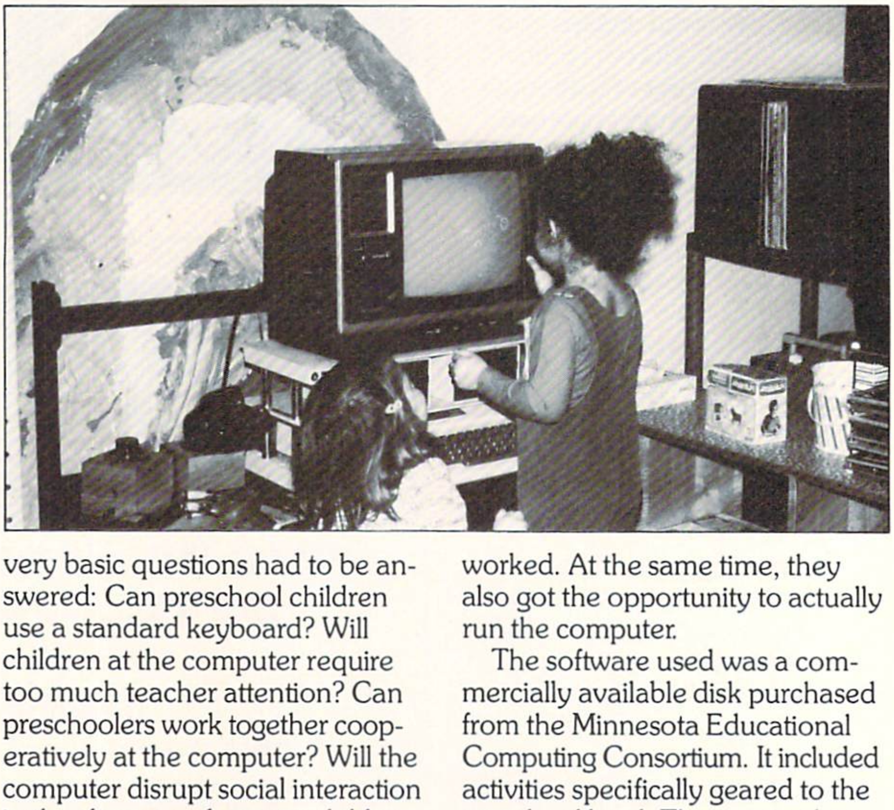
507-263-4821

VIC-20 is a registered TM of Commodore Business Machines Inc.

Preschoolers at the Computer

by Alexandra Muller
Postdoctoral Associate • Institute of Child Development • University of Minnesota

Educators used to feel that young people needed extensive training in mathematics and logic in order to properly use and benefit from computers. They felt it was useless to introduce computers before college. Nevertheless, computing began to be introduced in high school, and finally in elementary school. What about preschool? Can and should children in the preschool years be exposed to computers?



Given the increasing prevalence of computers, it is essential that the age at which children can begin to profit from interaction with this technology not be underestimated. The issue of how young children can or should be formally exposed to computers is important, because it is likely that those who are exposed to computers early will be more comfortable and facile with them later. Therefore, a research project was initiated under the auspices of the Institute of Child Development at the University of Minnesota in order to study preschool children's interactions when using a computer. The purpose of the study was to find out whether preschoolers' intellectual and social development permits meaningful computer use.

In the study, a number of very basic questions had to be answered: Can preschool children use a standard keyboard? Will children at the computer require too much teacher attention? Can preschoolers work together cooperatively at the computer? Will the computer disrupt social interaction in the classroom because children will prefer to play with the computer rather than with each other?

The children studied were a classroom of four- and five-year-olds at the University Child Care Center. They were introduced to the microcomputer in small groups, during a half-hour session in which they received verbal explanations of how the computer worked. At the same time, they also got the opportunity to actually run the computer.

The software used was a commercially available disk purchased from the Minnesota Educational Computing Consortium. It included activities specifically geared to the preschool level. There were three alphabet games, three counting/number games, and three concentration-type matching from memory games using pictures, words, or shapes. In order to choose a program, the child had only to press a number corresponding to a picture which depicted the program they wanted. To respond to a program, a child needed only

The issue of how young children can or should be formally exposed to computers is important, because it is likely that those who are exposed to computers early will be more comfortable and facile with them later.

to press a single letter or number.

The computer was placed in a central location against one wall of the preschool classroom, and turned on with the program directory visible on the screen. It was freely accessible in the classroom during playtime, along with the other activities usually available. We wanted to provide free access so that children would feel the computer was something to be readily approached and used.

The children were allowed to work at the computer in groups of two during their free playtimes. We found that a maximum of two children at a time provided the best opportunity for each child to interact with the computer.

The children also were allowed to decide on their own how long they played with the computer. We had tried to regulate the amount of time they spent at the computer with a timer, but it frequently stopped children in the middle of an activity, which they found very frustrating. When other activities were also available, we found that children stayed at the computer an average of about 20 minutes, which allowed the opportunity for several groups to use the computer during a 90 minute session.

The teachers were asked to interact with the children at the computer to the same extent they would if the children were engaged in the usual classroom activities. Teachers usually let the children play independently, unless their help or company were actively sought or seemed to be needed by a child. The teachers followed this same pattern when children were at the computer. We came in to observe the children interacting at the computer three times per week over a two-month period during the summer.

Initially, we thought that a standard keyboard might be too confusing for the children, that the children might accidentally damage the computer or that the children might be too young to work cooperatively at the computer. As the study progressed, we realized that we had drastically underestimated the children's capabilities on all counts.

What we found was that under these carefully managed circumstances the preschoolers spontaneously shared use of the computer and helped each other with minimal intervention from teachers. They were well able to use the standard keyboard, and had little trouble finding the right key to make the simple single-key responses required. Working as teams the preschoolers would often help each other pick out the correct key by pointing to it or telling the other child where it was. Although they did occasionally ask for a teacher's opinion or help if one was nearby, they usually worked with other children, independently of the teachers.

Interestingly, the children's help to each other was mainly through verbal instructions rather than by pointing or pressing the key for the other child. For example, they would say, "You forgot to press RETURN," or they would say the ABCs to help the other child figure out the letter that was missing in the five-letter sequence on the screen. We and the teachers had imagined that preschoolers would have more trouble explaining things to other children than they did. So preschoolers were able to work cooperatively at the computer, seemingly without requiring more teacher attention than usual.

We were also interested in seeing if children would choose to work alone at the computer or with others. We found that the computer did not seem to disrupt normal social activity in the classroom. Children preferred to work with someone and would often look for another child to work with them at the computer. This did not seem to be because they were intimidated by the computer, but because it was more fun to play with another person than alone. The fact that helping and sharing behaviors were common suggests to us that computers could be a focus for children's social interactions as well as any other enjoyable activity.

Clearly, our findings show that with age-appropriate software even preschoolers are capable of interacting with a computer and working cooperatively with their peers, without the need of constant supervision by teachers. That preschoolers can perform competently at the computer is interesting, but to what purpose does one introduce the computer to children of this or any age?

One reason you might want to introduce a child to computers and computing at an early age is to develop computer literacy. This can consist of at least two levels: computer awareness and a working knowledge of how to use computers to perform certain tasks.

Computer awareness means many things to many people, but in general we can say that it means a familiarity with how computers work, what tasks they can and can't perform, and the contexts in which computers may be found. Computer awareness can be said to be a type of "computer readiness" or stage of preparedness to learn to use computers. For example, even a very young child can be familiarized with the way computers look and operate.

I'm sure everyone is acquainted with at least one adult who is a computerphobe. That is, a person who is afraid to have anything to do with a computer. This fear is simply due to their lack of familiarity with computers. If children are introduced to computers before they have had a chance to develop fear of computers, they will be more likely to be willing to learn to use it for various types of applications. Young children are naturally confident of their abilities and preschool may not be too early to begin to get children comfortable with this important tool, if it can be done in a relaxed and enjoyable way.

Further, since computers can be used to present school material, they can be used to introduce or improve academic skills. There are a number of characteristics that computers have which may make them particularly suited for this function.

First of all, children seem to enjoy working with material presented on the computer more than with material presented in a traditional manner on paper or blackboard. This may in part be due to the novelty of the computer itself, or because it is possible to introduce animation into the programs, which makes them more visually stimulating.

Second, material presented on the computer can be paced by the child more readily than in the traditional classroom setup. And the rewards administered by the computer are likely to be more accurate and timely than those presented by a human teacher with many other children to attend to.

Finally, children may be less exposed to shame and ridicule if they make a mistake, since their mistakes are not publicly exposed, and because the feedback from a computer does not convey the negative emotions that corrections by an individual might.

It has been argued that because of the possibility of interactive feedback, the computer can be an important tool for stimulating problem-solving abilities in children. Seymour Papert and his colleagues at the Massachusetts Institute of Technology have developed a programming language called LOGO, which is designed to be easy for children to learn and to provide optimal opportunity for the stimulation of programming or problem-solving skills. Through use of simple English-like commands, children can almost immediately produce interesting designs, by directing the motion on the screen of a cursor called a "turtle".

This type of interaction with a computer provides the child with a working knowledge of how they can control what the computer does. LOGO is designed to incorporate many of the basic ideas underlying computer programming. Thus, using LOGO can provide an added dimension of involvement with the computer by showing that it is a unique instrument, rather than simply a technologically advanced method for presenting traditional material. Even preschoolers can master the rudiments of LOGO, since it need not require reading ability. LOGO activities can illustrate principles such as the ability to save information, recursion, editing, building a greater whole from component parts and so on. Some have argued LOGO may provide better preparation for learning computer languages than learning one of the existing programming languages, which may soon be out of date.

However, the purpose of LOGO is not to train programmers or to take the place of other programming languages, but to stimulate children's ability to intellectually explore and to provide an enjoyable environment for this exploration. This is the type of intellectual activity that has not had much place in schools until now. Most formal education concerns the learning of specific information and skills such as reading, arithmetic, history etc. On the other hand, LOGO's format seems to stimulate children's curiosity concerning the way computers work, which in small part is curiosity about how the world works.

However, the availability of this powerful tool alone will not necessarily improve the intellectual quality of education for most children. Because, unlike the case in which structured academic material is presented, LOGO requires well-trained and qualified teachers to implement LOGO-learning environments that can do justice to its potential.

It is too early to say with any certainty what specific gains might occur in children's future performance as a result of working with computers. However, these are some of the benefits that many people think may occur if young children are exposed to computers in a playful and enjoyable way.

(Editor's note: For coverage of how preschoolers across the country are using Commodore computers at Kinder-Care daycare centers, see the last issue (#24) of *Commodore*.) C

FOXSOFT™
Presents.....
SpryteByter™
For the Commodore 64™

The user affectionate sprite development program. Menu-driven, Mono/Multicolor sprites, joystick/keyboard, Tape/Disk. 20K w/FAST machine language routines. Over 60 commands: ROTATE (any angle 0-360), INVERT/OBVERT, SHIFTS, SYMMETRY, AND/OR, REVERSE, REVIEW, MOVIE (animation). Create and Edit up to 128 sprites per file. For programming efficiency and FUN!

PLUS
The Game Maker
Automatically prepares a base for game development-up to 32 sprites WITHOUT DATA STATEMENTS! Adds SPEED to your games. Saves memory too.

To order: Send check (U.S. Funds) $29.95 for Cassette, $34.95 for Disk. Dealer inquiries invited.

FOXSOFT™
P.O. Box 507
Deer Park, Texas 77536
(713) 473-6723

A Division of Foxfire Systems, Inc.

VIC-20™

VIC-20 INTERFACING BLUE BOOK
Did you know that your VIC can be used to control a 99¢ toy motor so effectively that it runs like a precision machine? Or that you can build an accurate digital thermometer using the VIC and four parts costing less than $5?
These and other 18 interfacing projects selected for usefulness, ease of construction and low cost are detailed in the VIC-20 Interfacing Blue Book, a veritable gold mine of practical information on how to build a variety of interfaces for your computer.
Projects include: Connecting VIC to your stereo; Pickproof digital lock; Capacitance meter; Liquid level sensor; Telephone dialer; Voice output; 8K/16K RAM/ROM expansion; 128K RAM expansion; 8-bit precision D/A; 8-bit A/D converter; MX-80 interface and more.
Written by a college professor in a friendly and informative style, the Blue Book gives you theory of operation, schematics, program listings, parts list, construction hints and sources of materials for each one of the 20 projects.
If you want to get the most out of your VIC this book is a must. Cost is $14.95 (less than 75¢ per project!). Price includes postage.

VIC-20 VIC-20 VIC-20 VIC-20 VIC-20
VIC-20 VIC-20 VIC-20 VIC-20 VIC-20

VIC 20 is a trademark of Commodore Electronics Ltd.

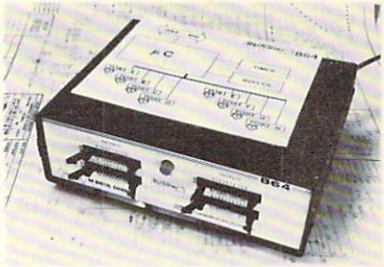
microsignal Dept C6
P.O. BOX 22
MILLWOOD NY 10546

Please send me a copy of the Blue Book.
Enclosed my check for $____
NAME __________
ADDRESS __________

Above prices include postage in the U.S. CA res. add 6% tax. Foreign add $2.

BUSSter™

LABORATORY INTERFACES FOR COMPUTERS



ANALOG AND DIGITAL INPUT/OUTOUT

The BUSSter interfaces provide analog and digital connections between any computer with an IEEE-488 or RS-232 interface and real world events. Each BUSSter product is self-contained, with its own case and power supply. They allow data acquisition while your computer is busy with other tasks. Built-in timer operates from .01 seconds to 48 hours.

- **BUSSter A64 —** 64 channel digital input module to read 64 digital signals. Built-in buffer *$495.00*
- **BUSSter B64 —** 64 channel digital output module to output 64 digital signals *$495.00*
- **BUSSter C64 —** 64 channel digital input/output module to input 32 and output 32 digital signals. Built-in buffer *$495.00*
- **BUSSTER D16 —** 16 channel analog input module to read up to 16 analog signals with 8 bit resolution (1/4%). Built-in buffer *$495.00*
- **BUSSter D32 —** 32 analog channel version of the D16 .... *$595.00*

Add the suffix -G for IEEE-488 (GPIB) or -R for RS-232.

All prices are USA only. Prices and specifications subject to change without notice.

30 DAY TRIAL — Purchase a BUSSter product, use it, and if you are not completely satisfied, return it within 30 days and receive a full refund.

US Dollars Quoted
$10.00 Shipping & Handling
MASTERCARD / VISA

CmC
Connecticut microComputer, Inc.
INSTRUMENT DIVISION
36 Del Mar Drive
Brookfield, Ct. 06804
(203) 775-4595 TWX: 710-456-0052

Fill In The Blanks

by Allen Patterson

A fill-in-the-blanks program for computer assisted instruction. This particular version of the program, which will run on any Commodore computer, helps students learn the correct forms of French verbs. But the program can be modified to accommodate many other applications. For any computer except the Commodore 64, delete line 98 in the program listing.

One of the most valuable assets that computers bring to education is their ability to supply immediate feedback in a non-threatening manner. However, if a new program has to be written (requiring valuable teacher's time) for every new skill that a student is expected to master, the value of the computer diminishes. In addition, in order for the computer to be truly effective in the classroom, it should present material consistent with other educational methods that have withstood the test of time. For example, many educators have relied upon a "fill in the blanks" type of exercise to reinforce learning, provide practice and review material. The computer can quite easily take this proven educational strategy and improve on it. Not only will the computer reward the student for correct responses but it will present the questions in a random order with the possibility that questions not answered correctly could be repeated. Alternately, these incorrectly answered questions could be recorded on paper for future reference.

The following program is set up so that the "fill in the blanks" sentences are located in data statements and can be changed at any time—by anyone. In the example that follows, the correct form of the French verbs etre, avoir, or aller are to be inserted. This

```
9 REM      ************************
10 REM     ** FILL IN THE BLANKS **
15 REM     ************************
20 :
30 :
50 REM       THIS PROGRAM WRITTEN BY
60 REM       ALLEN PATTERSON 83/3/24
61 :
62 REM BOX 178, BRAESIDE, ONTARIO
65 REM CANADA  K0A 1G0 (613)623-6867
70 :
75 REM  COPYRIGHT (C) 1983
78 :
80 REM TO ENTER DATA--FIRST RUN PROGRAM
81 REM AND PUSH STOP BUTTON SO THAT YOU
```

```
82 REM WILL HAVE UPPER AND LOWER CASE
83 REM LETTERS.
84 :
85 :
98 POKE 59468,14
99 NU=25
100 D$="[HOME,DOWN6]":DIM F(NU),F$(NU),Q$(NU),AN$(NU),AW$(NU)
110 FOR S=1 TO NU:READ Q$(S),AN$(S):NEXT S
145 TT$="ETRE, AVOIR, ET ALLER"
150 PRINT"[CLEAR]";D$;TAB(LEN(TT$)/2);TT$
160 PRINT"[DOWN2]ECRIVEZ LA FORME CORRECTE DU VERBE DANS LE TIRET."
165 GOSUB 600:REM IF STUDENT CHOOSES #  OF QUESTIONS THEN USE:GOTO550
170 :
200 J=J+1:A$="":IF J>NU THEN 1000
205 REM  IFJ>NE THEN 1000:REM USE THIS LINE IF STUDENT SELECTS # OF
    QUESTIONS
210 K=INT(RND(1)*NU+1):IF F(K)=1 THEN 210
220 F(K)=1:F$(J)=Q$(K):AW$(J)=AN$(K)
230 :
240 B=B+1:X$=MID$(F$(J),B,1):IF X$="*"THEN X=B-1:B=0:GOTO 260
250 GOTO 240
260 PR$=LEFT$(F$(J),X)+" -------- "+RIGHT$(F$(J),LEN(F$(J))-(X+1))
262 PRINT"[CLEAR]"
300 IF LEN(PR$)<40 THEN PRINT D$;PR$:GOTO 400
305 I=40
310 I=I-1:X$=MID$(PR$,I,1):IF X$<>" "THEN 310
320 Y=I
330 PRINT D$;LEFT$(PR$,Y):PRINT"[DOWN]";RIGHT$(PR$,LEN(PR$)-Y)
350 :
400 GET AN$:IF AN$<>""THEN 400
405 GET AN$:IF AN$=CHR$(13)THEN 500
410 IF AN$=""THEN 405
412 IF AN$=CHR$(20)OR AN$=" "THEN 420
413 IF AN$>CHR$(192)AND AN$<CHR$(219)THEN 420
415 IF AN$<CHR$(65)OR AN$>CHR$(90)THEN 405
420 A$=A$+AN$
425 IF LEN(A$)>10 THEN 500
426 IF AN$=CHR$(20)AND LEN(A$)=1 THEN A$="":GOTO 405
430 PRINT D$;TAB(X+1);"[RVS]";A$
435 IF AN$=CHR$(20)THEN A$=LEFT$(A$,LEN(A$)-2)
   :PRINT D$;TAB(X+1);"[RVS]";A$;CHR$(148)
440 GOTO 405
450 :
500 IF A$=AW$(J)THEN PRINT"[DOWN6,RVS]CORRECT![RVOFF]":R=R+1:
    GOSUB 600:GOTO 200
```

```
510 PRINT"[DOWN3,RVS]INCORRECT[RVOFF,SPACE]-- THE ANSWER IS: ";AW$(J)
512 IF LEN(PR$)<40 THEN PRINT"[DOWN]";PR$:PRINT"[UP]";TAB(X+1);
    "[RVS]";AW$(J):GOTO 517
514 PRINT"[DOWN]";LEFT$(PR$,Y):PRINT"[DOWN]";RIGHT$(PR$,LEN(PR$)-Y)
516 PRINT"[UP3]";TAB(X+1);"[RVS]";AW$(J)
517 REM:   F(K)=0:J=J-1:REM USE THIS LINE TO HAVE INCORRECT
    QUESTIONS REPEATED
520 GOSUB 600:GOTO 200
600 PRINT"[DOWN4,RIGHT7]PUSH [RVS]SPACE BAR[RVOFF,SPACE]TO CONTINUE"
605 GET G$:IF G$<>""THEN 605
610 GET G$:IF G$<>" "THEN 610
615 PRINT"[CLEAR]"
620 RETURN
680 :
690 REM  DATA GOES HERE: PUT QUOTATION MARKS AROUND QUESTIONS WITH
    A COMMA
693 :
694 REM  PUT QUESTION THEN COMMA THEN ANSWER
695 :
696 :
700 DATA"TU*L'AMI DE GEORGES?",ES
710 DATA"LA FILLE*FAIM. OU SONT LES SANDWICHS?",A
720 DATA"MONSIEUR LEBLANC*DANS LE RESTAURANT.",EST
730 DATA"NOUS*DINER A MIDI.",ALLONS
740 DATA"J'*CINQ ANS. QUEL AGE AS-TU?",AI
750 DATA"OU EST-CE QUE VOUS*?  JE VAIS A L'ECOLE.",ALLEZ
760 DATA"LES GARCONS*TRES GENTILS.",SONT
770 DATA"MAMAN*DEVANT LA MAISON AVEC PAPA.",EST
780 DATA"JE*TRES CONTENT QUAND IL NEIGE.",SUIS
790 DATA"MADAME, VOUS*LA SOEUR DE MADAME LEBRUN.",ETES
800 DATA"TU*JOUER AU HOCKEY APRES LES CLASSES?",VAS
820 DATA"LE CHIEN*A COTE DE LA MAISON.",EST
830 DATA"LES STYLOS DE MONSIEUR*SUR SON BUREAU.",SONT
840 DATA"ELLE*AU PARC POUR NAGER.",VA
850 DATA"ILS*SOMMEIL PARCE QU'IL EST DEUX HEURES DU MATIN.",ONT
860 DATA"JACQUELINE ET MOI*VISITER LA VILLE DE MONTREAL.",ALLONS
870 DATA"PIERRE ET GEORGES*LES FRERES DE SUZANNE.",SONT
880 DATA"TOI, TU*MON CHANDAIL, N'EST-CE PAS?",AS
890 DATA"ELLES*CHANTER A LA SOIREE.",VONT
900 DATA"GEORGES N'*PAS DE SOEURS.",A
910 DATA"NOUS*DANS LA MEME CLASSE QUE MARIE.",SOMMES
920 DATA"JE*PARLER AU DOCTEUR.",VAIS
930 DATA"CHANTAL ET MOI, NOUS*DE TUQUES BLEUES.",AVONS
940 DATA"VOUS N'*PAS DE SOULIERS.",AVEZ
950 DATA"ELLE*RESTER A LA MAISON PARCE QU'ELLE EST MALADE.",VA
```

```
1000 PE=INT((R/(J-1))*100)
1020 POKE 59468,12
1030 PRINT"[CLEAR,DOWN4]YOUR PERCENTAGE IS ";PE
8999 END
```

demonstrates the versatility of the program.

To fully appreciate the potential of the program, we should analyse each section individually.

Line 98 sets upper/lower case character mode.

Line 99 sets the number of questions to be asked (25 in the example).

Line 100 D$ is the location on the screen where the sentence will be printed. The dimension of variables is set at 25. (This will be changed if more or less than 25 questions are to be used.)

Line 110 reads the 25 questions and answers.

Lines 140-160 print the title and instructions.

Line 200 J is the question number being asked this time and limits the program to 25 questions. A$ is the answer input by the student and is set to be empty.

Lines 210 to 220 select a random number and then check to see if it has been selected before. If it has, a new number is selected. Setting F(K)=1 indicates that K has now been selected. F$(J) and AW$(J) are the question and answer to be dealt with this time around.

Lines 240 to 260 insert the blank in the proper place in the question.

Lines 300 to 330 insure that no words wrap around the screen. If the length of the statement is less than 40, it is printed. Otherwise, a space is found and the statement divided into two lines before printing.

Line 400 eliminates any accidental entries.

Lines 405 to 440 get entries one at a time and print them in the blank in the sentence. Lines 415 and 416 eliminate unwanted entries (e.g., graphics).
Line 425 limits the length of the answer to ten characters (this may be altered as needed).

Lines 500 to 520 separate correct and incorrect responses. The word "correct" could be replaced by a suitable graphics subroutine to be called up at this time as a reward. (Don't forget to POKE 59468,12 before the graphics characters are needed and POKE 59468,14 after the subroutine is completed and before returning.) If the answer is incorrect, the correct answer is given. Use line 517 if you wish this question to reappear sometime later. In addition, the question answered incorrectly could be printed on paper if desired.

Lines 600 to 616 are used as a subroutine to halt the program until the space bar is pressed. Line 605 eliminates premature return from the subroutine.

Lines 690 to 698 are instructional reminders that the data should be entered with the use of quotation marks. This allows for the use of commas in the sentences. Don't forget to put an asterisk where the blank is to be inserted.

Lines 700 to 950 are the data statements.

Line 1000 computes a percentage score.

Line 1020 returns computer to graphics mode. Once again a graphics reward routine could be used instead of line 1030.

Another variation would be to ask the user how many questions he/she would like to try. Get this number by using a subroutine similar to lines 400-440. The following changes would work:

```
165 GOTO 550
205 IF J>NE THEN 1000
550 PRINT"[DOWN3]HOW MANY QUESTIONS WOULD YOU LIKE TO TRY?";
560 GET K$:IF K$<>""THEN 560
565 H$=""
570 GET K$:IF K$=CHR$(13)THEN 580
571 IF K$=""THEN 570
572 H$=H$+K$:IF LEN(H$)>2 THEN 580
575 PRINT K$;:GOTO 570
580 NE=VAL(H$):IF NE>0 AND NE<26 THEN 590
585 PRINT:PRINT"CHOOSE A NUMBER BETWEEN 1 AND ";NU:FOR T=1 TO 1000:
    NEXT T :GOTO 150
590 GOTO 200
```

As you can see, this program would be very useful and very adaptable. In fact, many of the above subroutines would fit nicely into other programs of your own. C

What does **COMMODORE** have that Apple, Radio Shack and IBM don't?

backpack™

BATTERY BACKUP SYSTEM

For CBM™/PET® 2000, 4000, 8000, and 9000 series computers and CBM 4040/8050 dual disk drives. Installs within the cabinets of the computer and disk drive. Recharges continually from the machine's own power supply and automatically supplies 30 minutes (max.) of reserve power during outages. Also eliminates surges and spikes. User installable.

OFFICIALLY APPROVED BY COMMODORE

In Canada call: Van-Hoy Group (604) 542-1138 or (604) 545-0794
In United Kingdom call: Wego Computers (0883) 49235

ETCETERA OF CSC CORPORATION
APEX, NORTH CAROLINA, U.S.A.
(919) 362-4200

SOLD ONLY BY INTELLIGENT COMPUTER DEALERS • DEALER INQUIRIES WELCOME

PETSpeed Tips

by Joe Rotello

We're happy to have our PETSpeed expert from Tucson begin a regular column with this issue, so our readers can keep up with the latest developments for using this popular BASIC compiler to their best advantage.

Welcome to PETSpeed Tips! This column will be devoted to the pursuit of PETSpeed™ and the Integer BASIC Compiler™. In response to many requests for data, tips, "inside information", programming aids and program reviews, this column is dedicated to Commodore users everywhere. Please support us and help keep this column going by sending us your questions, problems, ideas and any software that you have put under PETSpeed and/or Integer BASIC.

We will try to include topics relating to each *Commodore* magazine issue "theme" as well. This month we will discuss some topics related to business uses of PETSpeed/Integer BASIC.

PETSpeed Update

In early May a new version of PETSpeed was introduced. Version 3.0 now allows for use with the PET "fat forty" computer as well as the 8000 series CBM. Memory locations immediately below the start of BASIC, decimal 1023 and below, are no longer required by PETSpeed.

PETSpeed for the Commodore 64 is now out. The operating procedure is nearly exactly the same as in the PET/CBM version. The program cannot be run as it is received, however. The user must first make two backup copies on the 1541 disk drive: a "PETSpeed Master" and a "Utilities Master". The programs have to be split over two disks due to the large number of PETSpeed system and utilities programs present.

The Commodore 64 "security podule", otherwise known as a dongle, is placed into either the cassette port or control port 2, depending on which podule type is supplied. Note that, as in the case of the 8000 series version, the security podule/dongle is required only for compiling the actual BASIC source code.

When compiling on the Commodore 64/1541 system, the disk should contain only the PETSpeed system programs and the BASIC source code. Disk space is at a premium on the single drives compared to the dual disk drives. With the advent of PETSpeed on the 64, users and programmers now have a viable way to generate and make excellent use of fast and efficient business programs where the speed of compiled BASIC is necessary.

Questions & Answers

Q: Can PETSpeed be used to compile an existing business package, for example an accounting system that presently runs on the PET/CBM/Commodore 64??

A: Yes, but with a few precautions:

a) Under many circumstances, the BASIC source code must not contain any machine language SYS calls. Although most problems with this situation can be programmed around, such changes are best left to experienced programmers.

b) Since the compiled version of the program(s) will take up more disk space than the BASIC counterpart, be careful to not run out of disk space, especially when the program suite consists of multiple programs on the same disk. This problem will be most evident on the 1541 disk drives, where it is common to store both programs and data on the same disk.

c) We are beginning to see many software suppliers rerelease their business and homeowner software in PETSpeed versions. This should aid in clearing up any potential problems caused by (a) and (b) above.

Q: How can PETSpeed access a machine code subroutine??

A: The instructions and charts included with the PETSpeed manual are indeed a little dry. But by careful examination and trial-and-error testing on a simple program, the method of accessing variables is very clear. The key is to locate where PETSpeed stores your variables and subscripts. This is made easy by the REPORT program present on the PETSpeed system or utility disk.

It is easy to allow PETSpeed to work with machine language routines if those routines are POKEd into memory via data statements. In that case, make sure that the machine language does not conflict with the PETSpeed program. Again, refer to the PETSpeed system map and the output from the REPORT program.

We will be discussing variables, and how they are treated by PETSpeed, in our next column.

Q: Can PETSpeed be used in a modem program?

A: We assume that you mean, "can an existing operational modem program be compiled?" In general, yes. If the data character conversions (ASCII to PET, PET to ASCII) are carried out in BASIC, the PETSpeed version will not only operate easily at 300 baud, you will be able to add more options to your modem program without affecting the overall program performance. Be sure to read the two Q/A above for further information.

Q: I have a BASIC program that does bit-level work. Will it function under PETSpeed?

A: With a few reservations, yes. We have not yet seen a bit-level BASIC program that did not function well under PETSpeed. By the way, bit-level execution under PETSpeed is about five times as fast as the BASIC counterpart.

The reservations concern the long code that many programmers use in BASIC, sometimes exceeding 75 characters! PETSpeed may need the source code line broken up into two distinct parts in order to accept it.

Q: I have heard a rumor that it is possible to change a BASIC program to get up to 50% faster execution under PETSpeed, than even PETSpeed normally does. Is this true?

A: True, but 25% to 30% is a better figure. See this month's tips section below.

PETSpeed Tips

Did you know that even PETSpeed can be given a boost? Well, not PETSpeed itself, but by making very minor changes in your BASIC source code, you can gain even more speed out of the compiled version. Here are a couple of tips:

1. Under PETSpeed, POKEs and PEEKs can be negative numbers! (What?) PETSpeed allows negative numbers to be assigned to the PEEK/POKE ranges you request. See Program 1 for a small sample. This program is intended to be compiled (the negative POKE routine won't work in BASIC) and the times required to fill the screen will be displayed. The range of POKEs will have to be modified if you have a 40-column PET, and the POKE values themselves will have to be changed for the Commodore 64.

2. In CBM BASIC, the CMD command can and is used to change the default output device; any print commands carried out after the CMD are directed to the device that you CMD'ed (sorry, bad English) until you turn it off with the appropriate PRINT # command. Nice in BASIC; even faster when compiled under PETSpeed. An example is:

```
PROGRAM:   PETSPEED EX

 10 OPEN 5,8,8,"0:TEST
    ,S,W"
 20 CMD 5
 30 FOR I=0 TO 100
 35 PRINT I
 50 NEXT
 60 PRINT#5
 70 CLOSE 5
```

You are reading correctly. Line 35 says "PRINT I" instead of the familiar "PRINT #5,I". And likewise, line 60 has to have the "PRINT #5" command in it in order to insure that the file is properly closed.

Using this method, file data transfer is about 15% to 25% faster than the traditional BASIC code.

Ok, now for the goodie we promised. See Program 2? Ok, that program is made to be compiled exactly as shown (well, you can have different line numbers if you want), and it reads the simple data laid to disk by the program above. Lines 20 and 40 are NOT misprints. Under PETSpeed, they are valid operators and commands.

The beauty of the code is that the file data transfer rate of the PETSpeed version of Program 2 is about 30% *faster* than the PETSpeed version of a so-called "normal" way of coding!!

Aha! There ARE ways to give even PETSpeed a helping hand!

Feel free to use the above ideas in your own programs and enjoy NEW! MORE POWERFUL! PETSpeed!! (Commercial is over)

Remember, the code shown in Programs 1 and 2 will *not* work in BASIC. They are made especially to be compiled under PETSpeed, or to be part of a BASIC source code that will be compiled later. **C**

Program 1: PETSpeed with Negative POKEs

```
10 PRINT"[CLEAR,UP]";
20 INPUT"WHICH POKE (NEG (OR) POS)";A$
25 IF A$="P"THEN 100
30 IF A$="N"THEN PRINT"[CLEAR,UP]";:TI$="
   000000":FOR I=-32767 TO-30768
40 POKE I,156:NEXT:PRINT"[HOME,DOWN2]"
   ;TI/60" SECONDS"
45 GET A$:IF A$=""THEN 45
50 GOTO 10
100 PRINT"[CLEAR,UP]";:TI$="000000":FOR
    I=32767 TO 34687
140 POKE I,156:NEXT:PRINT"[HOME,DOWN2]
    ";TI/60" SECONDS"
145 GET A$:IF A$=""THEN 45
150 GOTO 10
```

Program 2: PETSpeed with Fast File Transfer

```
10 OPEN 5,8,8,"0:TEST,S,R"
20 #5
30 FOR I=0 TO 100
40 GET A$
50 PRINT A$;
60 NEXT
70 PRINT#5
80 CLOSE 5
```

Calling on LOG() and EXP()

by C. D. Lane

So you always thought logarithms didn't have any place in programming, did you? In this very clear explanation of what could be a murky subject, C. D. Lane shows how logs can work, directly and indirectly, to add speed and power to your programs.

Tables of logarithms were first published in 1641 by John Napier, and logarithms are still in use today. Even the BASIC language on your microcomputer uses them in the guise of the numeric functions EXP() and LOG(). Some of us may remember that logarithms are the basic mechanism behind the slide rule (the devices scientists carried on their belts before calculators) (just as computers on belts will come into fashion!). What do logarithms do and what use are they in programming?

A logarithm is an *exponent.* It is defined in terms of a *base.* The logarithm of a number is the power the *base* has to be raised to in order to equal the number. One can take logarithms of any positive number greater than zero (the *domain* of logarithms), and the logarithm itself can be any real number (the *range*).

Although logarithms can use any number as a base, only a few are commonly used. In mathematics we learn about logarithms of base ten (common logarithms or Brigg's system), the base of our number system. The base of our computer's number system is two, and this base for logarithms is useful for computer work which we will discuss later. Another common base for logarithms is the constant *e*.

Logarithms to the base *e* (2.71828183 in our microcomputer's floating point) are called natural logarithms and are notated *ln*(). They are "natural" since various events in nature can be quantified using the natural log (such as the decay rate of capacitors). The natural log is the one included on Commodore computers, among others. You can find out what base your computer uses by evaluating EXP(1), that is the base raised to the first power.

The Definition of *ln* ()

The *ln*(X) (natural log of X) is defined by calculus as the area from 1 to X under the curve 1/X (see Figure 1). We can also calculate logarithms using polynomial or series evaluation. This is the method our computer uses. Series evaluation is a relatively fast method that allows us to get as close an approximation as we need, although not always the exact solution provided by calculus. In our computer's BASIC interpreter are tables of constants for doing this evaluation.

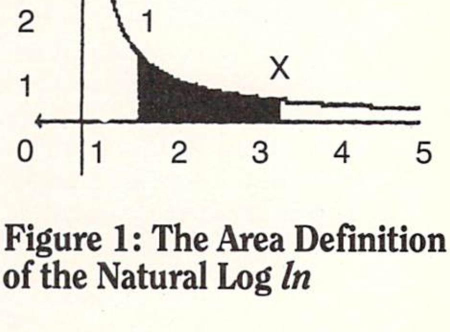


Figure 1: The Area Definition of the Natural Log *ln*

Logarithms have special properties that make them very important. One property of logs is that log(A*B) = log(A) + log(B), along with the variants that can be derived from this property:

log(A/B) = log(A) − log(B)

log(A^B) = B*log(A)

The EXP() function is the *inverse* of the *ln*() function, the *antilog*, meaning that A = exp(log(A)) = log(exp(A)). The notation EXP(X) is just another way of notating e^x; both are equivalent. The EXP() function has the same properties as any exponent, such as:

exp(A)*exp(B) = exp(A + B)

exp(A)^B = exp(A*B)

Now if we combine the logarithms with the EXP() function we get:

A*B = exp(log(A*B)) = exp(log(A) + log (B))

A/B = exp(log(A) − log(B))

A^B = exp(B*log(A))

This means we can multiply by adding, divide by subtracting and raise to a power using multiplication; all the tricks the slide rule uses.

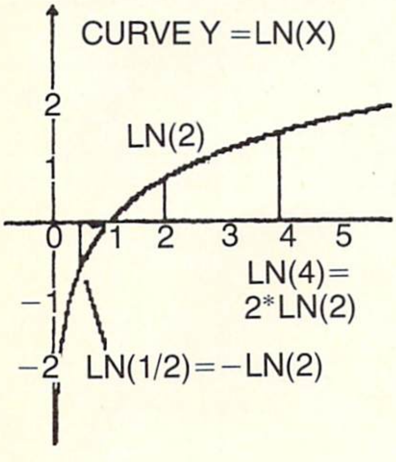


Figure 2: Comparison of Logarithms Along y=ln(x)

Some of these relationships can be seen graphically in Figure 2.

We can see from Figure 2 why logarithms of zero or less are not allowed; the function approaches but never quite reaches zero. The function crosses the X axis at (1,0) or log(0)=1, for *all* bases.

Now that we have established what logarithms are, how are they used in our microcomputer? If we time the following equivalent expressions on our computer over a range of values we notice a surprising result.

```
10 C=A↑B
20 C=EXP(B*LOG(A))
```

The time it takes to do the second expression is only slightly longer than the time it takes to do the first—not an intuitive result based on the apparent difference in complexity on first glance. If we dig a little deeper, however, we find that BASIC uses the code for LOG() and EXP() to evaluate expressions using the ↑ operator! In fact the evaluation of the ↑ operator is done along the same lines as the second expression above. The reason it takes slightly longer to evaluate the second expression is that this expression does its function calls from BASIC while the first expression does its function calls in machine language. Did you realize that every time you used ↑ (or even SQR()) in your program you were actually using those functions LOG() and EXP() which you thought you *never* use?

Another Useful Base for Logarithms

Another useful base for logarithms, for us computer fans, is two, the number base of our computer. (For an introduction to the base two number system, see Jeff Hand's article in Issue 24.) Another special property of all logarithms is that given any logarithm function in one base we can derive logarithms in any other base:

$\log_a B = \log_n B / \log_n A$

To get logarithms of base two on our computer we can do the following:

```
10 DEFFNL2(X)=LOG(X)/LOG(2)
```

where FNL2(X) gives us $\log_2$(X). One can define a logarithm of base ten, or any base, in a similar fashion. Now how are *ln*() and $\log_2$() useful to us beyond their scientific uses? One use of logarithms allows us to examine the number system of our computer.

Our computer manual tells us that the maximum and minimum numbers our computer can represent are 1.70141183 * 10^{38} and 2.93873588 * 10^{-39}. Rather ragged looking numbers; where do they come from? If we take their $\log_2$ we get 127 and −128, even powers of two, the base of our number system. This means that for our computer the maximum and minimum numbers we can represent are 2^{127} and 2^{-128}. Our manual also says that we can use EXP() on numbers between 0 and 88.0296919, and with our new-

COW BAY COMPUTING
has a lot to offer you and your
PET/CBM/COMMODORE 64

SOFTWARE FOR THE CLASSROOM

- The Pet Professor $499.00
- With Management System $649.00

A total arithmetic package with step-by-step instruction.77 programs in addition, subtraction, multiplication and division on cassettes or disks. Ask for a sample.

Specify disk drive and computer configuration.

WORKBOOKS FOR COMPUTER LITERACY

- Feed Me, I'm Your PET $5.95
- Looking Good With Your PET $5.95
- Teacher's PET $5.00

STUDENT SCHEDULING SYSTEM

- Fast, accurate, easy-to-use and less expensive than current methods.
- Uses a 8032 computer and 8050 disk drive
- 90 Day Money Back Guarantee to prove that it can be done.

COW BAY COMPUTING
P.O.Box 515
Manhasset, N.Y. 11030
(516) 365-4423

found knowledge we compute:

```
10 MAX=127*LOG(2)
```

where Log() is *ln*() from before and we find out where the maximum number we can use EXP() on comes from. Now what use is this exact figure to us? We can use it for overflow detection, a practical use of logarithms in programming.

We have two numbers; we want to raise the first to the second, but the result may be larger than the computer can handle. If this happens while some user is inputting values to our program, the program will halt with an error, a very undesirable result. We can determine if this will happen before it happens, and avoid it, using LOG() and EXP(). Using MAX as defined above:

```
20 INPUT"X";X
30 INPUT"Y";Y
40 Z=Y*LOG(X)
50 IFZ>MAXTHENPRINT
   "OVERFLOW!":GOTO20
60 PRINT"X↑Y=";EXP(Z)
   :GOTO20
```

This kind of test can be done for multiplication, division or any operation that can cause an overflow or underflow, allowing our program to detect and correct otherwise fatal errors.

Another use of logarithms in everyday programming is for bit detection. We will use the $\log_2()$, or FNL2(), for this. If we define:

```
10 DEFFNL2(X)=LOG(X)/LOG(2)
20 DEFFNCH(X)=INT(FNL2(X))
```

FNCH(X) gives us the part of the logarithm to the left of the decimal point, or as it is known in mathematics, the *characteristic*. Before calculators, people looked up logarithms in tables, where usually only the decimal part was included, and the characteristic was left for the user to determine. For $\log_2()$, the characteristic tells us the highest power of two in the number, allowing us to easily find the left-most bit in a given number. We can subtract off this value and call $\log_2()$ again, until we have found all the bits we are looking for. This procedure only needs to be repeated for each bit that is on, not the zero bits. Compare this to stepping through the number, comparing powers of two, where we have to test every bit every time!

Even though we may not directly use LOG() and EXP() in our everyday programming, we indirectly call upon them all the time in evaluating mathematical expressions, where they are used to our advantage, speeding up calculations when possible. Furthermore, logarithms are useful for defining new functions that we can directly apply in our computer programs, increasing both their speed and power. **C**

Getting the Most Out of (And Into) Your Disk Drive
Part 3

by John Heilborn

This is part three of a three-part series on getting more out of your disk system. In this section you will learn some of the basic concepts of developing mailing list programs, from data entry through list sorting.

The Screen Display

One of the most important features of any good program is its ease of use. For the most part, computers do not perform functions that people cannot perform. They just help people do the jobs faster and easier.

Keeping this in mind, let's write a routine that will display a menu of the functions the operator can select. Here's a routine that displays a heading, the options and a prompt line. You can either use this screen display or write your own, but try to keep it as simple as possible; we're designing for function not beauty.

```
10 REM ** DISPLAY MENU **
20 PRINT "(SHIFT CLR/HOME)";
30 PRINT "(CTRL RVS/ON)          M E N U        (CTRL RVS/OFF)"
40 PRINT: PRINT: PRINT
50 PRINT "(CRTL RVS/ON)1(CTRL RVS/OFF)... FORMAT DISKETTE"
60 PRINT
70 PRINT "(CRTL RVS/ON)2(CTRL RVS/OFF)... NEW ITEM"
80 PRINT
90 PRINT "(CRTL RVS/ON)3(CTRL RVS/OFF)... FIND ITEM"
100 PRINT: PRINT
110 PRINT "(CRTL RVS/ON)4(CTRL RVS/OFF)... UPDATE ITEM"
120 PRINT: PRINT
130 PRINT "ENTER SELECTION _"
140 GET A$: IF A$ = "" THEN 140
150 IF A$ = "1" THEN 200
160 IF A$ = "2" THEN 300
170 IF A$ = "3" THEN 400
180 IF A$ = "4" THEN 500
```

Let's review the routine. First, line 20 clears the screen readying it to display our menu. Line 30 displays the heading MENU in reverse at the top of the screen. Lines 40-110 display our four options and the prompt line. Finally, lines 120-190 accept the operator's menu selection. Note that if the input is not one of the four we allow, the program will return to the selection input line (140). This keeps the operator from accidently entering the wrong thing and crashing the program.

Once the operator has made a selection, our menu transfers control of the program to one of four routines. These are:

Line 200: Format a diskette
Line 300: New item
Line 400: Find an item
Line 500: Update an item

Each of these functions will become independent routines. To write the routines as easily as possible, let's define each of them first.

Formatting a Diskette: Creating a Directory

Ordinarily, when you SAVE a file using the DOS, the data is stored and a directory entry is made for you automatically. However, this program doesn't use the system SAVE because the system is limited to 142 files and with this routine, we'll be able to put more than 600 files onto a single diskette. By not using the system SAVE, however, we'll need to make our own directory entries.

The easiest way I've found to do this is to set up alphabetical files when you format your data diskette in the first place. This also allows you to incorporate a FORMAT routine into your program, making it easier for an operator to set up a new diskette.

This FORMAT routine asks the operator to name the diskette. The name of the diskette is then combined with an internally generated random number which is used in the diskette name and is also used to generate a diskette number. By giving each diskette a different number, the computer will be able to determine what diskette is in the drive and when to update the Block Availability Map (see Part 2 of this series).

```
200 INPUT "DISKETTE NAME"; D$
205 OPEN 15,8,15,"N:"+D$+",W"
210 CLOSE 15
215 DATA A,B,C,D,E,F,G,H,I,J,K,L.
    M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z
220 READ A$
225 OPEN 1,8,4,A$+",W"
230 PRINT #1, A$
235 CLOSE 1
240 IF A$ = "Z" THEN 20
245 GOTO 220
```

This is how the routine works. First, in line 200 it gets a diskette name from the operator. Lines 202 and 210 OPEN a command file, FORMAT the diskette (using the INPUT name, D$) and CLOSE the command file.

Line 215 is a DATA statement containing the names of all the alphabetical files we need to write onto the diskette. Line 220 reads the file names while 225-235 OPEN, write the files onto the diskette and CLOSE the files. Line 240 looks for the end of the data (the letter Z). When the files have all been written, it returns to the main menu routine.

Entering a New Item

This is just another data entry routine. Like the menu routine, it should have a header, entry options and a selection line. In this case, we'll also want to have a line that allows the operator to enter data.

Here's a routine that includes all of the above features:

```
300 REM ** NEW ITEM **
305 PRINT "(SHIFT CLR/HOME)";
310 PRINT "(CTRL RVS/ON)   N E W   I T E M    (CTRL RVS/OFF)"
315 PRINT: PRINT
320 PRINT "(CRTL RVS/ON)1(CTRL RVS/OFF)... LAST NAME"
325 PRINT
330 PRINT "(CRTL RVS/ON)2(CTRL RVS/OFF)... FIRST NAME"
335 PRINT
```

```
340 PRINT "(CRTL RVS/ON)3(CTRL RVS/OFF)... STREET ADDRESS"
345 PRINT
350 PRINT "(CRTL RVS/ON)4(CTRL RVS/OFF)... CITY"
355 PRINT
360 PRINT "(CRTL RVS/ON)5(CTRL RVS/OFF)... STATE"
365 PRINT
370 PRINT "(CRTL RVS/ON)6(CTRL RVS/OFF)... ZIP CODE"
375 PRINT:
380 PRINT "(CTRL RVS/ON)7(CTRL RVS/OFF)... SAVE"
385 PRINT
390 PRINT "(CTRL RVS/ON)8(CTRL RVS OFF)... EXIT"
392 PRINT: PRINT "ENTER SELECTION_": GOSUB 600
395 GOTO 305
```

All this routine does is clear the screen and display the NEW ITEM option menu. The reason we don't want to perform the data entry part of this routine here is that the UPDATE routine can use the input subroutine we'll be writing at line 500 also.

Data Input

The easiest way to enter data from the program above is by creating an array. This is just a series of data that has the same variable name combined with a unique number to distinguish it from the other members of the array. For example, if you had a list of seven variables that needed to be defined, you could give each member a different name such as:

SLEEPY
HAPPY
BASHFUL
DOC
GRUMPY
SNEEZY
DOPEY

Or, you could give them all a common name and differentiate them by giving each a unique number such as:

DWARF (1)
DWARF (2)
DWARF (3)
DWARF (4)
DWARF (5)
DWARF (6)
DWARF (7)

By giving each member of an array the same name and a unique identifying number you can more easily access each member of the array. Here's one way to do it:

```
600 REM ** INPUT ROUTINE **
610 GET A$: IF A$="" THEN 600
620 IF VAL(A$)=0 THEN 600
630 A=VAL(A$)
640 IF A=7 THEN 700
650 IF A=8 THEN 20
660 PRINT "ITEM";A;
670 INPUT I$(A)
680 RETURN
```

The routine above will work as a standard input routine for this program. This is how it works. First the routine looks for a single-number input. We're inputting into a string variable to avoid letting the operator bomb the program. If we tried to input into a numeric variable and the operator accidently entered a string value, it would cause BASIC to respond with an error and would scroll up the screen by one line. This would move the heading up out of view. Also it is better to remain in control of the program at all times. Using a string here accomplishes that.

The next thing we do is check the input value. If an invalid entry has been made, the program returns to the input line (610). If the entry was valid, the program checks to see if either a 7 or 8 was entered. If the entry was 7 then the program jumps to line 700 which will be our SAVE routine. If the entry was 8, the program returns to the MENU (line 20). If the entry was an input selection, the number of the selection is automatically translated into one of the string variables in the array and the INPUT is stored in that variable. Finally, the program returns to the NEW ITEM selection screen.

Saving The Data

The only routine that remains to be written for the NEW ITEM routine is a SAVE subroutine (this will also be used by the UPDATE routine). The lines for this subroutine have been derived from the routines developed in the first two parts of the article. Let's apply the programs here. First we need to allocate a sector:

```
700 REM ** SAVE ROUTINE **
710 OPEN 15,8,15
720 PRINT#15, "B-A:"0;1;1
730 INPUT#15, A,B$,T,S
740 IF B$="OK" THEN T=1:S=1:
    GOTO 760
750 PRINT#15, "B-A:"0;T;S
```

and store the data in the allocated sector:

```
760 PRINT#15, "B-W:"4;0;T;S
770 CLOSE 2: CLOSE 15
```

Then we'll have to save our file name in the directory so the data can be found again later. To SAVE the file name, look at the first letter of the name:

```
780 F$ = LEFT$(I$(1),1)
```

Now take that letter and OPEN the appropriate file.

```
790 OPEN 1,8,4,F$+",R"
```

Here's the tricky part. We need to append (add to the end of the file) the name of our new file. Unfortunately, the VIC doesn't have a DOS command that does an append, so we need to create one.

One way to append a file is to first OPEN a new file and read the existing one into it. Then before CLOSEing the new file, we write the information we want to add to the end of it. All that we have to do then is delete the old file and rename the new file with the old name.

```
800 OPEN 2,8,4,"TEMP,W"
810 INPUT#1,A$
820 PRINT#2,A$
830 IF ST=0 THEN 810
840 CLOSE 1
850 PRINT#2, I$(1)
860 PRINT#2, C
870 PRINT#2, D
880 CLOSE 2
890 OPEN15,8,15,"S:"+F$
900 PRINT#15," R:TEMP=" + F$
910 RETURN
```

The Find Function

The FIND function is the simplest function in this program. All you need to do to find a file is prompt the operator for the name of the file. You then look in the appropriate directory (alphabetic file) for the matching name and read in the data using the track and sector that is stored in the file following the name.

```
400 INPUT "FILE TO FIND"; FI$
405 FR$ = LEFT$(FI$,1)
410 OPEN 1,8,4,FR$ + ",R"
415 INPUT#1, G$
420 INPUT#1, G$, T, S
425 IF G$ = FI$ THEN 440
430 IF ST = 0 THEN 420
435 CLOSE 1: PRINT "FILE NOT
    FOUND": RETURN
440 CLOSE 1
445 OPEN 15,8,15
450 OPEN 2,8,4,"#"
455 PRINT#15,"B-R:"4;0;T;S
460 FOR R = 1 TO 6
465 INPUT#2, I$(R)
470 PRINT I$(R)
475 NEXT
480 RETURN
```

Updating a File

The last routine we'll need in this program modifies an existing item. To change an existing item, we'll need to look it up on the diskette first. Use the FIND routine above to find your item.

```
500 GOSUB 400
```

Then prompt the operator for those items that need to be changed:

```
502 REM ** UPDATE ITEM **
505 PRINT "(SHIFT CLR/HOME)";
510 PRINT "(CTRL RVS/ON) U P D A T E  F I L E (CTRL RVS/OFF)"
515 PRINT: PRINT
520 PRINT "(CRTL RVS/ON)1(CTRL RVS/OFF)... LAST NAME"
525 PRINT
530 PRINT "(CRTL RVS/ON)2(CTRL RVS/OFF)... FIRST NAME"
535 PRINT
540 PRINT "(CRTL RVS/ON)3(CTRL RVS/OFF)... STREET ADDRESS"
545 PRINT
550 PRINT "(CRTL RVS/ON)4(CTRL RVS/OFF)... CITY"
555 PRINT
560 PRINT "(CRTL RVS/ON)5(CTRL RVS/OFF)... STATE"
565 PRINT
570 PRINT "(CRTL RVS/ON)6(CTRL RVS/OFF)... ZIP CODE"
575 PRINT:
580 PRINT "(CTRL RVS/ON)7(CTRL RVS/OFF)... SAVE"
585 PRINT
590 PRINT "(CTRL RVS/ON)8(CTRL RVS OFF)... EXIT"
592 PRINT: PRINT "ENTER SELECTION_": GOSUB 600
595 GOTO 505
```

The last thing you'll need to do to finish the UPDATE routine is to SAVE the modified file. Enter:

```
597 GOSUB 800
```

and return to the main routine:

```
599 GOTO 20
```

C

PET Bits

by Elizabeth Deal

Raeto West's book, *Programming the PET/CBM*, confirms your worst suspicions about tape: you cannot save any area of memory higher than $7FFF (33767). Writing CHR$(PEEK(x)) to file, unfortunately, can't work, because several characters (0, 10, 29) can't be written to tape. The solution is to move the contents to a saveable area (forj=0tox: pokem2+j,peeKm1+j:next), then save it via the monitor.

The book includes a nicely annotated memory map, wedge techniques, machine language coding with real, live PET examples and is really a goldmine of information about BASIC programming. It is quite tutorial about machine code, and is probably the best reference on the details of disk you'll find anywhere.

West's book also blows a whistle on one slight misunderstanding about how BASIC functions in finding a line of a GOTO statement. The PET goes hunting for a line from the beginning of a program only when the desired line number is lower than the calling number; otherwise the PET goes forward. A selective placement of your subroutines makes sense in some circumstances, but don't worry about short forward jumps. It's the short backward ones that cost a bit of time. For instance, assume an evenly numbered program from 100: if we're now on line 500, GOTO 550 cost us practically nothing; GOTO 150 costs us practically nothing; but GOTO 450 takes some time.

° ° °

Data base management is a buzzword for organizing your data files. There are several valuable programs on the market for business and large applications. For many home computer users such programs are an overkill in terms of price and sophistication.

A cheap data base management system can be had for next to nothing: i.e., no system at all. All you need to do is write your data in program lines, edit them using PET's superb screen editor and search and change using such aids as Commodore's BASIC Aid or POWER (from Professional Software). A program is the ultimate in random accessibility. You can access what you want by using search commands, you can change segments, you can add and delete data and, of course, you can store it in the fastest imaginable way by saying SAVE.

One restriction: after a line number it's a good idea to have a non-numeric entry. A colon or quote work well. The system is cheap, workable and universal until someone designs a system that does not permit you to put garbage into BASIC lines. Let's hope it never happens.

Any program file-reading command, such as LIST "file" of POWAID2 can display such a file on the screen without disturbing a program in memory, so you don't even have to load to see it! (POWAID2 and POWAID4 are public domain programs written by Brad Templeton as extensions of POWER).

If you already have WordPro™ (Professional Software), this too can be a good filing program. Your data can be edited by the best editor around and sequential files can be put out for further processing. You'll be amazed at WordPro's usefulness beyond its normal purpose. The instructions for doing such things are buried at the end of the manual, but they are all there.

For instance, fast conversion of ASCII code to screen code for just a few characters can be done by:

```
PRINT"clear screen, several characters"
For J=1 TO number of characters
S(J)=PEEK(32767+J):NEXT J
```

A cheap data base management system can be had for next to nothing: i.e., no system at all. All you need to do is write your data in program lines, edit them using PET's superb screen editor and search and change using such aids as Commodore's BASIC Aid or POWER.

Array S will hold the screen code values.

Decimal to hexadecimal conversion of large numbers is tough; you NEED a computer for it. But hex to decimal or bit-string can be done in your head or using the PET's direct mode. For example, the processor's status word can be easily understood by 8421 8421 sequence. Try it: convert $4D. Is the decimal flag on or off? Try another: Convert $4D to decimal: 64*4+13. Or an address: $1234 is 1*4096+2*256+3*16+4.

WordPro 3 has some undocumented instructions that are handy for wedge addicts. There are two ways of initializing a floppy and using the utilities. One way is as described in the book. The other way is, you guessed it, ctrl> (rvs followed by greater-than key) followed by your command.

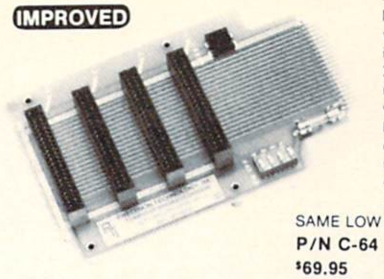
o o o

SUPERGRAPHICS by John Fluharty (sold by AB Computers in Colmar, Pennsylvania) has several self-contained graphic subroutines that can be used alone, without turning the entire package on. Many, but not all (experiment!) commands can be worked like this: SYS(x)list of parameters separated by commas. C

EXPANDER BOARDS

CBM 64™

IMPROVED

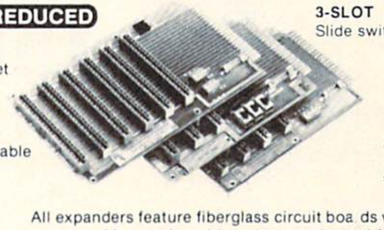


4-SLOT
Newly designed unit with solid-state switching on the GAME and XROM lines for universal compatibility with all cartridges. Normal computer operation is unaffected by cartridges left plugged in.

SAME LOW PRICE
P/N C-64
$69.95

VIC 20™

REDUCED



6-SLOT
Toggles, fuse and reset
P/N V-36
$69.95

6-SLOT with 3-ft. ribbon cable
P/N V-46
$89.95

3-SLOT
Slide switches and fuse
P/N V-23
$49.95

4-SLOT
Toggles, fuse and reset
P/N V-24
$59.95

All expanders feature fiberglass circuit boards with epoxy solder mask, gold contacts and metal feet.

MONITOR/AUDIO CABLE

Connects VIC 20 or CBM 64 to audio amplifier and TV monitor

Color 64 or VIC P/N MC-2 $12.95
B & W 64 only P/N MC-3 $12.95

PRECISION TECHNOLOGY, INC.
COMPUTER PRODUCTS DIVISION
P.O. BOX 15454
SALT LAKE CITY, UTAH 84115
(801) 487-6266

See your dealer or place your order directly

VISA - M/C - CHECK - COD

TM-Trademark of Commodore Electronics Limited

Loading Commodore 64 Programs Into the PET and Back

by Elizabeth Deal

Since the VIC 20 and the Commodore 64 appeared on the scene there seems to exist an epidemic of people who need to load VIC and 64 programs into the PET. Several ways have been proposed. Most assume that the programs will load in at $0801 (2049 decimal). Some methods I have seen require several nasty POKEs and, to make matters worse, require knowing where the program came from. I found myself continually creating Commodore 64 partitions on the PET and hunting for programs, until one day it dawned on me that the solution was staring me right in the face.

I found myself continually creating Commodore 64 partitions on the PET and hunting for programs, until one day it dawned on me that the solution was staring me right in the face.

Generally it involves using some sort of a toolkit program or the tape or disk merge methods of Brad Templeton and Jim Butterfield. These types of programs are relocators by definition. The XEC command of POWER does the job for you. If you don't have POWER see R. West's *Programming the PET/CBM* for the merge methods. The disk merge command was described in the *Transactor*, #8. But, by far, the easiest thing is to use the toolkit-type commands after typing NEW in the PET.

---

1. TOOLKIT from a Palo Alto I.C. has an APPEND command. TOOLKIT will append to nothing, ultimately relocating a Commodore 64 program to wherever you are in the PET.

---

2. BASIC Aid from Commodore has a MERGE command. It, too, should merge with nothing and relocate.

---

3. POWAID, available in the public domain, which is Brad Templeton's extension of his POWER chip, contains a MERGE command. MERGE"0:C64 PROGRAM" moves it exactly where you want it.

---

4. I'm sure other similar utilities on chip or in RAM will do the same thing.

There is a related issue; that of moving PET programs into the 64. I wrote several at $4000 (16384) on the PET and saved them via the machine language monitor from $4000 (it's a nice even number!). I thought the 64 would relocate correctly. Well, it did, but I botched the job. I ended up with a horrendous mess of crazy line numbers. The 64 moved the initial zero, of course. So the moral of this story is not to save the initial BASIC zero. In contrast to PET, a program in a PET partition at $4000 should be saved from $4001 if the intent is to move it to the 64. Of course using the LOAD"PET PROGRAM",8,1 does the trick on a $4000-type save if you can remember the ",1" part... **C**

Software Keyboard Conversion for Your Commodore 64

by Gregory Yob

Here you are, sitting at your new Commodore 64 computer, which is a shining example of a modern technical miracle. Yet, would you believe that your keyboard's arrangement is an anachronism? In this age of efficiency, when personal computers are being used as tools for personal productivity, it is a sad fact that the standard keyboard is set up to hinder your entry of data!

The Sholes Keyboard vs. the Dvorak Keyboard

The original typewriter had a few bugs in its design, one of which was a tendency for the keys to jam together if the typist struck them too rapidly. Mr. Sholes, the inventor of the typewriter, solved the problem by making the keyboard so difficult to use that the typist couldn't jam the machine. He did this by deliberately arranging the keys to force the typist to type slowly.

Since human beings are remarkably adaptable, the Sholes keyboard layout became the nearly universal standard keyboard—long after the mechanical problems of typewriters were solved and forgotten. As typewriters came into general use, several studies were made concerning the ease with which a typewriter keyboard could be used and how this depended on the arrangement of the letters. The Sholes keyboard turns out to be slower than a keyboard arranged A B C D E F and, in nearly every case, slower than a keyboard whose keys were arranged in random order!

If some thought is given to the strength of the hands, a look at the QWERTY keyboard (our friend Sholes here) reveals that the most common letters in English are mostly placed on the left side of the keyboard, and most of these are NOT in the "home position". (If you rest your hands on a keyboard in the approved manner your fingers will touch ASDF JKL;) Of the first ten most common letters, only two are in the "home position" and both are on the weakest two fingers of the left hand!

A different arrangement, known as the Dvorak keyboard, is arranged to take advantage of the hand's characteristics in typing. This layout puts the common letters in the "home position", with the vowels in the left hand and the consonants in the right. (Most words tend to alternate vowels and consonants with more consonants than vowels—so the strong right hand takes the load alternating with the left.) See Figure 1 for the Sholes and Dvorak keyboard layouts. Some studies indicate that a typist can type twenty times as rapidly on a Dvorak keyboard!

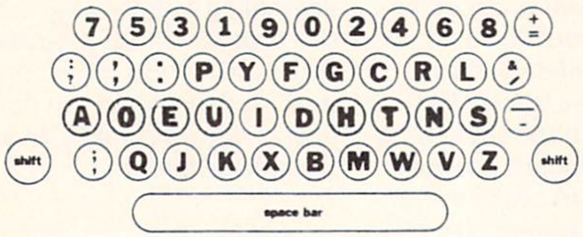
The Commodore 64 to the Rescue

The computers prior to the Commodore 64 had no easy way to rearrange the keys on the keyboard. Your choices were limited to redoing the ROMs or rewiring the keyboard. The Commodore 64 has an interesting feature which in effect will let you redo the ROM and thereby rearrange your keyboard. (By the way, this feature lets you make any re-arrangement you want—like the A B C D E F keyboard for a handicapped person for example.)

The Commodore 64 has a full 64K of RAM and 16K of ROM which shares the same address space (i.e., how do you fit 80K of memory into 64K of space?) If you take a look at pages 260 through

Figure 1. Sholes and Dvorak Keyboard Layouts

The diagram above shows the familiar Sholes keyboard layout. This is very similar to the one on your Commodore 64.



The Dvorak keyboard is shown above. This arrangement lets you type much more rapidly than the Sholes keyboard permits. Of course, you have to learn the new layout of the keys, which takes some time. If you convert the keyboard via the program, any commercial typing training program will work for learning the Dvorak keyboard.

---

267 in the *Commodore 64 Programmer's Reference Guide,* you'll see that the 6510 chip has a six-bit I/O port at location 1, with its data direction register in location 0. Bits 0,1 and 2 on this port (Bit 0 selects the BASIC ROM and is called Loram. Bit 1 selects the Kernal ROM and is called Hiram.), combined with two lines in the expansion port (game and Exrom) allow eight variations of the memory map to be set up. The normal memory map has ROM in effect in the areas of the Kernal and BASIC. If we want to, the Kernal and BASIC ROM can be switched off and the RAM in the same locations used instead.

Of course, if you go around switching ROM to RAM, you could get into some trouble! Most of the time your Commodore 64 is running a program in the Kernal or BASIC and if you turn either of these off, the machine will cease to function! If you want to check this out, a PEEK(1) reveals that the value 55 ($37 in hexadecimal) is the normal value—i.e., Loram and Hiram are both on and have the value of 1. Now try the other three combinations of Loram and Hiram—POKE 1,52 will set them both to off, POKE 1,53 leaves Loram on and Hiram off (i.e., BASIC as ROM and the Kernal replaced by RAM) and POKE 1,54 vice versa. Try these out—a small surprise awaits you!

A second feature of the Commodore 64 lets us actually change the keyboard from BASIC! A POKE to an area currently covered by ROM will write the value into the RAM anyway—so a simple loop to PEEK the current value in the ROM then POKE the same value into the RAM will copy the operating system (Kernal and BASIC) into the RAM. Once this is done, the keyboard decoding table, which lives in the Kernal area, can be modified for the Dvorak layout. The last step is to POKE location 1 to change from the Kernal ROM to the copy of the Kernal in RAM.

This ledgermain will now give you a Commodore 64 whose keyboard speaks Dvorak instead of Sholes. May your productivity shine!

On to the Nitty-Gritty

The program at the end of this article performs the conversion of your Commodore 64 keyboard from the Sholes to the Dvorak layout. Lines 10 to 60 serve only to protect my reputation. Lines 70 through 120 transfer a copy of the ROM to the RAM sitting "underneath" in the Commodore 64.

The Kernal is copied in Lines 90-100 and BASIC is moved in Lines 110-120. (By the way, there is no way to have BASIC in effect from ROM with the Kernal replaced by RAM. You must copy both of them, or the machine will simply do a warm start when you attempt the switchover from ROM. Though the description of Loram and Hiram would let you think otherwise, a chip called the PLA buried in the Commodore 64 has ideas of its own. So it is both BASIC and the Kernal or no dice.)

Line 130 changes to typewriter mode; that is, the character set is switched to lower case/upper case. The four strings LS$, US$, LD$, UD$ are set up in lines 170 through 440. LS for example, means "lower case Sholes" and you can discern the others from the remarks. By building the strings in four steps I am copying the keyboard layout. For example, line 220 shows the home row of the Sholes layout, which is what you have on your machine. When we get to the Dvorak strings LD$ and UD$, the Dvorak keyboard is similarly represented. See the similarity of these string assignments to Figure 1.

You can easily change LD$ and UD$ to represent the layout of your choice. To do the A B C D E F keyboard, lines 350 to 370 become:

```
350 LD$=LD$+"abcdefg
    hijk"
360 LD$=LD$+"lmnopqr
    stuv"
370 LD$=LD$+"wxyz?,
    .-/*"
```

Similarly, lines 420 and 440 can be set for the upper case version. Or, if you wish, exchange LD$ for UD$ to get the use of upper case without the shift key and lower case with shift. (Some of the old-timers will recall that the early PETs did this. I had that handicapped person in mind.)

If you peer closely at Figure 1, some differences between the Commodore 64's keyboard and the Sholes layout become apparent. This is particularly clear in the upper row and keys like + and *. Feel free to select the variations that suit you the best. The top row remains unchanged in shifted mode, since the Sholes layout isn't concerned with the punctuation above the number keys. Note that CHR$(34) is the quotation marks character in lines 270 and 410.

With the strings at the ready, the real work can begin. The keyboard table in the Commodore 64's Kernal resides in $EB81 to $EC43 (hexadecimal). Line 570 notes this as the variables KT and KE. Remember, when we PEEK, we see the ROM value and when we POKE, the RAM gets changed. If this weren't the case, the code used here would fail. (See if you can figure out why...) The loop C in line 580 picks out the characters from LS$ and LD$ one at a time and sets the ASC values of these in variables SK (Sholes key) and DK (Dvorak key) respectively. The loop K in line 610 searches through the keytable for a match for the Sholes key, and when it is found, line 620 performs the POKE of the Dvorak value (DK). Line 640 is a safety check for non-keyboard characters, which is never executed. (This line may not be needed now, but I wanted to know if I had made any mistakes when debugging the program.) When all this is done, line 650 tells me the conversion for one character is done.

Line 660 is a note about BASIC. Most of the time we get here still in the K loop, and a NEXT without the C would merely continue the K loop. But continuing the C loop is what we want.

Lines 680 through 790 do the same thing for the strings US$ and UD$ for the upper case keys. The values of KT and KE could be changed by only looking at the upper case part of the table, but I believe in letting the computer do my dirty work.

The last item is the POKE in line 840 which switches Hiram over to RAM. Remember the PLA also switches BASIC over as well. POKE 1,52 will also do. (*But* I haven't tested it!) You now have a Dvorak keyboard on your Commodore 64.

A Final Note

The method of moving the Kernal and BASIC into RAM has many other applications beyond changing the keyboard. Additional BASIC commands for sound or graphics could be added without using a "wedge" program; this isn't easy without the source code for BASIC and the Kernal, so we will have to hope Commodore will provide these eventually. Meanwhile, happy typing! **C**

(program listing on next page)

Keyboard Conversion Program for the Commodore 64

```
10 rem c-64 dvorak keyboard program
20 rem written by gregory yob
30 rem you may copy this program
40 rem if you don't remove these
50 rem remarks. >> thank you <<
60 rem
70 print"[clear]transferring rom
   to ram"
80 print"    -- be patient --"
90 for j=14*4096 to 16*4096-1
100 b=peek(j):poke j,b:next
110 for j=10*4096 to 12*4096-1
120 b=peek(j):poke j,b:next
130 print chr$(14)
140 rem
150 rem set strings for keyboard
160 rem representations
170 rem
180 rem lower case sholes keys
190 rem done row by row
200 ls$="234567890+-"
210 ls$=ls$+"qwertyuiop@"
220 ls$=ls$+"asdfghjkl:;"
230 ls$=ls$+"zxcvbnm,./"
240 rem
250 rem upper case sholes keys
260 rem done row by row
270 us$=chr$(34)+"#$%&'()0+|"
280 us$=us$+"QWERTYUIOP✓"
290 us$=us$+"ASDFGHJKL[]"
300 us$=us$+"ZXCVBNM<>?"
310 rem
320 rem lower case dvorak keys
330 rem done row by row
340 ld$="7531902468="
350 ld$=ld$+"?,.pyfgcrl/"
360 ld$=ld$+"aoeuidhtns-"
370 ld$=ld$+",qjkxbmwvz"
380 rem
390 rem upper case dvorak keys
400 rem done row by row
410 ud$=chr$(34)+"#$%&'()0+|"
420 ud$=ud$+"AEOUIDHTNS-"
430 ud$=ud$+"AOEUIDHTNS-"
440 ud$=ud$+";QJKXBMWVZ"
450 print
460 print"converting lower case to dvorak"
470 print
480 rem we scan through the rom table
490 rem which stores the keyword in
500 rem the order of its switch matrix
510 rem values for the key we want.
520 rem then we just poke in the dvorak
530 rem key value instead
540 rem
550 rem keytable boundaries
560 rem
570 kt=60289:ke=60483
580 for c=1 to len(ls$)
590 sk=asc(mid$(ls$,c,1))
600 dk=asc(mid$(ld$,c,1))
610 for k=kt to ke
620 if peek(k)=sk then poke k,dk:goto 650
630 next
640 print"<<<keytable error>>>":stop
650 print"sholes: "chr$(sk)" dvorak: "chr$(dk)
660 rem 'c' is required in next
670 next c:print
680 print"converting upper case to dvorak"
690 print
700 for c=1 to len(us$)
710 sk=asc(mid$(us$,c,1))
720 dk=asc(mid$(ud$,c,1))
730 for k=kt to ke
740 if peek(k)=sk then poke k,dk:goto 770
750 next
760 print"<<<keytable error>>>":stop
770 print"sholes: "chr$(sk)" dvorak: "chr$(dk)
780 rem 'c' is required in next
790 next c:print
800 print
810 print"dvorak keyboard is now installed"
820 print:print" ... have fun ..."
830 rem change over to ram
840 poke 1,53
```

House Inventory for the Commodore 64

by Robert W. Baker

This program provides an easy means of maintaining an inventory of personal possessions for insurance or other related purposes. Information is stored on floppy disk for later retrieval and easy storage, such as in safety deposit boxes.

Running the program is quite simple; to create a new data file simply select that mode and answer the questions concerning the item description, make, model, serial number or other identifying markings, date acquired, and original value. Typing RETURN for any question will automatically enter a question mark for that entry. When all questions are entered, the entire entry will be displayed and you will be asked if it is correct before it is actually written in the data file.

Typing "D" (for DONE) for any entry will abort that entire item entry, close the output file, and return to the program command mode. Typing "E" (for ERROR) will indicate an error and will abort the entire item entry and restart it again with the first question. Be careful when entering new items into the data file, do not use commas or colons to separate words within an entry since BASIC thinks you may be entering more than one string. Use dashes or some other graphic character and play it safe. Avoid using quotes for similar reasons.

To read an already created data file, insert the disk and select that program mode. Three items will be displayed at a time with all information. Hitting any key except "D" will display the next three entries. Typing "D" will terminate the read mode, close the input file, and return to the program command mode.

Other program modes are provided to copy or edit the data files produced by this program. The edit mode allows copying or deleting individual entries. You can insert new items at any point. Also, a search feature is included to copy all items till a specific item is found.

All program modes provide file and/or drive selection for ease of use. A default file name of INVENTORY DATA will be generated unless you enter a specific file name. If you should have a large number of items to catalog you may want to use separate data files for each room, for items acquired each year, specific collections, etc. Program use should be self-evident through prompting instructions displayed by the program. At present, the program does not provide a print option since it was designed for storage of large amounts of personal data. **C**

```
10 REM ** HOUSE INVENTORY *** DISK **
20 REM
30 REM          ROBERT W. BAKER
40 REM 15 WINDSOR DR., ATCO, NJ 08004
50 REM
60 REM ******************************
70 :
80 PRINT"[CLEAR,SPACE5]HOUSEHOLD INVENTORY PROGRAM": GOSUB 1290
```

Inventory Program

```
90 PRINT"DESIRED PROGRAM MODE:": PRINT: PRINT" 0 = DONE"
100 PRINT" 1 = READ DATA"
110 PRINT" 2 = WRITE NEW DATA FILE": PRINT" 3 = COPY DATA FILE"
120 PRINT" 4 = EDIT DATA FILE": PRINT" 5 = HELP (INFORMATION)"
130 GOSUB 1290: PRINT: PRINT"MODE ?";
140 GOSUB 1360: IF R$="0" THEN PRINT"[CLEAR]": END
150 R=VAL(R$): IF R<1 OR R>5 THEN 140
160 IF Z<5 THEN OPEN 15,8,15
170 Z=R: ON R GOTO 310,180,310,310,1400
180 GOSUB 1250
190 INPUT"[DOWN]OUTPUT TO DISK DRIVE# (0 OR 1)   0[LEFT3]";T$:
    T$=LEFT$(T$,1)
210 T=VAL(T$): IF T$<>"0" AND T$<>"1" THEN 80
220 PRINT: PRINT"OUTPUT ";: GOSUB 1340
230 IF F$<>"-" THEN 260
240 F$="INVENTORY DATA"
250 PRINT: PRINT"DEFAULT FILE = ";T$;":";F$
260 OPEN 2,8,5,T$+":"+F$+",S,W": GOSUB 1600
270 IF Z=3 THEN 560
280 IF Z=4 THEN 610
290 GOSUB 900: IF C THEN GOSUB 1130: GOTO 290
300 GOTO 550
310 GOSUB 1250
320 INPUT"[DOWN]INPUT FROM DISK DRIVE# (0 OR 1)   0[LEFT3]";T$
330 T=VAL(T$): T$=LEFT$(T$,1)
340 IF T$<>"0" AND T$<>"1" THEN 80
350 PRINT: PRINT"INPUT ";: GOSUB 1340
360 IF F$="-" THEN F$="INVENTORY DATA": PRINT"[DOWN]DEFAULT FILE =
    ";T$;":";F$
370 OPEN 1,8,6,T$+":"+F$+",S,R": GOSUB 1600
380 X$=""
390 IF Z>2 THEN 190
400 GOSUB 1160: IF C>1 THEN 490
410 GOSUB 1090: IF C THEN 510
420 GOSUB 1160: IF C>1 THEN 510
430 GOSUB 1100: IF C THEN 510
440 GOSUB 1160: IF C>1 THEN 510
450 GOSUB 1100: IF C THEN 510
460 GOSUB 1300
470 GOSUB 1380: IF R$<>"D" THEN 400
480 GOTO 550
490 PRINT"[CLEAR,RVS]END OF MODE #1[RVOFF,SPACE2]DONE READING
    DATA FILE" : PRINT
510 GOSUB 1300
520 IF C=1 THEN PRINT"END OF DATA FILE!"
```

```
530 IF C>1 THEN PRINT"DISK ERROR ( STATUS =";ST;")"
540 GOSUB 1350
550 CLOSE 1: CLOSE 2: CLOSE 15: GOTO 80
560 I9$="": GOSUB 1250: PRINT"[RVS]PLEASE WAIT[RVOFF,SPACE2]
   ******  COPYING DATA FILE![DOWN]"
570 GOSUB 1160:IF C>1 THEN 820
580 IF Z=4 AND LEFT$(I$,LEN(I9$))=I9$ THEN GOSUB 1250: GOTO 620
590 GOSUB 1130: IF C=1 THEN 820
600 IF Z=3 OR I9$<>"" THEN 570
610 GOSUB 1160: IF C>1 THEN 820
620 GOSUB 1250: GOSUB 1100: GOSUB 1290: PRINT"DESIRED ACTION:": PRINT
630 PRINT"  1 = COPY THIS ITEM, NO CHANGE"
640 PRINT"  2 = DELETE THIS ITEM"
650 PRINT"  3 = INSERT ITEMS BEFORE THIS ONE"
660 PRINT"  4 = SEARCH & COPY TILL ITEM FOUND": PRINT
670 PRINT"ACTION ? ";
680 GOSUB 1360: R=VAL(R$): IF R<1 OR R>4 THEN 680
690 PRINT R$
700 PRINT"OK": I9$="": ON R GOTO 590,710,730,760
710 IF C=1 THEN 820
720 GOTO 610
730 I9$=I$: W9$=W$: M9$=M$: S9$=S$: D9$=D$: V9$=V$: C9=C
740 GOSUB 900: IF C THEN GOSUB 1130: GOTO 740
750 I$=I9$: W$=W9$: M$=M9$: S$=S9$: D$=D9$: V$=V9$: C=C9: GOTO 620
760 GOSUB 1250: PRINT"ALL ENTRIES WILL BE COPIED UNTILL"
770 PRINT: PRINT"DESIRED ITEM IS FOUND;"
780 PRINT: PRINT: PRINT"ENTER ITEM TO SEARCH FOR:"
790 INPUT"   -[LEFT3]";I9$
800 IF I9$="-" THEN I9$="": PRINT"[DOWN3]SEARCH ABORTED": GOTO 620
810 PRINT: PRINT: PRINT: PRINT"SEARCHING": GOTO 580
820 IF Z=3 THEN 520
830 GOSUB 1250: IF C>1 THEN 530
840 PRINT"END OF INPUT FILE!"
850 PRINT: PRINT"DO YOU WANT TO ADD ANY ENTRIES TO THE"
860 PRINT: PRINT"END OF THE DATA FILE";
870 GOSUB 1310: IF R$="N" THEN 550
880 GOSUB 900: IF C THEN GOSUB 1130: GOTO 880
890 GOTO 550
900 C=0: PRINT"[CLEAR]ENTER ITEM INFORMATION:[DOWN]"
   : PRINT"D = DONE ENTERING DATA"
910 PRINT"E = ERROR, RESTART ENTIRE ITEM"
920 PRINT: PRINT"DO NOT USE ',' OR ':' WITHIN THE DATA"
930 PRINT: PRINT"PRESS [RVS]RETURN[RVOFF,SPACE]AFTER EACH ENTRY"
940 GOSUB 1290: INPUT"[RVS]ITEM[RVOFF,SPACE3]?[LEFT3]";I$: IF I$="E"
    THEN 900
```

```
950 IF I$="D" THEN RETURN
960 INPUT"[RVS]MAKE[RVOFF,SPACE3]?[LEFT3]";W$: IF W$="E" THEN 900
970 IF W$="D" THEN RETURN
980 INPUT"[RVS]MODEL[RVOFF,SPACE3]?[LEFT3]";M$: IF M$="E" THEN 900
990 IF M$="D" THEN RETURN
1000 INPUT"[RVS]SERIAL#/ID[RVOFF,SPACE3]?[LEFT3]";S$: IF S$="E"
     THEN 900
1010 IF S$="D" THEN RETURN
1020 INPUT"[RVS]DATE ACQ'D[RVOFF,SPACE](MONTH/DAY/YEAR)
     ?[LEFT3]";D$ : IF D$="E" THEN 900
1030 D$=LEFT$(D$,8): IF D$="D" THEN RETURN
1040 INPUT"[RVS]$VALUE[RVOFF,SPACE3]?[LEFT3]";V$: IF V$="E" THEN 900
1050 IF V$="D" THEN RETURN
1060 GOSUB 1090: GOSUB 1290
1070 PRINT"IS THIS ENTRY CORRECT";: GOSUB 1310: IF R$="N" THEN 900
1080 C=1: RETURN
1090 PRINT"[CLEAR]";
1100 PRINT"[RVS]ITEM:[RVOFF,SPACE]";I$: PRINT"[RVS]MAKE:[RVOFF,SPACE]
     ";W$ : PRINT"[RVS]MODEL:[RVOFF,SPACE]";M$
1110 PRINT"[RVS]SERIAL#/ID:[RVOFF,SPACE]";S$
1120 PRINT"[RVS]DATE ACQ'D:[RVOFF,SPACE]"D$;TAB(22);"[RVS]VALUE
    :[RVOFF,SPACE]$";V$: PRINT: RETURN
1130 X$=I$: GOSUB 1150: X$=W$: GOSUB 1150: X$=M$: GOSUB 1150
1140 X$=S$: GOSUB 1150: X$=D$: GOSUB 1150: X$=V$
1150 PRINT#2,X$;CHR$(13);: GOTO 1600
1160 GOSUB 1230: I$=X$: IF C THEN RETURN
1170 GOSUB 1230: W$=X$: IF C THEN RETURN
1180 GOSUB 1230: M$=X$: IF C THEN RETURN
1190 GOSUB 1230: S$=X$: IF C THEN RETURN
1200 GOSUB 1230: D$=X$: IF C THEN RETURN
1210 GOSUB 1230: V$=X$: IF C=2 THEN C=1
1220 RETURN
1230 C=0: INPUT#1,X$: IF ST THEN C=3: IF ST=64 THEN C=2
1240 GOTO 1600
1250 IF Z=1 THEN PRINT"[CLEAR,RVS]MODE #1[RVOFF,SPACE2]READ DATA FILE"
1260 IF Z=2 THEN PRINT"[CLEAR,RVS]MODE #2[RVOFF,SPACE2]WRITE NEW DATA
     FILE"
1270 IF Z=3 THEN PRINT"[CLEAR,RVS]MODE #3[RVOFF,SPACE2]COPY DATA FILE"
1280 IF Z=4 THEN PRINT"[CLEAR,RVS]MODE #4[RVOFF,SPACE2]EDIT DATA FILE"
1290 PRINT
1300 PRINT"---------------------------------------": PRINT: RETURN
1310 PRINT" (Y/N) ? ";
1320 GOSUB 1360: IF R$<>"Y" AND R$<>"N" THEN 1320
1330 PRINT R$: RETURN
1340 INPUT"FILENAME    -[LEFT3]";F$: RETURN
```

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

user groups

User Group Listing

ALABAMA

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

ALASKA

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

ARIZONA

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

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

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

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

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

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

ARKANSAS

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

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

CALIFORNIA

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

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

Valley Computer Club
2006 Magnolia Blvd.
Burbank, CA
(213) 849-4094
1st Wed. 6 p.m.

Valley Computer Club
1913 Booth Road
Ceres, CA 95307

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

COLORADO

VICKIMPET Users Group
4 Waring Lane, Greenwood Village
Littleton, CO 80121
Contact: Louis Roehrs

Colorado Commodore Computer Club
2187 S. Golden Ct.
Denver, CO 80227
986-0577
Jack Moss
Meet: 2nd Wed.

CONNECTICUT

John F. Garbarino
Skiff Lane Masons Island
Mystic, CT 06355
(203) 536-9789

Commodore User Club
Wethersfield High School
411 Wolcott Hill Road
Wethersfield, CT 06109
Contact: Daniel G. Spaneas

VIC Users Club
c/o Edward Barszczewski
22 Tunxis Road
West Hartford, CT 06107

New London County Commodore Club
Doolittle Road
Preston, CT 06360
Contact: Dr. Walter Doolittle

FLORIDA

Jacksonville Area PET Society
401 Monument Road, #177
Jacksonville, FL 32211

Richard Prestien
6278 SW 14th Street
Miami, FL 33144

South Florida PET Users Group
Dave Young
7170 S.W. 11th
West Hollywood, FL 33023
(305) 987-6982

VIC Users Club
c/o Ray Thigpen
4071 Edgewater Drive
Orlando, FL 32804

PETs and Friends
129 NE 44 St.
Miami, FL 33137
Richard Plumer

Sun Coast VICs
P.O. Box 1042
Indian Rocks Beach, FL 33535
Mark Weddell

Bay Commodore Users Group
c/o Gulf Coast Computer Exchange
241 N. Tyndall Pkwy.
P.O. Box 6215
Panama City, FL 32401
(904) 785-6441
Richard Scofield

Gainesville Commodore Users Club
3604-20A SW 31st Dr.
Gainesville, FL 32608
Louis Wallace

64 Users Group
P.O. Box 561689
Miami, FL 33156
(305) 274-3501
Eydie Sloane

Brandon Users Group
108 Anglewood Dr.
Brandon, FL 33511
(813) 685-5138
Paul Daugherty

Commodore 64/VIC 20 User Group
Martin Marietta Aerospace
P.O. Box 5837, MP 142
Orlando, FL 32855
(305) 352-3252/2266
Mr. Earl Preston

Brandon Commodore Users Group
414 E. Lumsden Rd.
Brandon, FL 33511

Gainesville Commodore Users Group
Santa Fe Community College
Gainesville, FL 32602
James E. Birdsell

Commodore Computer Club
P.O. Box 21138
St. Petersburg, FL 33742

Commodore Users Group
545 E. Park Ave.
Apt. #2
Tallahassee, FL 32301
(904) 224-6286
Jim Neill

The Commodore Connection
P.O. Box 6684
West Palm Beach, FL 33405

GEORGIA

VIC Educators Users Group
Cherokee County Schools
110 Academy St.
Canton, GA 30114
Dr. Al Evans

Bldg. 68, FLETC
Glynco, GA 31524
Richard L. Young

VIC-tims
P.O. Box 467052
Atlanta, GA 30346
(404) 922-7088
Eric Ellison

HAWAII

Commodore Users Group of Honolulu
c/o PSH
824 Bannister St.
Honolulu, HI
(808) 848-2088
3rd Fri. every month

IDAHO

GHS Computer Club
c/o Grangeville High School
910 S. D St.
Grangeville, ID 83530
Don Kissinger

S.R.H.S. Computer Club
c/o Salmon River H.S.
Riggins, ID 83549
Barney Foster

Commodore Users
548 E. Center
Pocatello, ID 83201
(208) 233-0670
Leroy Jones

Eagle Rock Commodore Users Group
900 S. Emerson
Idaho Falls, ID 83401
Nancy J. Picker

ILLINOIS

Shelly Wernikoff
2731 N. Milwaukee
Avenue
Chicago, IL 60647

VIC 20/64 Users Support
Group
c/o David R. Tarvin
114 S. Clark Street
Pana, IL 62557
(217) 562-4568

Central Illinois PET User
Group
635 Maple
Mt. Zion, IL 62549
(217) 864-5320
Contact: Jim Oldfield

ASM/TED User Group
200 S. Century
Rantoul, IL 61866
(217) 893-4577
Contact: Brant Anderson

PET VIC Club (PVC)
40 S. Lincoln
Mundelein, IL 60060
Contact: Paul Schmidt,
President

Rockford Area PET Users
Group
1608 Benton Street
Rockford, IL 61107

Commodore Users Club
1707 East Main St.
Olney, IL 62450
Contact: David E. Lawless

VIC Chicago Club
3822 N. Bell Ave.
Chicago, IL 60618
John L. Rosengarten

Chicago Commodore 64
Users & Exchange Group
P.O. Box 14233
Chicago, IL 60614
Jim Robinson

Fox Valley PET Users
Group
833 Willow St.
Lake in the Hills, IL 60102
(312) 658-7321
Art DeKneef

The Commodore 64 Users
Group
P.O. Box 572
Glen Ellyn, IL 60137
(312) 790-4320
Gus Pagnotta

Oak Lawn Commodore Users Group
The Computer Store
11004 S. Cicero Ave.
Oak Lawn, IL 60453
(312) 499-1300
Bob Hughes

The Kankakee Hackers
RR #1, Box 279
St. Anne, IL 60964
(815) 933-4407
Rich Westerman

INDIANA

PET/64 Users
10136 E. 96th St.
Indianapolis, IN 46256
(317) 842-6353
Jerry Brinson

Cardinal Sales
6225 Coffman Road
Indianapolis, IN 46268
(317) 298-9650
Contact: Carol Wheeler

CHUG (Commodore
Hardware Users Group)
12104 Meadow Lane
Oaklandon, IN 46236
Contact: Ted Powell

VIC Indy Club
P.O. Box 11543
Indianapolis, IN 46201
(317) 898-8023
Ken Ralston

Northern Indiana
Commodore Enthusiasts
927 S. 26th St.
South Bend, IN 46615
Eric R. Bean

Commodore Users Group
1020 Michigan Ave.
Logansport, IN 46947
(219) 722-5205
Mark Bender

Computer Workshop VIC 20/64 Club
282 S. 600 W.
Hebron, IN 46341
(219) 988-4535
Mary O'Bringer

The National Science Clubs of America
Commodore Users Division
7704 Taft St.
Merrillville, IN 46410
Brian Lapley or Tom Vlasic

East Central Indiana VIC User Group
Rural Route #2
Portland, IN 47371
Stephen Erwin

National VIC 20 Program Exchange
102 Hickory Court
Portland, IN 47371
(219) 726-4202
Stephen Erwin

IOWA

Commodore User Group
114 8th St.
Ames, IA 50010

Quad City Commodore Club
1721 Grant St.
Bettendorf, IA 52722
(319) 355-2641
John Yigas

Commodore Users Group
965 2nd St.
Marion, IA 52302
(319) 377-5506
Vern Rotert
3rd Sun. of month

Siouxland Commodore Club
2700 Sheridan St.
Sioux City, IA 51104
(712) 258-7903
Gary Johnson
1st & 3rd Monday of month

421 W. 6th St.
Waterloo, IA 50702
(319) 232-1062
Frederick Volker

Commodore Computer Users
Group of Iowa
Box 3140
Des Moines, IA 50316
(515) 263-0963 or (515) 287-1378
Laura Miller

KANSAS

Wichita Area PET
Users Group
2231 Bullinger
Wichita, KS 67204
(316) 838-0518
Contact: Mel Zandler

Kansas Commodore
Computer Club
101 S. Burch
Olathe, KS 66061
Contact: Paul B. Howard

Commodore Users Group
6050 S. 183 St. West
Viola, KS 67149
Walter Lounsbery

KENTUCKY

VIC Connection
1010 S. Elm
Henderson, KY 42420
Jim Kemp

LOUISIANA

Franklin Parish Computer
Club
#3 Fair Ave.
Winnisboro, LA 71295
James D. Mays, Sr.

NOVA
917 Gordon St.
New Orleans, LA 70117
(504) 948-7643
Kenneth McGruder, Sr.

VIC 20 Users Group
5064 Bowdon St.
Marrero, LA 70072
(504) 341-5305
Wayne D. Lowery, R.N.

MARYLAND

Assoc. of Personal
Computer Users
5014 Rodman Road
Bethesda, MD 20016

Blue TUSK
700 East Joppa Road
Baltimore, MD 21204
Contact: Jim Hauff

House of Commodore
8835 Satyr Hill Road
Baltimore, MD 21234
Contact: Ernest J. Fischer

Long Lines Computer Club
323 N. Charles St., Rm. 201
Baltimore, MD 21201
Gene Moff

VIC & 64 Users Group
The Boyds Connection
21000 Clarksburg Rd.
Boyds, MD 20841
(301) 428-3174
Tom DeReggi

VIC 20 Users Group
23 Coventry Lane
Hagerstown, MD 21740
Joseph Rutkowski

Hagerstown Users Group
1201-B Marshall St.
Hagerstown, MD 21740
(301) 790-0968
Greg Stewart
1st & 3rd Friday of month 6:30 p.m.

Rockville VIC/64 Users Group
13013 Evanstown St.
Rockville, MD 20853
(301) 946-1564
Meryle or Tom Pounds

The Compucats' Commodore
Computer Club
680 W. Bel Air Ave.
Aberdeen, MD 21001
(301) 272-0472
Betty Jane Schueler

MASSACHUSETTS

Eastern Massachusetts
VIC Users Group
c/o Frank Ordway
7 Flagg Road
Marlboro, MA 02173

VIC Users Group
c/o Ilene Hoffman-Sholar
193 Garden St.
Needham, MA 02192

Commodore Users Club
Stoughton High School
Stoughton, MA 02072
Contact: Mike Lennon

user groups

Berkshire PET Lovers
CBM Users Group
Taconic High
Pittsfield, MA 01201

The Boston Computer Society
Three Center Plaza
Boston, MA 02108
(617) 367-8080
Mary E. McCann

VIC Interface Club
c/o Procter & Gamble Inst. Shop
780 Washington St.
Quincy, MA 02169
C. Gary Hall

Masspet Commodore Users Group
P.O. Box 307
East Taunton, MA 02718
David Rogers

Raytheon Commodore Users Group
Raytheon Company
Hartwell Rd. GRA-6
Bedford, MA 01730
John Rudy

Commodore 64 Users Group of The Berkshires
184 Highland Ave.
Pittsfield, MA 01201
Ed Rucinski

MICHIGAN

David Liem
14361 Warwick Street
Detroit, MI 48223

VIC Users Club
University of Michigan
School of Public Health
Ann Arbor, MI 48109
Contact: John Gannon

Commodore User Club
32303 Columbus Drive
Warren, MI 48093
Contact: Robert Steinbrecher

Commodore Users Group
c/o Family Computer
3947 W. 12 Mile Rd.
Berkley, MI 48072

W. Michigan VIC 20-64 Users
1311 Portland NE
Grand Rapids, MI 49505
(616) 459-7578
Jim D'Haem

VIC for Business
6027 Orchard Ct.
Lansing, MI 48910
Mike Marotta

South Computer Club
South Jr. High School
45201 Owen
Belleville, MI 48111
Ronald Ruppert

Commodore Users Group
c/o Eaton Rapids Medical Clinic
101 Spicerville Hwy.
Eaton Rapids, MI 48827
Albert Meinke III, M.D.

South East Michigan Pet Users Group
Box 214
Farmington, MI 48024
Norm Eisenberg

Commodore Computer Club
4106 Eastman Rd.
Midland, MI 48640
(517) 835-5130
John Walley
9:30 p.m. Sept/May

VIC, 64, PET Users Group
8439 Arlis Rd.
Union Lake, MI 48085
363-8539
Bert Searing

VIC Commodore User Club
486 Michigan Ave.
Mariesville, MI 48040
(313) 364-6804
M. Gauthier

MINNESOTA

MUPET (Minnesota Users of PET)
P.O. Box 179
Annandale, MN 55302
c/o Jon T. Minerich

Twin Cities Commodore Computer Club
6623 Ives Lane
Maple Grove, MN 55369
(612) 424-2425
Contact: Rollie Schmidt

MISSOURI

KCPUG
5214 Blue Ridge Boulevard
Kansas City, MO 64133
Contact: Rick West
(816) 356-2382

PET SET Club of St. Louis
633 Bent Oak Drive
Lake St. Louis, MO 63367
(314) 625-2701 or 625-4576
Tony Ott

VIC INFONET
P.O. Box 1069
Branson, MO 65616
(417) 334-6099
Jory Sherman

Worth County PET Users Group
Grant City, MO
(816) 564-3551
David Hardy

Mid-Missouri Commodore Club
1804 Vandiver Dr.
Columbia, MO 65201
(314) 474-4511
Phil Bishop

MONTANA

Powder River Computer Club
Powder River County High School
Broadus, MT 59317
Contact: Jim Sampson

Commodore User Club
1109 West Broadway
Butte, MT 59701
Contact: Mike McCarthy

NEVADA

Las Vegas PET Users
Suite 5-315
5130 E. Charleston Blvd.
Las Vegas, NV 89122
Gerald Hasty

NEW JERSEY

Amateur Computer Group
18 Alpine Drive
Wayne, NJ 07470

Somerset Users Club
49 Marcy Street
Somerset, NJ 08873
Contact: Robert Holzer

Educators Advisory
P.O. Box 186
Medford, NJ 08055
(609) 953-1200
John Handfield

VIC-TIMES
46 Wayne Street
Edison, NJ 08817
Thomas R. Molnar

VIC 20 User Group
67 Distler Ave.
W. Caldwell, NJ 07006
(201) 284-2281
G. M. Amin

VIC Software Development Club
77 Fomalhaut Ave.
Sewell, NJ 08080
H. P. Rosenberg

ACGNJ PET/VIC/CBM User Group
30 Riverview Terr.
Belle Mead, NJ 08502
(201) 359-3862
J. M. Pylka

South Jersey Commodore Computer Users Club
46-B Monroe Park
Maple Shade, NJ 08052
(609) 667-9758
Mark Orthner
2nd Fri. of month

NEW HAMPSHIRE

Northern New England Computer Society
P.O. Box 69
Berlin, NH 03570

TBH VIC-NICs
P.O. Box 981
Salem, NH 03079

NEW MEXICO

Commodore Users Group
6212 Karlson, NE
Albuquerque, NM 87113
(505) 821-5812
Danny Byrne

NEW YORK

Capital District 64/VIC 20 Users Group
363 Hamilton St.
Albany, NY 12210
(518) 436-1190
Bill Pizer

Long Island PET Society
Ralph Bressler
Harborfields HS
Taylor Avenue
Greenlawn, NY 11740

PET User Club of Westchester
P.O. Box 1280
White Plains, NY 10602
Contact: Ben Meyer

LIVE (Long Island VIC Enthusiasts)
17 Picadilly Road
Great Neck, NY 11023
Contact: Arnold Friedman

Commodore Masters
25 Croton Ave.
Staten Island, NY 10301
Contact: Stephen Farkouh

VIC Users Club
76 Radford St.
Staten Island, NY 10314
Contact: Michael Frantz

Rockland County Commodore Users Group
c/o Ross Garber
14 Hillside Court
Suffern, NY 10901
(914) 354-7439

West Chester County VIC Users Group
P.O. Box 146
Pelham, NY 10552
Joe Brown

SPUG
4782 Boston Post Rd.
Pelham, NY 10803
Paul Skipski

VIC 20 User Club
151-28 22nd Ave.
Whitestone, NY 11357
Jean F. Coppola

VIC 20 User Club
339 Park Ave.
Babylon, NY 11702
(516) 669-9126
Gary Overman

VIC User Group
1250 Ocean Ave.
Brooklyn, NY 11230
(212) 859-3030
Dr. Levitt

L&M Computer Club
VIC 20 & 64
4 Clinton St.
Tully, NY 13159
(315) 696-8904
Dick Mickelson

Commodore Users Group
1 Corwin Pl.
Lake Katrine, NY 12449
J. Richard Wright

VIC 20/Commodore 64 Users Group
31 Maple Dr.
Lindenhurst, NY 11757
(516) 957-1512
Pete Lobol

VIC Information Exchange Club
336 W. 23 St.
Deer Park, NY 11729
Tom Schlegel
SASE & phone please

New York Commodore Users Group
380 Riverside Dr., 7Q
New York, NY 10025
(212) 566-6250
Ben Tunkelang

Parsippany Computer Group
51 Ferncliff Rd.
Morris Plains, NJ 07950
(201) 267-5231
Bob Searing

Hudson Valley Commodore Club
1 Manor Dr.
Woodstock, NY 12498
F.S. Goh
1st Wednesday of month

LIVICS (Long Island VIC Society)
20 Spyglass Lane
East Setauket, NY 11733
(516) 751-7844
Lawrence Stefani

VIC Users Group
c/o Stoney Brook Learning Center
1424 Stoney Brook Rd.

Stoney Brook, NY 11790
(516) 751-1719
Robert Wurtzel

Poughkeepsie VIC User Group
2 Brooklands Farm Rd.
Poughkeepsie, NY 12601
(914) 462-4518
Joe Steinman

VIC 20 User Group
Paper Service Division
Kodak Park
Rochester, NY 14617
David Upham, Sr.

NORTH CAROLINA

Amateur Radio PET Users Group
P.O. Box 30694
Raleigh, NC 27622
Contact: Hank Roth

VIC Users Club
c/o David C. Fonenberry
Route 3, Box 351
Lincolnton, NC 28092

Microcomputer Users Club
Box 17142 Bethabara Sta.
Winston-Salem, NC 27116
Joel D. Brown

VIC Users Club
Rt. 11, Box 686
Hickory, NC 28601
Tim Gromlovits

OHIO

Dayton Area PET
User Group
933 Livingston Drive
Xenia, OH 45385
B. Worby, President
(513) 848-2065
J. Watson, Secretary
(513) 372-2052

Central Ohio PET
Users Group
107 S. Westmoor Avenue
Columbus, OH 43204
(614) 274-6451
Contact: Philip H. Lynch

Commodore Computer Club of Toledo
734 Donna Drive
Temperance, MI 48182
Gerald Carter

Chillicothe Commodore
Users Group
P.O. Box 211
Chillicothe, OH 45601
William A. Chaney

Licking County 64 Users Group
323 Schuler St.
Newark, OH 43055
(614) 345-1327

11433 Pearl Rd.
Strongsville, OH 44136
Paul M. Warner

OKLAHOMA

Southwest Oklahoma
Computer Club
P.O. Box 6646
Lawton, OK 73504
Garry Lee Crowell
1:30 1st Sunday at
Lawton City Library

Tulsa Area Commodore Users Group
Tulsa Computer Society
P.O. Box 15238
Tulsa, OK 74112
Annette Hinshaw

Commodore Oklahoma Users Club
4000 NW 14th St.
Oklahoma City, OK 73107
(405) 943-1370
Stanley B. Dow

Commodore Users
Box 268
Oklahoma City, OK 73101
Monte Maker, President

OREGON

NW PET Users Group
John F. Jones
2134 N.E. 45th Avenue
Portland, OR 97213

PENNSYLVANIA

PET User Group
Gene Beals
P.O. Box 371
Montgomeryville, PA 18936

Penn Conference Computer Club
c/o Penn Conference of SDA
720 Museum Road
Reading, PA 19611
Contact: Dan R. Knepp

PACS PET Users Group
20th & Olney Sts.
Philadelphia, PA 19141
(215) 951-1258
Stephen Longo

Glen Schwartz
807 Avon
Philadelphia, PA 19116

Gene Planchak
4820 Anne Lane
Sharpsville, PA 15150
(412) 962-9682

PPG (Pittsburgh PET Group)
c/o Joel A. Casar, DMD
2015 Garrick Drive
Pittsburgh, PA 15235
(412) 371-2882

Westmoreland Commodore
Users Club
c/o DJ & Son Electronics
Colonial Plaza
Latrobe, PA 15650
Jim Mathers

COMPSTARS
440 Manatawny St.
Pottstown, PA 19464
Larry Shupinski, Jr.
Meet at Audio Video
Junction

Commodore Users Club
3021 Ben Venue Dr.
Greensburg, PA 15601
(412) 836-2224
Jim Mathers

VIC 20 Programmers, Inc.
c/o Watson Woods
115 Old Spring Rd.
Coatesville, PA 19320
Robert Gougher

G.R.C. User Club
300 Whitten Hollow Rd.
New Kensington, PA 15068
Bill Bolt

NADC Commodore Users Club
248 Oakdale Ave.
Horsham, PA 19044
Norman McCrary

CACC (Capitol Area Commodore
Club)
134 College Hill Rd.
Enola, PA 17025
(717) 732-2123
Lewis Buttery
Union Deposit Mall at 7 p.m.

G/C Computer Owners Group
P.O. Box 1498
Reading, PA 19607
Jo Lambert
(215) 775-2600, ex 6472

Boeing Employees Personal
Computer Club
The Boeing Vertol Co.
P.O. Box 16858
Philadelphia, PA 19142
(215) 522-2257
Jim McLaughlin

PUERTO RICO

CUG of Puerto Rico
RFD #1, Box 13
San Juan, PR 00914
Ken Burch

VIC 20 User Group
655 Hernandez St.
Miramar, PR 00907
Robert Morales, Jr.

RHODE ISLAND

Irving B. Silverman, CPA
160 Taunton Ave.
E. Providence, RI 02914
Contact: Michelle Chavanne

Newport VIC/64 Users
10 Maitland Ct.
Newport, RI 02840
(401) 849-2684
Dr. Matt McConeghy

The VIC 20 Users Club
Warwick, RI 02886
Tom Davey

SOUTH CAROLINA

Beaufort Technical College
100 S. Ribaut Rd.
Beaufort, SC 29902
Dean of Instruction

Computer Users Society
of Greenville
Horizon Records-Home Computers
347 S. Pleasantburg Dr.
Greenville, SC 29607
(803) 235-7922
Bo Jeanes

SOUTH DAKOTA

PET User Group
515 South Duff
Mitchell, SD 57301
(605) 996-8277
Contact: Jim Dallas

VIC/64 Users Club
203 E. Sioux Ave.
Pierre, SD 57501
(605) 224-4863
Larry Lundeen

TENNESSEE

River City Computer
Hobbyists
Memphis, TN
1st Mon. at Main Library

Nashville Commodore Users Group
P.O. Box 121282
Nashville, TN 37212
3rd Thurs. at Cumberland Mus

Commodore User Club
Metro Computer Center
1800 Dayton Blvd.
Chattanooga, TN 37405
Mondays 7:30 pm

Metro-Knoxville 64 Users Club
7405 Oxmoor Rd., Rt. #20
Knoxville, TN 37921
(615) 938-3773
Ed Pritchard

TEXAS

SCOPE
1020 Summit Circle
Carrolton, TX 75006

PET Users
2001 Bryan Tower
Suite 3800
Dallas, TX 75201

Larry Williams
P.O. Box 652
San Antonio, TX 78293

PET User Group
John Bowen
Texas A & M
Microcomputer Club
Texas A & M, TX

CHUG (Commodore Houston
Users Group)
8738 Wildforest
Houston, TX 77088
(713) 999-3650
Contact: John Walker

Corpus Christi Commodores
3650 Topeka St.
Corpus Christi, TX 78411
(512) 852-7665
Bob McKelvy

Commodore Users Group
5326 Cameron Rd.
Austin, TX 78723
(512) 459-1220
Dr. Jerry D. Frazee

VIC Users Group
3817 64th Dr.
Lubbock, TX 79413

Southeast Houston VIC
Users Group
11423 Kirk Valley Dr.
Houston, TX 77089
(713) 481-6653

64 Users Group
2421 Midnight Circle
Plano, TX 75075
S.G. Grodin

Savid Computer Club
312 West Alabama
Suite 2
Houston, TX 77006
Davi Jordan, Chairman

UTAH

Utah PUG
Jack Fleck
2236 Washington Blvd.
Ogden, UT 84401

The Commodore Users
Club
742 Taylor Avenue
Ogden, UT 84404
Contact: Todd Woods Kap,
President
David J. Shreeve,
Vice President

The VIClic
799 Ponderosa Drive
Sandy, UT 84070
Contact: Steve Graham

user groups

VIC 20 Users
324 N. 300 W.
Smithfield, UT 84335
Dave DeCorso

Northern Utah VIC & 64
Users Group
P.O. Box 533
Garland, UT 84312
David Sanders

The Commodore Users Group
652 West 700 North
Clearfield, UT 84015
(801) 776-3950
Rodney Keller, Richard Brenchly

VIRGINIA

Northern VA PET Users
Bob Karpen
2045 Eakins Court
Reston, VA 22091
(803) 860-9116

VIC Users Group
Rt. 2, Box 180
Lynchburg, VA 24501
Contact: Dick Rossignol

VIC Users Group
c/o Donnie L. Thompson
1502 Harvard Rd.
Richmond, VA 23226

Dale City Commodore
User Group
P.O. Box 2004
Dale City, VA 22193
(703) 680-2270
James Hogler

Tidewater Commodore
Users Group
4917 Westgrove Rd.
Virginia Beach, VA 23455
Fred Monson

Fredericksburg Area
Computer Enthusiasts
P.O. Box 324
Locust Grove, VA 22508
(703) 972-7195
Michael Parker

Commonwealth 20/64
Users Group
1773 Wainwright Dr.
Reston, VA 22090
(703) 471-6325
Tal Carawan, Jr.

VIC 20 Victims
4301 Columbia Pike #410
Arlington, VA 22204
(703) 920-0513
Mike Spengel

Peninsula Commodore 64
Users Group
124 Burnham Place
Newport News, VA 23606
(804) 595-7315
Richard G. Wilmoth

WASHINGTON

NW PET Users Group
2565 Dexter N. 3203
Seattle, WA 98109
Contact: Richard Ball

PET Users Group
c/o Kenneth Tong
1800 Taylor Ave. N102
Seattle, WA 98102

Whidbey Island Commodore
Computer Club
947 N. Burroughs Ave.
Oak Harbor, WA 98277
Michael D. Clark

Central Washington
Commodore Users Group
1222 S. 1st St.
Yakima, WA 98902
Tim McElroy

Blue Mountain Commodore
Users Club
667 Canary Dr.
Walla Walla, WA 99362
(509) 525-5452
Keith Rodue

Spokane Commodore User Group
N. 4311 Whitehouse
Spokane, WA 99205
(509) 328-1464
Stan White

WEST VIRGINIA

Personal Computer Club
P.O. Box 1301
Charleston, WV 25325
Cam Cravens

WISCONSIN

Sewpus
c/o Theodore J. Polozynski
P.O. Box 21851
Milwaukee, WI 53221

Waukesha Area Commodore
User Group (WACUG)
256½ W. Broadway
Waukesha, WI 53186
Contact: Walter Sadler
(414) 547-9391

Commodore User Group
1130 Elm Grove St.
Elm Grove, WI 53122
Tony Hunter

Commodore 64 Software
Exchange Group
P.O. Box 224
Oregon, WI 53575
E. J. Rosenberg

C.L.U.B. 84
6156 Douglas Ave.
Caledonia, WI 53108
(414) 835-4645 pm
Jack White
2nd Sat every month 10:00 am

VIC-20 & 64 User Group
522 West Bergen Dr.
Milwaukee, WI 53217
(414) 476-8125
Mr. Wachtl

Menomonie Area Commodore
Users Group
510 12th St.
Menomonie, WI 54751
(715) 235-4987
Mike Williams

WYOMING

Commodore Users Club
c/o Video Station
670 North 3rd #B
Laramie, WY 82070
(307) 721-5908
Pamela Nash

CANADA

Toronto PET
Users Group
381 Lawrence Ave. West
Toronto, Ontario, Canada
M5M 1B9
(416) 782-9252
Contact: Chris Bennett

PET Users Club
c/o Mr. Brown
Valley Heights Secondary School
Box 159
Langton, Ont. N0E 1G0

Vancouver PET Users Group
P.O. Box 91164
West Vancouver, British
Columbia
Canada V7V 3N6

CCCC (Canadian
Commodore Computer Club)
c/o Strictly Commodore
47 Coachwood Place
Calgary, Alberta, Canada
T3H 1E1
Contact: Roger Olanson

W.P.U.G.
9-300 Enniskillen Ave.
Winnipeg, Manitoba R2V 0H9
Larry Neufeld

VIC-TIMS
2-830 Helena St.
Trail, British Columbia
V1R 3X2
(604) 368-9970
Greg Goss

Arva Hackers
Medway High School
Arva, Ontario N0M 1C0
D. Lerch

Nova Scotia Commodore
Computer Users Group
66 Landrace Cres.
Dartmouth, N.S. B2W 2P9
Andrew Cornwall

Bonnyville VIC Cursors
Box 2100
Bonnyville, Alberta T0A 0L0
(403) 826-3992
Ed Wittchen

FINLAND

VIC-Club in Helsinki
c/o Matti Aarnio
Linnustajankj 2B7
SF-02940 ESP00 94
Finland

ITALY

Commodore 64 Club
Universita di Studi shan
V. Avigliana 13/1
10138 TORINO
ITALY

KOREA

Commodore Users Club
K.P.O. Box 1437
Seoul, Korea
Contact: S. K. Cha

User Bulletin Board

User Groups Forming:

FLORIDA

Tampa Bay area
Contact Jeff Cornes
10208 N. 30th St.
Tampa 33612
(813) 977-1056

NEW YORK

Buffalo PET Users Group
369 Niagra Falls Blvd.
Amherst, NY 14226
Contact Paul Van Sickle at
(716) 835-5825
or Peter Heffner at
(716) 832-1806

TEXAS

Anyone interested in organizing a
user group in the 76XXX zip code
area? Contact Charles Knerr
2705 Kidd Dr.
Pantego 76013
265-1381

MEXICO

Asociacion De Usarios
Commodore
c/o Alejandro Lopez
Arechiga
Holbein 174-6° Piso
Mexico 18, D.F.

Club de Usarios Commodore
Sigma del Norte
Mol del Valle, Local 44
Garza Garcia, N.L. 66220

NEW ZEALAND

Commodore Users Group
Meet at VHF Clubrooms
Hazel Ave.
Mount Roskill
3rd Wed. of month, 7:30 pm
Roger Altena 278-5262

Nelson VIC Users Group
c/o P.O. Box 860
Nelson, New Zealand
Peter Archer

E.R. Kennedy
c/o New Zealand Synthetic
Fuels Corp. Ltd.
Private Bag
New Plymouth

NORWAY

VIC Club of Norway
Nedre Bankegt 10,
1750 Halden
Norway

UNITED KINGDOM

North London Hobby
Computer Club
Dept. of Electronics &
Communications
Engineering
The Polytechnic of North
London
Holloway Rd.
London N7 8DB

Croydon Microcomputer Club
111 Selhurst R.
Selhurst, London SE25 6LH
01-653-3207
Vernon Gifford

T'AIDE SOFTWARE

for use with
PET/CBM™ Computers, CBM disk drives and
CBM Dot-Matrix, Tractor-Feed Printers
(Excludes™ VIC 20 and Commodore 64)

MATHEMATICS, BASIC SKILLS

Twenty-five programs on disk designed for use by the classroom teacher as a primary source of written exercises in arithmetic. Included are problem sets in integers, decimals, fractions, mixed numbers and percent. Most problems are presented explicitly. Some few, however, require student set-up. In all cases, work is performed directly upon the computer-produced, exercise sheets. Digits are "spaced," where applicable, to provide room for carries, etc. Difficulty levels are easily adjustable. Answers are optimized for in-class use. Five interaction programs are included for diagnostics and intensive drill. $99.99

ALGEBRA

Eighteen programs on disk that produce written exercises in linear and fractional equations, simultaneous equations, quadratics, signed and complex number arithmetic. Answers are optimized for in-class use. $74.99

both disks $149.99
Please specify disk drive employed.

Add $2.50 for postage and handling.
(Arizona residents, add 5% sales tax.)

Please note. T'Aide sells only to schools. Express permission is granted to the purchaser to copy these disks and disseminate freely within the district-of-purchase.

For information or order, contact

T'AIDE SOFTWARE COMPANY
P.O. BOX 65
EL MIRAGE, ARIZONA 85335

™Trade marks of Commodore Business Machines

@#?!@ I CUS®
For C-64™ & VIC™ People

If you belong to only one user group you need to belong to "I CUS."

- A user group for people without the time or location to attend local meetings.
- Access to free public domain and member software.
- Newsletter with unbias, helpful articles and programs. Plus! want ads for members with used hardware, software, services wanted by members or offered, etc.
- A network of members to help with your computer problems and much much more!

Free choice of any disk or tape of programs with $20 membership fee.

WRITE: "I CUS"
335 West 39th St.
New York, NY 10018

C-64 and VIC are trademarks of Commodore Business Machines, Inc.

BIBLE Software

For Commodore 64TM Bible learning games for home and sunday school. Action game JONAH and The WHALE only $16.95.

To order or request information write:
BIBLE Software
4169 S.E. Davidson
Albany, OR. 97321

The Banner Machine™
For the Commodore 64 (with 5 fonts) or the VIC-20 with 24K memory. • Use on any Epson MX with Graftrax or the FX and RX printers. • Menu-driven program operates like a word processor. • Makes signs up to 10" tall by any length. • Makes borders of variable width up to ¾". • 8 sizes of letters from ¾" to 6½" high. • Proportional spacing; Automatic centering; Right and left justifying. $49.95

For the Commodore 64:

Space Raider An amazing arcade simulation. Your mission is to destroy the enemy ships. $19.95

Super Roller Challenging dice game. Sprite graphics and sound. Yahtzee-style rules of play. $14.95

Microbroker Exciting, realistic and educational stock market simulation based on plausible financial events. $29.95—Tape, $34.95—Disk

ABC Fun Part of a series of educational programs for preschool aged children. $14.95

File Organizer Creates Data Base storage system. For mailing lists, club rosters, check books, etc. $29.95

Home Finance Manager Complete mini-financial management program that provides flexibility in maintaining all details of home record keeping. $39.95

Sprite Editor The easy way to create, copy, alter, and save up to 224 sprite shapes. $24.95

Cross Reference Generator for BASIC programs Displays line numbers in which any word of BASIC vocabulary appears. Allows you to change variable name and ask for lines where it appears, and more. $19.95

For the VIC-20:

Caves of Windsor A cave adventure game. The object is to restore wealth and happiness to the small village of Windsor. $14.95

Burger & Fries Fast action joystick game. Eat the burgers and fries but avoid the shakes for a top score. $14.95

Catalog available
Dealer inquiries invited
PHONE ORDERS: (703) 491-6502
HOURS: 10 a.m. to 4 p.m. Mon.—Sat.

Virginia Micro Systems

Virginia Micro Systems
13646 Jeff Davis Hwy
Woodbridge, VA 22191

Commodore 64 and VIC-20 are registered trademarks of Commodore Electronics Ltd.

Commodore 64™ Software

Nobody else competes with our Quality • Prices • Support

FREE with any purchase — CREATE-A-POEM A natural language processing program.

Choose From These Fine Products:

Screen Editor $29.90 Hybrid BASIC/machine language word processor. Can be used to write BASIC programs. Printing features. No character restrictions.

Smart Terminal $29.90 Full machine language program for maximum speed. Software controllable. Full set of escape characters. Downloading and uploading. Full keyboard set (inc. tilda, pipe, backslash).

Electronic Artist $14.90 *(Requires Joystick.)* Color palette selection. Draw. Region fill. Image save. Circle generator. Straight edge feature.

Graphics Pkg. $14.90 Plot and unplot. Fast horizontal/vertical line draw. Grayscale option. Sprite editor. Programmable character editor.

Math/Statistics Package $14.90 Linear, logarithmic regression. Matrix routines. General statistics. Base conversions.

Mini-Assembler/Monitor $14.90 Full syntax checking. Understandable error messages. All addressing modes. Load and execute commands. Timing stats. Pre-defined Kernel labels.

All software available in either cassette or disk

Our Guarantee to You!
Our software is guaranteed to be bug-free and dummy-proof with user-friendly, professional documentation. If you are not satisfied, we will replace the software.

Please send check or money order. Specify cassette or disk. California residents add sales tax. *Send for our free listing of available software.*

Bay Area Software
Suite CM1, P.O. Box 322, Livermore, CA 94550

Commodore 64 is a trademark of Commodore Business Machines

'MIGHTY V' IS HERE TO SAVE THE DAY!

BUSINESS & HOME SOFTWARE

*Commodore 64
*Vic 20

OVER 65 APPLICATIONS AVAILABLE ON CASSETTE OR DISK

POWERBYTE
BOX 579, 905 LORIEN DRIVE
GWYNEDD VALLEY, PA 19437
(609) 346-3063

that does not compute...

"Subject-Oriented Educational Software"

Commodore, May 1983

On page 65, the phone number for SLED Software is incorrect. The phone number should be: (612) 926-5820. Helen Beaubaire of SLED also points out that the company produces software for Junior/Senior High School Language Arts, although they were omitted from that category in our listing. **C**

new books

From Hayden Book Company

50 Essex Street
Rochelle Park, NJ 07662

VIC Graphics by Nick Hampshire. Includes 38 complete graphics programs for the VIC 20. Applications range from art to games to education and business. Programs build to reveal techniques of three-dimensional drawing. Requires use of the Super Expander cartridge.

Using Microcomputers in Business: A Guide for the Perplexed, Second Edition by Stanley S. Veit. Describes the advantages and disadvantages of computerization and enables the potential purchaser to make intelligent decisions.

Secrets of Better BASIC by Ernest E. Mau. Offers faster and more effective programs for testing and debugging, more efficient use of memory, string-handling, using loops and subroutines and creating disk files.

From Osborne/McGraw Hill

2600 Tenth Street
Berkeley, CA 94710

54 VisiCalc™ Programs: Finance-Statistics-Mathematics by Robert H. Flast. Manage investments, loans, taxes and solve over 30 different statistical and mathematical problems with this collection of ready-to-use VisiCalc programs.

MicroSoft™ BASIC Made Easy by Walter A. Ettlin and Gregory Solberg. Gain a better understanding of programming while you learn to develop and customize programs with this fundamental guide.

The Programmer's CP/M™ Handbook by Andy Johnson-Laird. An exhaustive coverage of CP/M-80™—its internal structure and major components. 750 pages, written for the serious programmer. **C**

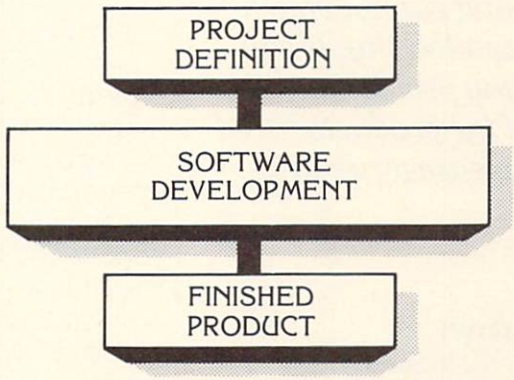
Finally, the most powerful productive software development tool!

More and more programmers are turning to assembly language for its flexibility and performance. Yet up until now they've had to rely on non-friendly, cumbersome, non-flexible, one processor development systems that left assembly language programming at best difficult.

BUT NOW THE SIMA™-UNIVERSAL ASSEMBLER DEVELOPMENT PACKAGE is going to REVOLUTIONIZE the way machine language programs are written by offering the most advanced software development capabilities available on any computer system today. Analytical Software Design is offering its in-house SIMA Development Package for Commodore Computers. Once you compare the SIMA Development Package you'll soon see why you CAN'T be without one.

IMAGINE

- 6502, 65C02, 6800, 6809, 8080, Z80 processors supported by one assembler.
- By far the most advanced source code EDITOR you have ever seen.
- Up to 12 character label lengths.
- Online HELP screens so you won't have to remember command syntax's.
- Upper/Lower/Graphic symbols supported.
- Over 75 source code subroutines for the 6502 processor.
- Linker - Loader that down loads object code to EPROM program, software RS-232 port, Centronic parallel port, IEEE modem or RAM memory.
- Sorted label file.
- Label cross reference file with source code file numbers.
- Conditional assembly.
- The most powerful 6502 debugging monitor.
- Complete BASIC and DISK DRIVE memory maps.
- Source Code conversion program for the MAE and CBM Assemblers.
- USER MANUAL that teaches you 6502 programming.
- PRICE TO GOOD TO BE TRUE.



Price: $75.00
with MAE or CBM® Assembler trade in $50.00

ASD®

ANALYTICAL SOFTWARE DESIGN, INC.
220 City Boulevard West, Suite #111
Orange, California, 92668
(714) 978-3111

SIMA is a trademark of Analytical Software Design, Inc. CBM is a registered trademark of Commodore Electronics Ltd. Product specifications may change without notice.

new products

The following information is taken from new product announcements sent to us by independent manufacturers and is provided only to help keep our readers abreast of developments. Commodore does not endorse any of the products mentioned, has not tested them and cannot vouch for their availability. If you have any problems with any of the products listed here, please write to us.

Company:
RAK Electronics
Box 1585
Orange Park, FL 32067-1585

Product:
Commodore 64 World Clock—See the time in cities all around the globe at a single glance. Plots a high-res graphic map of the world, along with numerous cities and their times. Calculates world time from your local time. Even plots the apparent position of the sun. Instructions included allow you to customize the program by adding your city and local time to the display. Corrects for Daylight Savings Time and AM and PM in the United States.
Price: $7.95 tape; $10.95 disk. Add $2.00 shipping and handling.

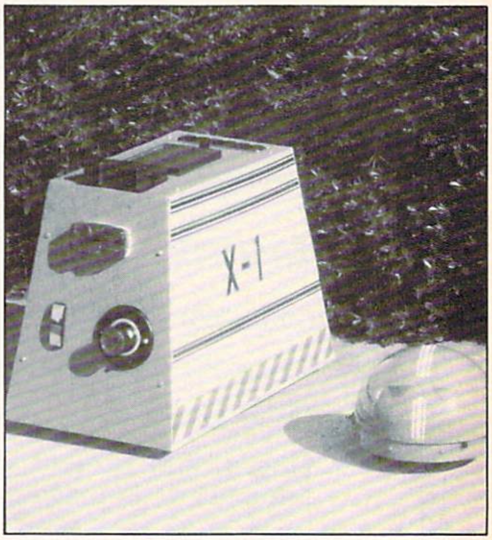
Company:
PC Specialties
P.O. Box 23
Fleming, PA 16833

Product:
VIC 20 expansion hardware—Model VM101 expands the VIC's one expansion slot to six slots. All six slots are addressed through line drivers, which provide reliable buffered software slot selection. The board can shut off the eight data lines from three slots with a rotary switch, so even autostart game cartridges can be left plugged in. The other three slots feature an octal bus transceiver, which buffers all data lines into and out of memory expansion or I/O interfaces. The VM101 also provides a solid state microprocessor reset switch to recover keyboard control when RUN/STOP-RESTORE won't, and has an on-board power supply for loaded systems, isolation of "noisy" I/O devices or non-volatile memory.

Company:
Robot Shack
P.O. Box 582
El Toro, CA 92630
714-768-5798

Product:
Two Home Robot Kits—*DROID BUG* Kit can be assembled in several hours to teach basic robot construction. The droid runs around the floor, and when it senses an object in its way it makes a buzz sound and automatically turns away from the obstacle. The *X-1* Kit is an advanced home robot that can move about anywhere at the speed of a slow walk. Some of its options include: on-board computer control, a hearing sense, human-approaching detection and alarm, obstacle sensing, ambient light sensing, eight-channel remote radio control and solar battery charging. Both are designed for ease of assembly.

Also available for more advanced roboteers: all parts needed to build your own robot from scratch.
Price: DROID-BUG $99.95; X-1 $299.95; Home Robot start-up package, including photos, catalog and club membership, $5.00 refundable with first order.
X-1 and Droid Bug Home Robots



X-1 and Droid Bug Home Robots

Company:
(M)agreeable Software
5925 Magnolia Lane
Plymouth, MN 55442
612-559-1108

Product:
Stock HELPER™—for the Commodore 64. Written by a "weekend investor" for other weekend investors, the program lets you maintain a history on disk of stock prices and market indicators. A menu-driven tool that displays charts and calculates moving averages over a 52-week period. Accommodates stock splits, name and symbol changes and sorting by name and market. Refrains from giving you advice.
Price: $30.00 U.S. plus $1.25 shipping; $37.00 Canada plus $1.50 shipping.

Company:
Pro-Line Software
Mississauga, Ontario, Canada
L4Y 4C5

Product:
POWER 64—a comprehensive programmer's BASIC utility for the Commodore 64. Written by Brad Templeton, with comprehensive manual by Jim Butterfield. Provides automatic line numbering and re-numbering, complete tracing functions, single stepping through programs, debugging ease with a "why" command, ability to merge programs, hexadecimal and decimal conversions and more. Uses only 4K of memory.
Price: $99.95

Company:
Right On Programs
P.O. Box 977
Huntington, NY 11743
516-271-3177

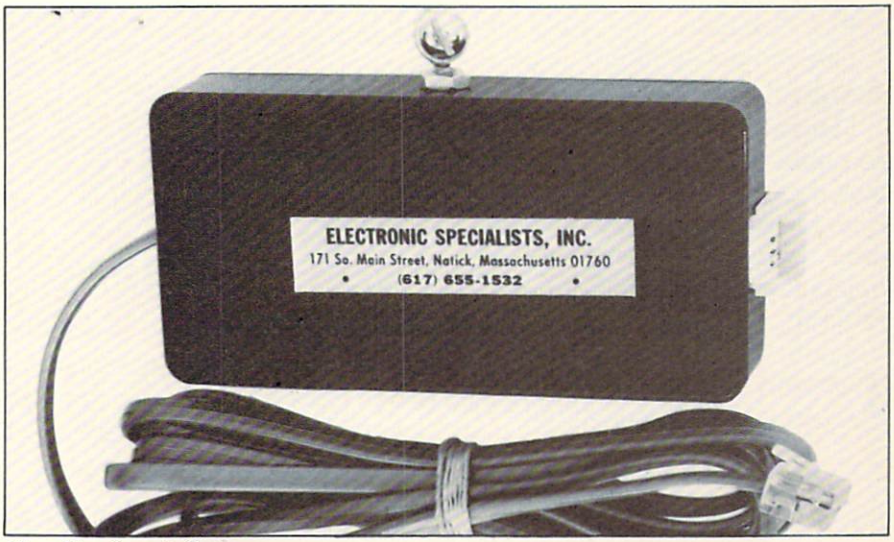
Product:
CHALLENGEIT!!! Series—educational programs for 32K PET. Sold in packages containing three different programs on the sixth grade level and three on the fifth grade level. Each package consists of six sections: lessons, a game based on the lessons, questions and activities, vocabulary, a crossword puzzle based on the vocabulary and a bibliography.
Price: $100 per set

Company:
H & H Enterprises
5056 North 41st Street
Milwaukee, WI 53209

Product:
Disk Support—for VIC 20 and Commodore 64. Provides a 1K machine language extension that adds twelve new commands to the VIC and 64. You can SAVE, SAVE WITH REPLACE, LOAD, VERIFY, DELETE and RENAME disk files with two keystrokes. Also provided are commands that INITIALIZE, FORMAT or VALIDATE a diskette, EXECUTE any program, print ERROR messages to the screen and list the diskette's directory to the screen without affecting the contents of the computer's memory. Compatible with all memory expansion cartridges and with Commodore's Programmer's Aid and Super Expander cartridges.
Price: $14.95

Company:
Electronic Specialists
171 South Main Street
Natick, MA 01760
617-655-1532

Product:
Kleen Line Security System—modem protection. Intended to suppress damaging telephone line spikes, the system uses two-stage semi-conductor and gas discharge tube suppression techniques. An isolated ground is employed to



Kleen Line Security System

new products

isolate equipment from damaging lightning and discharge current.
Price: $56.95

Company:
Spinnaker Software
215 First Street
Cambridge, MA 02142
617-868-4700

Product:
Two educational games for the Commodore 64—*Fraction Fever,* on cartridge, combines numerical and visual representations of fractions, using quick joystick action. *Alphabet Zoo* teaches children ages 3-8 the relationship of letters and sounds and how to spell while having fun. On disk or cartridge.
Price: Contact company

Company:
Computer Directions
for Schools
P.O. Box 1136
Livermore, CA 94550

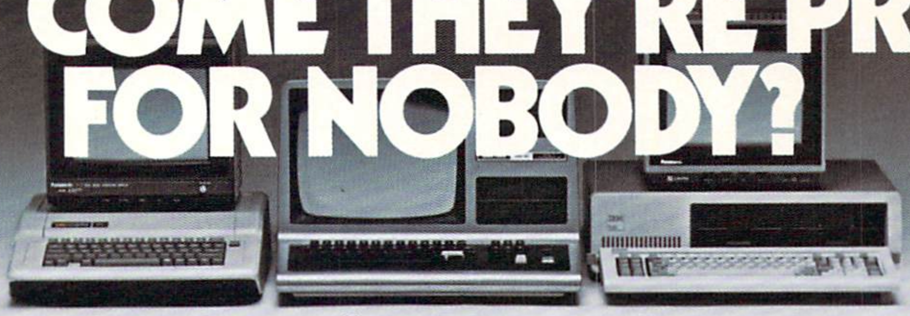
Product:
Manuals to help educators plan computer-related activities—Titles include: *Organizing a Computer Club for Elementary School Children; Student Involvement—Implementing a Computer Tutor Program; Gaining Community Support—Planning a Computer Awareness Day; Teaching Word Processing in the Elementary School* and *Organizing Your Computer Program—Lab vs. Classroom Usage.* Several new titles available soon.
Price: $6.50-$6.95

Company:
Riverside Data, Inc.
P.O. Box 300
Harrods Creek, KY 40027
502-228-3820

Product:
PLUMB: Probing the World of Personal Telecommunications—newsletter to help computer users explore the many services available when their computer is connected with a modem and telephone. Provides usable, non-technical information about telecommunications.
Price: $20.00 for five issues. **C**

advertisers index

IF PERSONAL COMPUTERS ARE FOR EVERYBODY, HOW COME THEY'RE PRICED FOR NOBODY?



$1395* APPLE® IIe 64K

$999* TRS-80® III 16K

$1355* IBM® PC 64K

A personal computer is supposed to be a computer for persons. Not just wealthy persons. Or whiz-kid persons. Or privileged persons.

But person persons.

In other words, all the persons whom Apple, IBM, and Radio Shack seem to have forgotten about (including, most likely, you).

But that's okay. Because now you can get a high-powered home computer without taking out a second mortgage on your home.

It's the Commodore 64. We're not talking about a low-priced computer that can barely retain a phone number. We're talking about a memory of 64K. Which means it can perform tasks most other home computers can't. Including some of those that cost a lot more. (Take another look at the three computers above.)

By itself, the Commodore 64 is all the computer you'll ever need. Yet, if you do want to expand its capabilities some day, you can do so by adding a full complement of Commodore peripherals. Such as disk drives. Modems. And printers.

You can also play terrific games on the Commodore 64. Many of which will be far more challenging than those you could ever play on a game machine alone.

And as great as all this sounds, what's even greater-sounding is the price. It's hundreds of dollars less than that of our nearest competitor.

So while other companies are trying to take advantage of the computer revolution, it seems to us they're really taking advantage of something else:

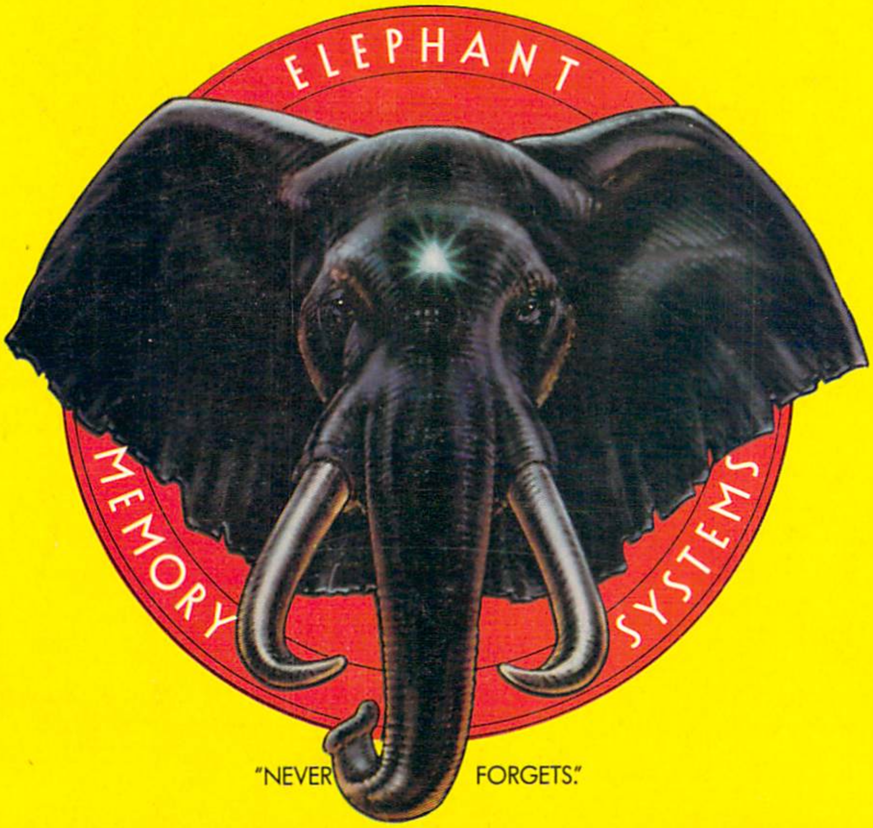
Their customers.

*Manufacturers' suggested list prices.
Monitor included with TRS-80 III only. Commodore Business Machines–P.O. Box 500R, Conshohocken, PA 19428; Canada–3370 Pharmacy Avenue, Agincourt, Ont., Can. M1W 2K4.

commodore COMPUTER

Apple is a registered trademark of Apple Computer, Inc.
TRS-80 is a registered trademark of Tandy Corp. IBM is a registered trademark of International Business Machines Corp.

REMEMBER:



MORE THAN JUST ANOTHER PRETTY FACE.

Says who? Says ANSI.

Specifically, subcommittee X3B8 of the American National Standards Institute (ANSI) says so. The fact is all Elephant™ floppies meet or exceed the specs required to meet or exceed all their standards.

But just who is "subcommittee X3B8" to issue such pronouncements?

They're a group of people representing a large, well-balanced cross section of disciplines—from academia, government agencies, and the computer industry. People from places like IBM, Hewlett-Packard, 3M, Lawrence Livermore Labs, The U.S. Department of Defense, Honeywell and The Association of Computer Programmers and Analysts. In short, it's a bunch of high-caliber nitpickers whose mission, it seems, in order to make better disks for consumers, is also to make life miserable for everyone in the disk-making business.

How? By gathering together periodically (often, one suspects, under the full moon) to concoct more and more rules to increase the quality of flexible disks. Their most recent rule book runs over 20 single-spaced pages—listing, and insisting upon—hundreds upon hundreds of standards a disk must meet in order to be blessed by ANSI. (And thereby be taken seriously by people who take disks seriously.)

In fact, if you'd like a copy of this formidable document, for free, just let us know and we'll send you one. Because once you know what it takes to make an Elephant for ANSI . . .

We think you'll want us to make some Elephants for you.

ELEPHANT.™ HEAVY DUTY DISKS.

For a free poster-size portrait of our powerful pachyderm, please write us.
Distributed Exclusively by Leading Edge Products, Inc., 225 Turnpike Street, Canton, Massachusetts 02021
Call: toll-free 1-800-343-6833; or in Massachusetts call collect (617) 828-8150. Telex 951-624.