commodore
the microcomputer magazine

Volume 4, Number 5, Issue 26
$2.50 U.S.
$3.50 Canada

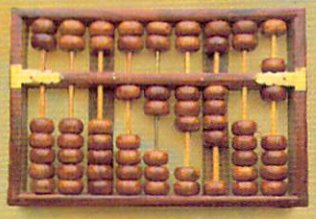
Commodore Business Machines, Inc.
1200 Wilson Drive · West Chester, PA 19380

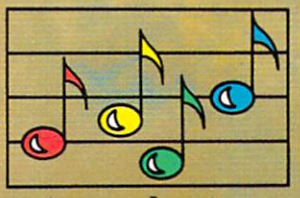
It writes, rates, creates, even telecommunicates. Costs less, does more- the Commodore 64.

When Commodore introduced the 64, the industry suddenly realized that there would be a computer in every home, school and business years before anyone ever dreamed.

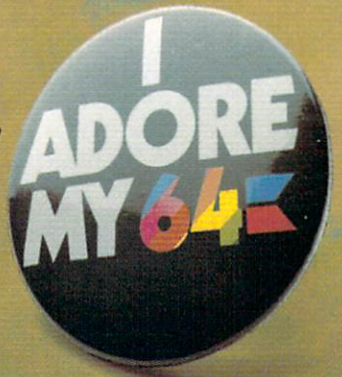
That's because Commodore 64 halved the price of high technology: while you can compare the 64's capabilities with those of any sophisticated business PC, you can compare its price with that of an average television.

What can you do with it? Create with its high resolution Sprite Graphics. Add a printer and type with it. Add a disk drive to use spread sheets and other financial programs. Learn and play music through your home sound system on the 64's professional quality music synthesizer.

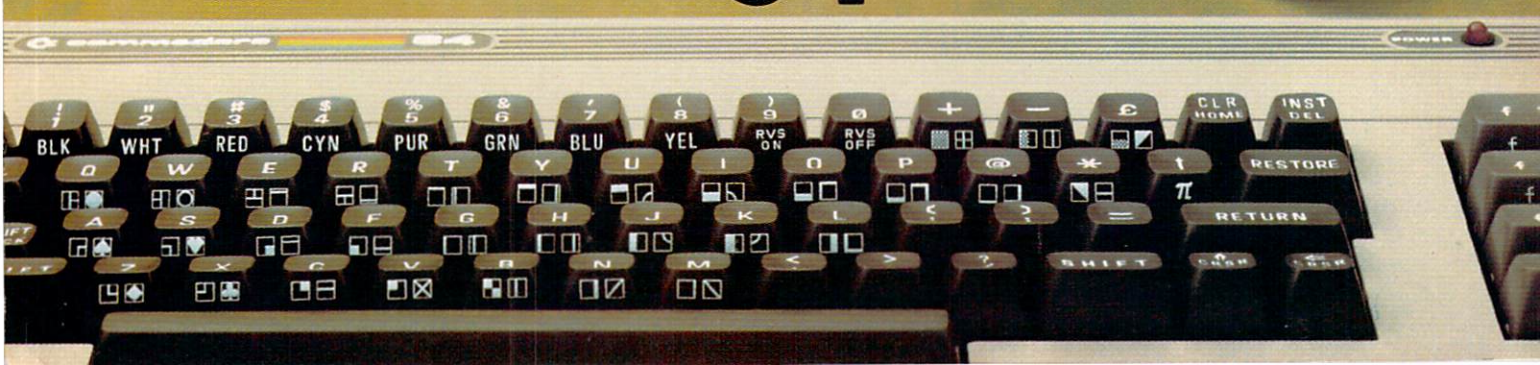




Add a modem, and hook up with the vast computer networks through your telephone. In short, the Commodore 64 is the ultimate personal computer, at a price you can afford.



COMMODORE 64



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.

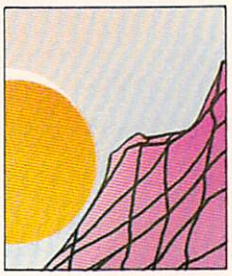
Computer Peripherals Division, 1120 Empire Central Place, Suite 216, Dallas, TX 75247 (214) 631-8560

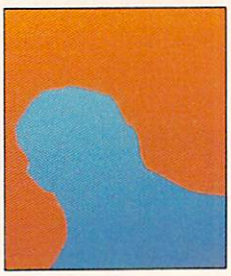
The new STX-80 printer for only $199.*

commodore
the microcomputer magazine

features

Volume 4, Number 5, Issue 26





departments

commodore the microcomputer magazine

staff

Publishing Manager
Neil Harris
Editor
Diane LeBold
Technical Editor
Jim Gracely
Staff Writers
Ginger Bardi
Bernard Falkoff
Larry Greenley
Lynn Kalcheries
Barbara Karpinski
Cyndie Merten
Sarah Meyer
Stephen Murri
Contributing Writers
Gail Austin
Chris Bennett
Danny Byrne
Bruce and David Cameron
Elizabeth Deal
Robert Embree
Donald Hassler
Craig Hessel
Bruce Jaeger
Jane Reh
Joe Rotello
Jim Strasma
Dave Whomsley
Gregory Yob
Mark Zimmermann
Technical Staff
Barbara Karpinski
April Koppenhaver
Cyndie Merten
Sarah Meyer
Stephen Murri
Circulation Manager
John O'Brien
Circulation Assistant
Kathy Reigel
Advertising Coordinator
Sharon Steinhofer
Graphic Design
Neumann Greenberg Schlenker
King of Prussia, Pennsylvania
Cover Art
Robert Neumann
Typography
Associates International, Inc.
Wilmington, Delaware
Printing
Volkmuth Printers
St. Cloud, Minnesota

ISBN 0-88731-003-6

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. Volume IV, Number 5, Issue 26.

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).

Watch for our end-of-the-year special issues!

Power/Play, Winter: Our Christmas special will feature all the hottest new items to make this holiday season a real treat for those Commodore owners on your list. We'll be out there in early November with games, home applications, peripherals and much more.

Commodore, Issue 27: If you've ever wondered exactly what a 6502 semiconductor was, this is your chance to find out—in laymen's terms. Our 6502 high-tech special will make the internal workin's of your computer a little clearer to you, without getting over your head. This bolt of enlightenment will fly your way in early December.

Key to Entering Program Listings

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

Predict the Weather

On Your Vic 20™ or Commodore 64™ Computer

This exciting, new weather package allows you to learn, monitor, and predict the weather using the same equipment as the pros.

Be Your Own Weatherman

The new HAWS (Home Automatic Weather Station) combines a quality weather sensor with software to let you track weather conditions inside or outside your home. HAWS monitors weather data including dew point, temperature, humidity, and atmospheric pressure; plots historical data and graphically displays weather trends; determines the comfort and chill factors to help you dress; HAWS even forecasts the weather and compares your predictions to the local weatherman's.

HAWS uses the same sensor employed by weather services in 60 countries worldwide. The HAWS package costs only $199.95 and includes the sensor, cassette or floppy disc program, 15-foot cable with connector for the computer, and a complete user's manual.

Send today or call toll free to order your HAWS package.

Ordering Information

Send $199.95 in check or money order payable to Vaisala, Inc. Include name, address, telephone number, your computer model (VIC 20 or Commodore 64), and program preference (tape cassette or floppy disc).

Or call toll free to place your order.

Major credit cards accepted.

Dealer inquiries invited.

1-800-227-3800 ext. H-A-W-S

VIC 20 and Commodore 64 are trademarks of Commodore Electronics Ltd.

Learn/Teach Meteorology

More than a toy or game, HAWS provides the first opportunity to use your computer as a data sampler and data analysis system for meteorology, allowing the user to interact with incoming data to monitor and forecast weather conditions. HAWS includes an instructive software program and a complete user's manual that teaches meteorological terms and equations to allow anyone to quickly grasp weather concepts either at home or in the classroom. Simple plug-in components and easy hookup also means you can free up your computer at any time for other duties. HAWS is a great educational tool for anyone. Order today.

VAISALA

VAISALA INC.
2 TOWER OFFICE PARK
WOBURN, MA 01801

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

INSTEAD OF SA
KID'S EDUCATION, A
BE SPENDING FOR I

ING FOR YOUR
AYBE YOU SHOULD

We'd be the first to encourage parents to save for their kids' education.

But money alone isn't enough to get anybody into college. Let alone, through it.

At more and more colleges today, computer skills are becoming mandatory. And at some colleges, those skills are becoming required for admission itself.

That's one good reason to get a home computer. Many parents are also discovering how helpful a computer can be for themselves in their homes and businesses.

But with saving for tuition and room and board, who has money to spend for a computer?

One answer is the Commodore 64.™

The Commodore 64 gives you a powerful 64K memory. That's as much memory as either the Apple® IIe or IBM® Personal Computer. But at far less than half the cost.

You also get full-color graphics, a nine-scale music synthesizer, and a wide range of software.

The combination of power, graphics, music and software makes it the perfect computer for a student to start out with. And stay with, right through college.

And beyond.

commodore COMPUTERS

In more homes than any other home computer.

Commodore Business Machines–P.O. Box 500T, Conshohocken, PA 19428;
Canada–3370 Pharmacy Avenue, Agincourt, Ont., Can. M1W 2K4

editor's notes

And the Best is Yet to Come

There was a time, brief as it may have been, when it was fashionable to make remarks like, "The Commodore 64? Oh yeah, a great computer, but there's no software..." You may have suspected, even then, that those kinds of things were being said mainly by folks who didn't know enough to have much else to say. Adopting a negative stance when everybody else is impressed is a terrific way to look knowledgeable. However, those of us who realized what the 64 could do knew right from jump street that it was just a matter of time before the tidal wave would hit.

Well, it's starting to hit—and Commodore itself is riding right on top of the crest with its own array of software for just about everything you might want the 64 to do and then some. The *Super Expander 64* and *Simons' BASIC* packages unlock the mysteries of the 64 (heretofore guarded jealously by a handful of experienced machine language programmers), so even beginners can take full advantage of their computer's capabilities. The wonder of those packages is that they're also a great boon to experienced programmers, as well. In addition, business programs like *Easy Finance* and educational cartridges like *Visible Solar System*, not to mention home applications programs like *Magic Desk* and a myriad of games, are all showing the 64 to be just what it was predicted to be: one of the all time greats in the history of the microcomputer industry.

And now, with the appearance of the SX 64, Commodore's new portable 64 with built-in single disk drive and six-inch color monitor, you can take it all with you. As a result, you can bet disks to donuts that 64 software is going to continue to be a priority at Commodore—and among independent developers as well.

But wait, I hear the clamor of voices... is that the sound of VIC 20 and PET/CBM owners yelling, "Hey, what about us!"? Never fear, kind people. Although we haven't reviewed software for your computers in this issue, we have included a complete list of all Commodore-marketed programs. On this list you'll find everything Commodore is producing for the VIC 20 and PET/CBM as well as the 64. In addition, you can refer to the lists of educational and business software that appeared in Issues 23 and 25 respectively if you'd like to get an even more comprehensive idea of what's available for the full range of Commodore computers. It just so happens that Commodore 64 software is very hot right now, which is why so much of this issue is devoted to it.

By the way, if you're having better luck with our new program listing format, we'd sure like to hear about it. We used to get a fair amount of hate mail from beginners who were having a hard time translating our old dot matrix listings. That has suddenly dropped off, which I assume is a good sign. But it would be nice to get some positive feedback on what we're doing right, once in a while. Anybody got a minute to drop us a line? We do read your letters, although we regret we don't have

time to respond to them all—except by fixing what you don't like in the magazines and giving you more of what you do like.

Speaking of responding. Those of you who've written to us requesting more articles about PET/CBM undoubtedly have noticed that articles have not exactly begun appearing in great numbers. In fact, now that Liz Deal has gotten a Commodore 64 we don't even have her usual helpful hints for the PET—at least for this issue. (Besides which Liz may never forgive us for messing up her machine language program listing in the last issue. We'll be running a fix for that in December.) We're having a little lag among our PET/CBM writers because, I suspect, they (like Liz) have all gone off to play with the 64 for a while. So, if you're using a PET or CBM (or SuperPET) and would like to jump into what has become a rather large blank spot, send for our "Guidelines for Writers" and then write us that article you've been thinking about for the past three months.

Our final issue for 1983 should be an interesting project for us all, since we're going to take the plunge right into the guts of your computer (with some trepidation, I might add) and introduce you to the world-famous 6502 semiconductor that made this all possible. And, no, we're NOT going to call it anything like "Hello, Mr. Chips", either. At least not if I have anything to say about it. C

—Diane LeBold
Editor

COMMODORE 64

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.

MasterCard VISA

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

letters

VAL($) Function Fix

To the Editor:

I would like to pass along a glitch that I have encountered in the VAL($) function of both the ROM 3.0/4.0 versions of Commodore BASIC used in the PET/CBM 2000/4000 computers and also the Commodore 64.

The problem occurs when the string used as the argument begins with an "E" and is immediately followed by numbers or space(s) and numbers, but not always! The following examples of A$ and the VAL(A$) returned illustrate the problem:

```
A$="E  1"    VAL(A$)=0
A$="E 38"    VAL(A$)=0
A$="E+99"    VAL(A$)=0
A$="E-99"    VAL(A$)=0

A$="E  100"   VAL(A$)=OVERFLOW ERROR
A$="E+9000"   VAL(A$)=OVERFLOW ERROR
A$="E-9000"   VAL(A$)=0
A$="E -100"   VAL(A$)=0
```

One would expect the VAL($) function to return a value of zero in all cases, since "E#..." is incorrect scientific notation (the mantissa is missing) and the first non-blank character is non-numeric. All the mathematical functions assign a zero value to this erroneous form of scientific notation, as well as SPC, TAB, FRE, SYS and PEEK. POKE and USR consider it a syntax error. All the other string functions handle this string correctly. The glitch would therefore appear to be in the BASIC routine for the VAL($) function.

When VAL(A$), where A$="E###...", is encountered in the program mode the resulting OVERFLOW ERROR crashes the program. This can be prevented by substituting the expression "VAL(LEFT$(A$,1))", resulting in a slight increase in execution time (and code).

Since the ROM used for BASIC 4.0 to evaluate the VAL($) function (Part #901465-20) is also used in the CBM 4000/8000 machines, I would expect this glitch to be somewhat universal. I am curious to know if this happens in the original BASIC 2.0 and with the VIC 20, since I do not have ready access to these machines. C

Jack B. Cooper
Princeton, New Jersey

Talking PET

To the Editor:

Last week when the June/July issue of *Commodore Magazine* was seen, we were impressed by several things in the article titled "Microcomputers: Truly Child's Play". But, although the story accurately describes Kinder-Care's computer set up for their daycare centers, it misrepresented Commodore's "new talking PET computer" and clearly omitted mention of the natural-voice learning system and software produced by Learning Tree Software that *allows* the PET to talk.

It appears to us that it is not only our loss but Commodore's as well that we were not mentioned, since we are a company that not only has developed materials for the PET and 64 but is in a position, with the uniqueness and quality of our audiographic system, to be of support in Commodore's efforts to supply software for the 64. Ours is the ONLY system of its kind that provides instructional software for young children who cannot read —or have difficulty reading— material on the screen. Our products have been seen as the most viable for this marketplace because of this very capability.

We are requesting that an article be written by the magazine that makes it clear that indeed Commodore's "talking" PET computer presently used in the preschool market is in fact the system created by Learning Tree Software.

Rita Kaplan-Spina, Ph.D.
Vice President, Learning Tree Software
Kings Park, New York

Editor's Note: The Learning Tree Package uses a specially modified tape recorder (*not* a datassette) controlled by the PET. The voice part of the lesson is on a cassette, which the PET turns on and off. We apologize if this wasn't clear in the article. C

New Briefcase Computer Expands Commodore's 64 Line

Commodore has introduced a portable computer designed for the traveling businessman. Designated the SX 64, the new portable has 64K RAM, a full upper/lower case low-profile detachable keyboard, built-in six-inch color monitor and a built-in single floppy disk drive with 170K capacity. The new unit weighs 27.6 pounds and is briefcase size: 5″ × 14½″ × 14½″.

The SX 64, a member of the Commodore 64 family, is fully compatible with VIC 20 and 64 peripherals, including the VICMODEM for telecommunications. External ports allow full-sized monitor and graphic printer hook-ups.

With a PET Emulator, the system can use much of the available PET software. Moreover, the SX 64 can use the large number of game cartridges available to the 64 family of computers and has full music and sound capabilities.

Resident in the unit's ROM is BASIC V2. Other high level programming languages include PASCAL, LOGO, COMAL, Assembler and PILOT. Additionally, the SX 64's 6510 central processor is 6502 program compatible.

The first few hundred portable SX 64s should be available in the U.S. this fall.

Commodore's portable SX 64.

Commodore Software Encyclopedia Lists Over 2,000 Programs

Commodore Software has announced the availability of the third edition of the *Commodore Software Encyclopedia.*

The new edition is the largest *Commodore Software Encyclopedia* ever, with over 800 pages. The encyclopedia contains nearly 2,000 entries including both Commodore and non-Commodore software available from software vendors around the world. It is the most comprehensive single software reference for Commodore computers.

Available through Commodore dealers at a list price of $19.95, the *Commodore Software Encyclopedia* includes the latest Commodore software releases for the full line of Commodore computers including the VIC 20, Commodore 64, PET Series, CBM 4032, 8032 and 8096 and the new B Series computers.

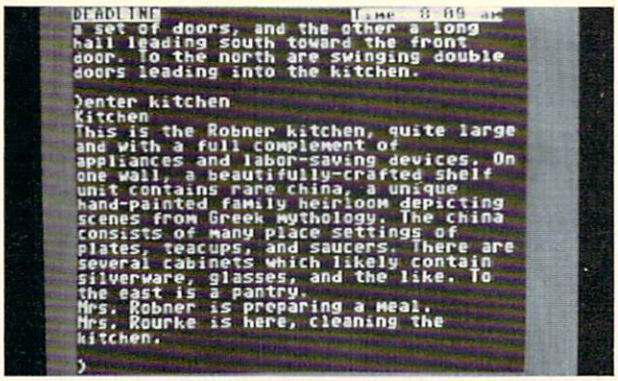
The encyclopedia has 18 major areas of interest including separate sections for the VIC 20 and Commodore 64 plus a 23-page index that makes it easy to find the right software for any Commodore computer owner's specific needs.

Commodore Announces Seven New Adventure Games for Commodore 64

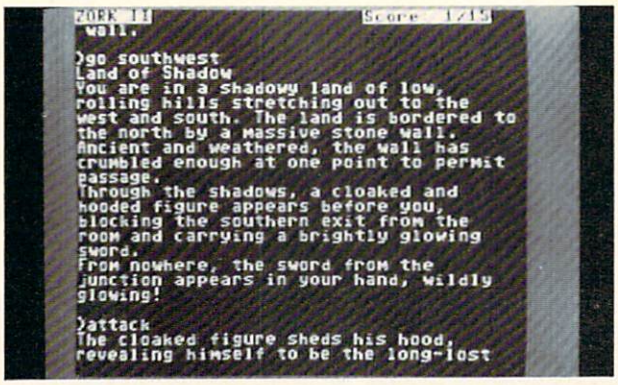
Commodore Software now has seven new adventure games for the Commodore 64 computer. The announcement was made by Sig Hartmann, president of Commodore Software. The new games include: *Zork I: The Great Underground Empire, Zork II: The Wizard of Frobozz, Zork III: The Dungeon Master, Deadline, Starcross* and *Suspended.*

According to Hartmann, "These games include some of the best-selling adventure games in the industry. The *Zork* series has been extremely popular and we believe it will be even more popular on the Commodore 64.

"Our research shows that the game-playing public wants more challenging games, games that make you think and analyze and make decisions... this excellent adventure series meets that need."



Deadline



Zork

The games, which were developed by INFOCOM, Inc.™, have a suggested retail price of $29.95 each.

The *Zork* Trilogy

The *Zork* trilogy for the Commodore 64 was developed using INFOCOM's proprietary INTER-LOGIC™ computer language. Each game contains a vocabulary of over 600 words. The player uses word commands to take various actions during the game.

Zork I: The Great Underground Empire

The object is to strive to discover the Twenty Treasures of Zork and escape with them and your life.

Zork II: The Wizard of Frobozz

The quest continues, with new challenges introduced by the Wizard of Frobozz, a new character to confound your quest.

Zork III: The Dungeon Master

Your final test, culminating in an encounter with the Dungeon Master.

Deadline

You have a 12-hour time limit to solve one of the most baffling cases in the annals of criminology. Game Kit includes an actual dossier on the crime. The player must piece together the clues encountered during the adventure.

Starcross

A mindbending science fiction adventure set in the year 2186. You meet a challenge issued eons ago. Manual and navigation chart are included.

Suspended

You're in "suspended animation." Working through six robots, each equipped with different capabilities, you try to solve a twisting puzzle of problems. The game comes with a detailed manual and schematic of the underground complex that is your "domain."

Commodore Announces Speech Module for the Commodore 64

Commodore has developed true-to-life speech for the Commodore 64. The speech module plugs directly into the user port of the Commodore 64. It contains an additional port into which other "talking" and "non-talking" cartridges can be inserted.

The Commodore speech module contains a built-in vocabulary of 235 words in a pleasant female voice. The voice speed can be user-defined to slow, normal, or fast. The words can be programmed directly from BASIC and/or assembler. The user can program music, graphics and speech simultaneously. The speech module supports a separate audio-out so the user may connect the speech output directly to a hi-fi system, a television or a color monitor. More words and different voices (male, cartoon characters, etc.) will soon be available on disk and cartridge.

Future educational applications on disk and/or cartridge include the alphabet, counting, spelling and animals. Higher level applications will include interactive foreign language modules, higher mathematics and science. Programs will be available from both Commodore and third party producers.

Because the Commodore speech module can be made to produce any voice and a wide range of sound effects, game cartridges will take on an even more realistic effect. Two games soon to be released that currently work with the speech module are *Wizard of Wor* and *GORF*. More games are being prepared for release soon. The speech module can support game cartridges of up to 128K bytes.

Commodore's speech module also plugs directly into Commodore's new portable computer, the SX 64, making it the only "talking" portable on the market. **C**

A VIC 20 and a Ray of Light

by Diane LeBold

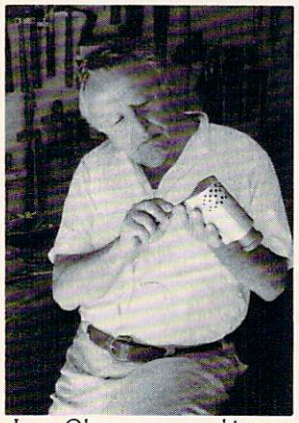
When a stroke felled Ed Ellner in 1982, the 64 year-old electrical engineer was left almost totally paralyzed. His mental faculties remained alert but he was locked inside his body, unable to move or speak. The doctors predicted that Ed would sink into depression and despair and become virtually unreachable. Enter Ed's lifelong friend, Jerry Oberwager, and a VIC 20 computer activated by a beam of light...

On the surface this is a story about a unique computer application—how a VIC 20 is being used by an almost totally paralyzed person to communicate again after a devastating stroke took away both movement and speech. But the truth is that it's really a story about love—the tenacious kind of love that just will not give up hope, no matter what the "authorities" say. And it's a story about the determination of a man to continue his life in spite of an extreme handicap.

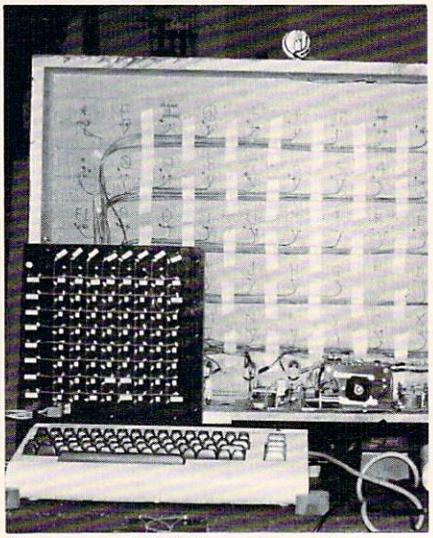
I'm going to tell you mainly about the computer application. But I want you to know right from the beginning that the main force behind all this is only partly the technology. The more important part is the unshakable faith and devotion of a friend, a wife and children—all of whom insisted that Ed Ellner would continue to participate in life, no matter what it took—and Ed's own spirit, which wouldn't let him give up even after the doctors said it was hopeless.

Let's backtrack a moment to get some history squared away. After retiring from his work as an electrical engineer Ed joined the Peace Corps together with his wife Alice, a teacher in the Stratford, Connecticut, school district. In the fall of 1980 the Ellners were sent to the Fiji Islands to assist the East Indian population there with land management. About 20 months later, just before their two-year service in the Fijis was up, Ed was struck down with a brainstem stroke.

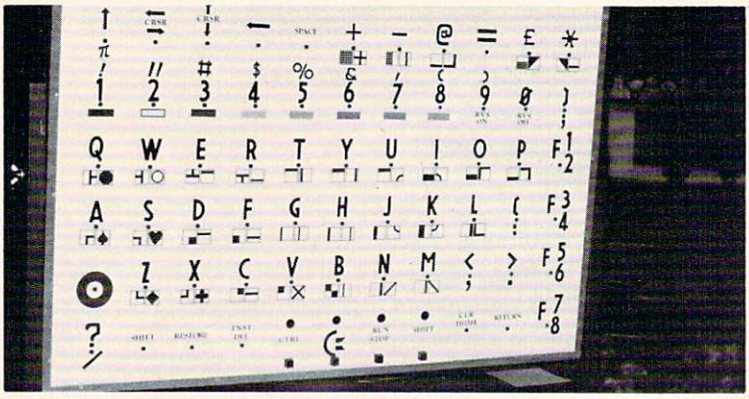
Although his mental faculties remained unimpaired, Ed could not move or speak. He was left with only a small amount of control over his head and the ability to move his eyes. Because his case was so extreme, not even the most sophisticated



Jerry Oberwager working on a new light pointer.



Keys on the VIC 20 are triggered by the photoelectric cells.

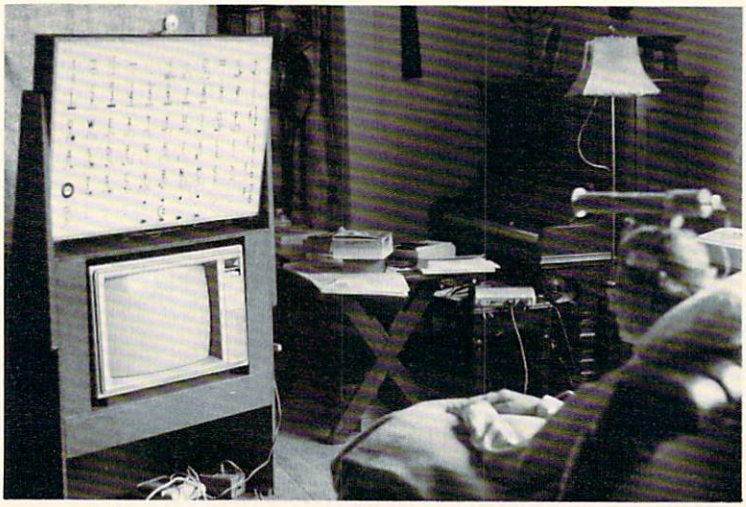


Jerry Oberwager's computerized letterboard began as a simple "spelling machine". Below each "key" on the board is a tiny photoelectric cell.

equipment available in the medical mainstream could help him regain any of his functions. So the doctors dismissed Ed as a lost cause, Alice Ellner says. They predicted her husband would lose the will to live and, as is often typical, withdraw into depression and despair. As a result, they insisted he remain in a convalescent hospital indefinitely.

The doctors, however, didn't realize what an un-typical family they were dealing with. And they didn't know that Jerry Oberwager, Ed's friend since they were in the fourth grade together, has a knack for tinkering around with things until they do what he wants them to do. First, Alice fought to bring Ed home—and won. Meanwhile both Jerry and Alice continued to check around for alternatives. They were looking mainly for some method that would allow Ed to communicate.

We're so used to hearing about the many miracles of medical technology, it's hard to imagine there was nothing suitable for Ed's needs. The reason, Jerry explains, is that to be cost-effective, manufacturers try to develop devices for the broad middle range of handicapped people, most of whom have a certain amount of motion. For instance, a common communication device for people who have no speech but some movement is a scanner or "spelling machine" on which the alphabet keeps scrolling by. Users stop it at each letter they want, usually by pressing a button. People without hand movement but with good head control can use a headstick (a wand strapped to their heads) to push down the keys of a typewriter. However, these and similar methods all require more motion than Ed has.



Ed Ellner uses a light pointer to activate the photoelectric cells on the letterboard. The photoelectric cells in turn activate the keys on the VIC 20 computer wired into the back of the board, and Ed's messages come up on the T. V. screen. He can also print out his messages on the VIC's printer.

Frustrated in their search for a commercial device, Jerry and the Ellners began to take elements from several of their sources to see if they could develop something—anything—that Ed could use. Ed's son Peter took the first step toward success when he created a letterboard, based roughly on the "spelling machine" idea. Peter would point to each letter of the alphabet in turn and Ed would blink his eyes at the letters he wanted. The next step was to try attaching a light pointer to Ed's head so he could point to the letters himself. If the board were set the right distance away, it didn't take much movement to sweep a concentrated ray of light across it, and Ed had some success using it.

By this time inexpensive VIC 20 home computers were becoming widely available. So Jerry naturally (naturally for him, anyway) turned in that direction and took Peter's idea one step further. He decided he could label a letterboard not just with the alphabet but with all the keys on the computer's keyboard. Since Ed could use a light pointer, that meant he could activate photoelectric cells on the letterboard. If each photoelectric cell activated the corresponding key on the computer, Ed would be able to compose his messages on the computer's monitor and even print them on a printer. And if it worked, Ed would not only be able to communicate, he'd be able to write programs!

Jerry began to experiment on a VIC 20 at his local Commodore dea'er near Great Neck, New York. Were he an electronics expert—in fact, were he Ed Ellner—he could have figured out how to wire the photoelectric cells directly into the computer's keys. But, for all his practical know-how, Jerry is no electronics wizard and he couldn't find anyone to advise him. So he resorted to a method he understood—

solenoids. He found a way to have the photoelectric cells activate little solenoids (plungers) that would physically push down on the appropriate keys. This was clumsy, but it worked.

Before he got too much further in his experiments, however, someone at Commodore heard about his efforts. It didn't take long for Commodore to get Jerry his own VIC 20 system (no more fiddling around at the dealer's) and the advice he needed to wire his photoelectric letterboard directly into the computer (no more solenoids). And it didn't take too much longer after that for Jerry to get the whole kit and kaboodle up to Ed's place in Connecticut, complete with a word processor cartridge supplied by Quick Brown Fox in New York City.

Then the acid test. It all sounded good in theory, but could Ed actually use the thing? It was a little tricky at first for several reasons, all of them having to do with simple physical logistics. First, if Ed held the light on a photoelectric cell too long he would get double letters, or if he wasn't precise enough as he swept the light over the board he'd activate letters he didn't want.

To try to fix that, Ed and Peter (who took a crash course in programming so he could help his father) devised a program that delayed the computer's response time. But, although the delay helped prevent unwanted letters, Ed then discovered it was very tiring to hold the light on a cell for the longer time required to activate it. To help eliminate that problem, Jerry has begun to re-design the board, providing more space between letters and allowing Ed to move the pointer diagonally, which Ed says will make it easier for him. Jerry is hoping this new, larger board will eliminate the need for a delayed response—and the resulting frustration and fatigue.

As he tested the letterboard, Ed also found it difficult to keep looking from the board (which sits above the T. V.) to the T. V. screen to make sure the letters were registering. So he and Peter further streamlined their delay program by adding a "beep" tone whenever a key is activated.

In addition, Ed found that the size of the specially designed light pointer (about 14 inches long) made it somewhat cumbersome to use. Jerry is remedying this with a new light that provides the intensity he needs, but is only about three inches long and half the weight of the original. He hopes this will give Ed more control with less fatigue.

In spite of its several limitations Jerry's computerized letterboard has been a success. Not only has Ed been able to communicate, but he has developed more control of his head because he has been "exercising" with the light pointer. After he re-designs the letterboard and improves the light pointer, Jerry says his next step will be to create a method that will allow Ed to choose whole words from a menu of commonly used words, so he won't have to spell out every single thing. As Jerry puts it, "You could do some fast talking with just ninety-nine words."

As a result of Jerry's efforts, in collaboration with the whole Ellner family, Ed Ellner has been able to go on with his life in spite of his extreme handicap. He has not slid down into depression as the doctors predicted. And not only has he been able to go on. He has, in fact, added a new facet: learning about computers. Sparked by that same curiosity and zest that sent him and Alice backpacking through South America several years ago and that motivated them to join the Peace Corps when most people would be heading for a retirement community, Ed has been spending many hours reading every computer book he can find and has been applying his new knowledge to devising programs on his VIC 20.

Nevertheless there are still many things to be worked out in order to make the letterboard even easier to use and help Ed increase his proficiency. At this stage, Jerry and the Ellners are in need of ideas and programming help more than anything else. If you are interested in providing that help or finding out more about how the letterboard works, please contact *Commodore Magazine* and we will put you in touch with Jerry or the Ellners. **C**

Programming Multiple-Voice Music in Machine Language

By Cyndie Merten and Sarah Meyer

You can program longer, more complex pieces of music on your Commodore 64 using this machine language program instead of BASIC.

If you're a machine language programmer, even a beginner, you can add features to multiple-voice music programs that aren't possible in BASIC. The multiple-voice music section in the *Commodore 64 Programmer's Reference Guide* presents a few problems that are hard to solve in BASIC but that you can overcome more easily in machine language. This article shows you a simple method for mastering the two major obstacles: synchronizing the three voices and encoding the data economically. We'll show you that machine language lets you program music with more notes and more rhythms because you use less memory than you would in BASIC.

As the 64 *Programmer's Reference Guide* explains, synchronizing three voices in BASIC is difficult because of the length in realtime a BASIC statement takes to execute. In addition, since IF/THEN statements use varying lengths of time depending on their truth value, it's not a good practice to use IF/THENs to control timing. The solution is machine language, which affords enough speed for sophisticated synchronizing and timing.

Using BASIC DATA statements to encode musical data consumes so much space that you can't program long songs. Machine language economizes storage space because you can access memory locations directly. But some of the memory-saving machine language encoding methods are cumbersome and difficult for beginners. This article shows you how to encode a note in two hex bytes: the first byte for octave and note, and the other byte for note duration.

Here's how the octave/note byte is programmed:

The first nybble (half a byte) is the octave.
—0 is highest
—3 is middle
—7 is lowest
—8 through F flag the end

The second nybble is the note.

0 = C	3 = D#	6 = F#	9 = A
1 = C#	4 = E	7 = G	A = A#
2 = D	5 = F	8 = G#	B = B
			C = rest or silence

EXAMPLES:

```
Middle C = .BYTE $30
Low G#   = .BYTE $58
High B   = .BYTE $1B
Silence  = .BYTE $0C
```

Here's how the duration byte is programmed:

$80 = whole note	$40 = half note
$20 = quarter note	$10 = eighth note
$08 = sixteenth note	$04 = thirty-second note

EXAMPLES:

```
Middle C, whole note    = .BYTE $30,$80
Low D#, quarter note    = .BYTE $53,$20
High A#, dotted quarter = .BYTE $1A,$30
Silence, half note      = .BYTE $0C,$40
```

The program counts down from the duration you specify. When the counter reaches $03, the note is released. Therefore a duration of $03 or less produces an inaudible note.

You can combine the settings to create different durations. For example, add $40 (½) and $20 (¼) to get a dotted half note; add $40 (½) and $10 (⅛) to get a ⅝ note.

Since these values count down a note's duration, you can start the duration counter at any number between $80 and $03 to create standard or unusual durations.

Try this scheme for encoding data with the following simple assembly language program for play-

ing music. The program is explained line-by-line so even a novice machine language programmer should be able to follow the program. The data we've included is from Pachelbel's "Canon in D".

Note: If you're a BASIC programmer interested in learning machine language, Commodore's *64 Assembler Development System* gives you the tools to write assembly language programs.

Another note: The authors wish to thank Bill Hindorff for his technical assistance in perfecting the music program. C

Multiple-Voice Music for the Commodore 64

```
1000 .PAGE 'MUSPRG'             This is an instruction to the
                                assembler to do a formfeed
                                when printing the code out.
1010 SID=$D400                  From now on the name "SID"
                                will refer to location $D400
                                (54272), the beginning of the
                                SID chip registers.
1020 TALO=$DC04                 56324 is the low byte of the
                                latch for timer A.
1030 TAHI=$DC05                 56325 is the high byte of the
                                latch for timer A.
1040 CRA=$DC0E                  56334 is the control register
                                for timer A.
1050 IRQVEC=$EA31               This is the system interrupt
                                vector ($EA31=59953).  Check
                                that your system uses the
                                same vector by looking at
                                locations $0314 (788) and
                                $0315 (789) while BASIC is
                                running.
1060 RESVEC=$FCE2               This is the system restore
                                vector ($FCE2=64738).  Check
                                that your system uses the
                                same vector by looking at
                                locations $fffc (65532) and
                                $fffd (65533) while BASIC is
                                running.
1070 PTR=$68                    This page zero location will
                                be used for indirect
                                addressing.
1080 *=$2000                    All variables will be stored
                                at $2000.  You may use any
                                locations you wish, of
                                course.
1090 DUR        *=*+3           Table of durations, one for
                                each voice.
```

```
1100 PTRS       *=*+6           Table of pointers to the
                                musical data (low byte, high
                                byte), a pair for each voice.
1110 DURA       *=*+1           A temporary storage location
                                for the duration of the next
                                note.
1120 FREQLO     *=*+1           The low byte of the frequency
                                of the next note.
1130 FREQHI     *=*+1           The high byte of the
                                frequency of the next note.
1140 NOTE       *=*+1           Temporary storage of note
                                code.
1150 VOICE      *=*+1           Current voice.
1160 TIMR       *=*+1           The value of the interrupt
                                counter.
1170 *=$4000                    The program starts at 4000.
                                Of course, you may start it
                                anywhere.
1180            SEI             This 6502 command shuts down
                                all maskable interrupts to
                                the microprocessor.
1190            LDA CRA         Find the current value of
                                timer A's control register.
1200            AND #$FE        Reset low bit to zero.
1210            STA CRA         Storing this value in the
                                control register will turn
                                the timer off.
1220            LDA #$23
1230            STA TALO        Set timer A's latch value
                                (low byte).  This is the
                                number from which the timer
                                will count down before
                                interrupting.
1240            LDA #$08
1250            STA TAHI        Set timer A's latch value
                                (high byte).
1260            LDA CRA         Get the timer A control
                                register.
1270            ORA #$01        Set the low bit on.
1280            STA CRA         Start the timer going.
1290            LDA #<IRQST     Get low byte of interrupt
                                service routine start
                                address.
1300            STA $0314       Store in IRQ vector.
```

```
1310            LDA #>IRQST     Get high byte of interrupt
                                service routine start
                                address.
1320            STA $0315       Store in IRQ vector.
1330            LDY #$00        Set the Y register to zero.
1340            LDX #$18        The SID has $19 registers.
1350 LAB010
1360            LDA SIDDAT,X    Get a byte of the
                                initialization data for the
                                SID.
1370            STA SID,X       Store in the appropriate SID
                                register.
1380            DEX             Decrement X
1390            BPL LAB010      until it is $FF.
1400            LDA #$00
1410            STA TIMR        Initialize the interrupt
                                counter to zero.
1420            LDX #$02        There are 3 voices.
1430 LAB020
1440            STA DUR,X       Reset each of the durations
                                to zero.
1450            DEX             Decrement X
1460            BPL LAB020      until it is $FF.
1470            LDX #$05        There are 6 pieces of pointer
                                data (2 for each voice).
1480 LAB022
1490            LDA VOIADD,X    Get next pointer byte.
1500            STA PTRS,X      Store away in pointer table.
1510            DEX             Decrement X
1520            BPL LAB022      until it is $FF.
1530            CLI             This 6502 command enables
                                maskable interrupts to the
                                microprocessor.
1540 LAB024
1550            LDX #$02        There are 3 voices.
1560 LAB025
1570            STX VOICE       Save which voice we are on.
1580            LDA DUR,X       Get the number of duration
                                counts left.
1590            BEQ LAB030      When zero, it means we need a
                                new note for this voice.
1600            CMP #$03        Check if 3 duration counts
                                are left.
1610            BNE LAB050      If not, just keep on going.
```

```
1620            LDA SIDOFF,X      Need to gate off the SID.
                                  Get offset from SID origin
                                  for this voice.
1630            TAX               Put it in X.
1640            LDA SIDDAT+$04,X  Get waveform control byte for
                                  this voice.
1650            STA SID+$04,X     Store in appropriate SID
                                  register.
1660            BNE LAB040        This branch is always taken.
1670 LAB030
1680            TXA               Put voice number in
                                  accumulator.
1690            ASL A             Multiply by two.
1700            TAX               X now points to the pointer
                                  table.
1710            LDA PTRS,X        Get the low byte of the
                                  pointer for this voice.
1720            STA PTR           Store on zero page.
1730            LDA PTRS+1,X      Get the high byte.
1740            STA PTR+1         Store on zero page.
1750            LDA (PTR),Y       Y is zero, so we get the next
                                  note code for this voice.
1760            BMI LAB060        This signals the end of the
                                  song.  It will be encountered
                                  for voice 0 and is therefore
                                  not necessary for other
                                  voices.
1770            JSR GTFREQ        This subroutine calculates
                                  the frequency of the note,
                                  and sets Y to 1.  It is only
                                  called once, but doing so
                                  allows us to use branching
                                  (rather than jumping)
                                  throughout the code.
1780            LDA (PTR),Y       Y is one, so we get the
                                  duration for this note.
1790            STA DURA          Save the duration
                                  temporarily.
1800            DEY               Reset Y to zero.
1810            CLC               Clear the carry.
1820            LDA PTR           Get low byte of pointer.
1830            ADC #$02          Add two.
1840            STA PTRS,X        Store in the pointer table.
1850            BCC LAB035        Branch if no carry, else
```

```
1860            INC PTRS+1,X      increment high byte in the
                                  pointer table.
1870 LAB035
1880            LDX VOICE         X is which voice we're on.
1890            LDA DURA          Get the duration back.
1900            STA DUR,X         Store in duration table.
1910            LDA SIDOFF,X      Get the offset to the SID for
                                  this voice.
1920            TAX               X is offset to the SID.
1930            LDA FREQLO        Get the low byte of the
                                  frequency.
1940            STA SID+$00,X     Set low byte of the
                                  frequency.
1950            LDA FREQHI        Get the high byte of the
                                  frequency.
1960            STA SID+$01,X     Set the high byte of the
                                  frequency.
1970            LDA SIDDAT+$04,X  Get waveform control for this
                                  voice.
1980            ORA #$01          Set the gate bit.
1990            STA SID+$04,X     Set the waveform control for
                                  this voice (gate it on).
2000 LAB040
2010            LDX VOICE         Set X to the current voice.
2020 LAB050
2030            DEC DUR,X         Decrement the duration for
                                  this voice.
2040            DEX               Decrement voice number.
2050            BPL LAB025        Go back to check next voice.
2060            LDA #TEMPO        Load tempo byte (from data
                                  file).
2070            STA TIMR          Start timer counting down.
2080 LAB055
2090            LDA TIMR          Check for timer zero.
2100            BNE LAB055        Branch back until done.
2110            BEQ LAB024        Always taken to start
                                  checking voices again.
2120 LAB060
2130            LDA #$00          Shut down the SID at the end.
2140            STA SID+$04       Voice 1 control.
2150            STA SID+$0B       Voice 2 control.
2160            STA SID+$12       Voice 3 control.
2170            JMP RESVEC        Jump to system reset vector.
                                  This will reset the timer and
```

```
                                interrupt vector values to
                                the system norm.
2180; ******************************************************
2190 IRQST
2200            LDA TIMR        Every interrupt, load timer
                                value.
2210            BEQ IRQ010      Take no action if zero.
2220            DEC TIMR        Decrement the timer.
2230 IRQ010
2240            JMP IRQVEC      Jump to the system interrupt
                                service routine.
2250 ;
2260 GTFREQ
2270            STA NOTE        Save the note code.
2280            AND #$0F        Get the note value.
2290            ASL A           Multiply by two.
2300            TAY             Transfer to Y register.
2310            LDA FRQTAB,Y    Get the low byte of the base
                                frequency for this note.
2320            STA FREQLO      Save it.
2330            LDA FRQTAB,Y    Get the high byte of the base
                                frequency for this note.
2340            STA FREQHI      Save it.
2350            LDA NOTE        Get the note code back, and
2360            LSR A           strip off low four bits to
2370            LSR A           get octave.
2380            LSR A
2390            LSR A
2400            TAY             Now, Y is the octave.
2410            BEQ GTL02       If highest octave, no
                                division is necessary, so
                                branch around.
2420 GTL01
2430            LSR FREQHI      Divide the frequency by two.
2440            ROR FREQLO
2450            DEY             Decrement octave counter.
2460            BNE GTL01       Branch back until done.
2470 GTL02
2480            INY             Y will now be one.
2490            RTS             Return to main program.
2500 ;
2510 SIDOFF
2520 .BYTE $00,$07,$0E
2530 FRQTAB
```

```
2540 .BYTE 30,134,24,142,139,150
2550 .BYTE 126,159,250,168,6,179
2560 .BYTE 172,189,243,200,230,212
2570 .BYTE 143,225,248,238,46,253
2580 .BYTE 0,0
2590 VOIADD
2600 .BYTE <VOI1,>VOI1,<VOI2,>VOI2,<VOI3,>VOI3
2610 .END
```

Data File

```
1000 .PAG 'CANON'
1010 ; CANON IN D
1020 ; BY JOHANN PACHELBEL
1030 .OPT NOLIST
1040 TEMPO=$0D
1050 .LIB MUSPRG
1060 SIDDAT
1070 .BYTE $00,$00,$00,$07,$40,$09,$00
1080 .BYTE $00,$00,$00,$08,$40,$95,$61
1090 .BYTE $00,$00,$00,$00,$20,$53,$71
1100 .BYTE $00,$00,$00,$0F
1110 VOI1
1120 .BYTE $0C,$10,$36,$10,$39,$10,$22,$10          ;01
1130 .BYTE $0C,$10,$34,$10,$39,$10,$21,$10
1140 .BYTE $0C,$10,$32,$10,$36,$10,$3B,$10          ;02
1150 .BYTE $0C,$10,$31,$10,$36,$10,$39,$10
1160 .BYTE $0C,$10,$4B,$10,$32,$10,$37,$10          ;03
1170 .BYTE $0C,$10,$36,$10,$39,$10,$22,$10
1180 .BYTE $0C,$10,$34,$10,$3B,$10,$22,$10          ;04
1190 .BYTE $0C,$10,$34,$10,$39,$10,$21,$10
1200 .BYTE $0C,$10,$36,$10,$39,$10,$22,$10          ;05
1210 .BYTE $0C,$10,$34,$10,$39,$10,$21,$10
1220 .BYTE $0C,$10,$32,$10,$36,$10,$3B,$10          ;06
1230 .BYTE $0C,$10,$31,$10,$36,$10,$39,$10
1240 .BYTE $0C,$10,$4B,$10,$32,$10,$37,$10          ;07
1250 .BYTE $0C,$10,$36,$10,$39,$10,$22,$10
1260 .BYTE $0C,$10,$34,$10,$3B,$10,$22,$10          ;08
1270 .BYTE $0C,$10,$34,$10,$39,$10,$21,$10
1280 .BYTE $0C,$10,$36,$10,$39,$10,$22,$10          ;09
1290 .BYTE $0C,$10,$31,$10,$34,$10,$39,$10
1300 .BYTE $0C,$10,$32,$10,$36,$10,$3B,$10          ;10
1310 .BYTE $0C,$10,$31,$10,$36,$10,$39,$10
```

the arts

```
1320 .BYTE $0C,$10,$4B,$10,$32,$10,$37,$10          ;11
1330 .BYTE $0C,$10,$36,$10,$39,$10,$22,$10
1340 .BYTE $0C,$10,$34,$10,$37,$10,$22,$10          ;12
1350 .BYTE $0C,$10,$34,$10,$39,$10,$21,$10
1360 .BYTE $0C,$20,$36,$20,$0C,$20,$31,$20          ;13
1370 .BYTE $0C,$20,$36,$20,$0C,$20,$31,$20          ;14
1380 .BYTE $0C,$20,$4B,$20,$0C,$20,$39,$20          ;15
1390 .BYTE $0C,$20,$4B,$20,$0C,$20,$31,$20          ;16
1400 .BYTE $0C,$20,$36,$20,$0C,$20,$34,$20          ;17
1410 .BYTE $0C,$20,$32,$20,$0C,$20,$31,$20          ;18
1420 .BYTE $0C,$20,$4B,$20,$0C,$20,$39,$20          ;19
1430 .BYTE $0C,$20,$32,$20,$0C,$20,$31,$20          ;20
1440 .BYTE $0C,$10,$36,$10,$39,$20                  ;21
1450 .BYTE $0C,$10,$31,$10,$39,$20
1460 .BYTE $0C,$10,$36,$10,$3B,$20                  ;22
1470 .BYTE $0C,$10,$49,$10,$36,$20
1480 .BYTE $0C,$10,$32,$10,$37,$10,$32,$10          ;23
1490 .BYTE $0C,$10,$36,$10,$39,$10,$36,$10
1500 .BYTE $0C,$10,$32,$10,$37,$20                  ;24
1510 .BYTE $0C,$10,$31,$10,$34,$20
1520 .BYTE $0C,$20,$36,$20,$0C,$20,$32,$20          ;25
1530 .BYTE $0C,$20,$32,$20,$0C,$20,$49,$20          ;26
1540 .BYTE $0C,$20,$32,$20,$0C,$20,$36,$20          ;27
1550 .BYTE $0C,$10,$32,$10,$37,$10,$32,$10          ;28
1560 .BYTE $0C,$10,$34,$10,$37,$10,$34,$10
1570 .BYTE $0C,$08,$36,$08,$39,$08,$36,$08          ;29
1580 .BYTE $39,$08,$36,$08,$39,$08,$36,$08
1590 .BYTE $0C,$08,$34,$08,$37,$08,$34,$08
1600 .BYTE $37,$08,$34,$08,$37,$08,$34,$08
1610 .BYTE $0C,$08,$32,$08,$36,$08,$32,$08          ;30
1620 .BYTE $36,$08,$32,$08,$36,$08,$32,$08
1630 .BYTE $0C,$08,$49,$08,$32,$08,$49,$08
1640 .BYTE $32,$08,$49,$08,$32,$08,$49,$08
1650 .BYTE $0C,$08,$4B,$08,$32,$08,$4B,$08          ;31
1660 .BYTE $32,$08,$4B,$08,$32,$08,$4B,$08
1670 .BYTE $0C,$08,$36,$08,$39,$08,$36,$08
1680 .BYTE $39,$08,$36,$08,$39,$08,$36,$08
1690 .BYTE $0C,$10,$32,$10,$37,$10,$32,$10          ;32
1700 .BYTE $0C,$10,$34,$10,$37,$10,$34,$10
1710 .BYTE $37,$20,$36,$10,$34,$10,$36,$40          ;33
1720 .BYTE $FF
1730 VOI2
1740 .BYTE $32,$40,$49,$40                          ;01
1750 .BYTE $4B,$40,$46,$40                          ;02
1760 .BYTE $47,$40,$32,$40                          ;03
```

```
1770 .BYTE $47,$40,$49,$40                                  ;04
1780 .BYTE $32,$40,$49,$40                                  ;05
1790 .BYTE $4B,$40,$46,$40                                  ;06
1800 .BYTE $47,$40,$32,$40                                  ;07
1810 .BYTE $47,$40,$49,$40                                  ;08
1820 .BYTE $32,$40,$49,$40                                  ;09
1830 .BYTE $4B,$40,$46,$40                                  ;10
1840 .BYTE $47,$40,$32,$40                                  ;11
1850 .BYTE $47,$40,$49,$40                                  ;12
1860 .BYTE $32,$40,$49,$40                                  ;13
1870 .BYTE $4B,$40,$46,$40                                  ;14
1880 .BYTE $47,$40,$32,$40                                  ;15
1890 .BYTE $47,$40,$49,$40                                  ;16
1900 .BYTE $32,$40,$49,$40                                  ;17
1910 .BYTE $4B,$40,$46,$40                                  ;18
1920 .BYTE $47,$40,$32,$40                                  ;19
1930 .BYTE $47,$40,$49,$40                                  ;20
1940 .BYTE $32,$40,$49,$40                                  ;21
1950 .BYTE $4B,$40,$46,$40                                  ;22
1960 .BYTE $47,$40,$32,$40                                  ;23
1970 .BYTE $47,$40,$49,$40                                  ;24
1980 .BYTE $32,$40,$49,$40                                  ;25
1990 .BYTE $4B,$40,$46,$40                                  ;26
2000 .BYTE $47,$40,$32,$40                                  ;27
2010 .BYTE $47,$40,$49,$40                                  ;28
2020 .BYTE $32,$40,$49,$40                                  ;29
2030 .BYTE $4B,$40,$46,$40                                  ;30
2040 .BYTE $47,$40,$32,$40                                  ;31
2050 .BYTE $47,$40,$49,$40                                  ;32
2060 .BYTE $32,$80                                          ;33
2070 VOI3
2080 .BYTE $0C,$80                                          ;01
2090 .BYTE $0C,$80                                          ;02
2100 .BYTE $0C,$80                                          ;03
2110 .BYTE $0C,$80                                          ;04
2120 .BYTE $26,$40,$24,$40                                  ;05
2130 .BYTE $22,$40,$21,$40                                  ;06
2140 .BYTE $3B,$40,$39,$40                                  ;07
2150 .BYTE $3B,$40,$21,$40                                  ;08
2160 .BYTE $22,$20,$26,$20,$29,$20,$27,$20                  ;09
2170 .BYTE $26,$20,$22,$20,$26,$20,$24,$20                  ;10
2180 .BYTE $22,$20,$3B,$20,$22,$20,$29,$20                  ;11
2190 .BYTE $27,$20,$2B,$20,$29,$20,$27,$20                  ;12
2200 .BYTE $26,$10,$21,$10,$22,$10,$32,$10                  ;13
2210 .BYTE $31,$10,$39,$10,$34,$10,$36,$10
```

```
2220 .BYTE $32,$10,$22,$10,$21,$10,$3B,$10          ;14
2230 .BYTE $39,$10,$36,$10,$39,$10,$3B,$10
2240 .BYTE $37,$10,$36,$10,$34,$10,$37,$10          ;15
2250 .BYTE $36,$10,$34,$10,$32,$10,$31,$10
2260 .BYTE $4B,$10,$39,$10,$37,$10,$36,$10          ;16
2270 .BYTE $34,$10,$37,$10,$36,$10,$34,$10
2280 .BYTE $22,$10,$26,$08,$27,$08                  ;17
2290 .BYTE $29,$10,$26,$08,$27,$08
2300 .BYTE $29,$08,$39,$08,$3B,$08,$21,$08
2310 .BYTE $22,$08,$24,$08,$26,$08,$27,$08
2320 .BYTE $26,$10,$22,$08,$24,$08                  ;18
2330 .BYTE $26,$10,$26,$08,$27,$08
2340 .BYTE $29,$08,$2B,$08,$29,$08,$27,$08
2350 .BYTE $29,$08,$26,$08,$27,$08,$29,$08
2360 .BYTE $27,$10,$2B,$08,$29,$08                  ;19
2370 .BYTE $27,$10,$26,$08,$24,$08
2380 .BYTE $26,$08,$24,$08,$22,$08,$24,$08
2390 .BYTE $26,$08,$27,$08,$29,$08,$2B,$08
2400 .BYTE $27,$10,$3B,$08,$39,$08                  ;20
2410 .BYTE $3B,$10,$21,$08,$22,$08
2420 .BYTE $39,$08,$3B,$08,$21,$08,$22,$08
2430 .BYTE $24,$08,$26,$08,$27,$08,$29,$08
2440 .BYTE $26,$30,$26,$10                          ;21
2450 .BYTE $26,$10,$27,$10,$26,$10,$24,$10
2460 .BYTE $22,$30,$22,$10                          ;22
2470 .BYTE $22,$10,$24,$10,$22,$10,$21,$10
2480 .BYTE $3B,$40,$22,$40                          ;23
2490 .BYTE $22,$10,$20,$10,$3B,$10,$20,$10          ;24
2500 .BYTE $39,$30,$39,$10
2510 .BYTE $22,$20,$0C,$10,$29,$10                  ;25
2520 .BYTE $29,$10,$2B,$10,$29,$10,$27,$10
2530 .BYTE $26,$30,$26,$10                          ;26
2540 .BYTE $26,$10,$27,$10,$26,$10,$24,$10
2550 .BYTE $22,$10,$20,$10,$3B,$10,$20,$10          ;27
2560 .BYTE $39,$30,$39,$10
2570 .BYTE $3B,$20,$22,$20,$21,$30,$21,$10          ;28
2580 .BYTE $22,$30,$29,$10                          ;29
2590 .BYTE $29,$10,$2B,$10,$29,$10,$27,$10
2600 .BYTE $26,$30,$26,$10                          ;30
2610 .BYTE $26,$10,$27,$10,$26,$10,$24,$10
2620 .BYTE $22,$40,$29,$20,$22,$20                  ;31
2630 .BYTE $3B,$40,$21,$40                          ;32
2640 .BYTE $22,$80                                  ;33
2650 .END
```

Note Code for the Commodore 64

Robert A. Embree

Composers of music on the Commodore 64 can use this program in conjunction with Lee Silvan's "Complex Rhythms" program (May, 1983, Commodore*) to avoid having to look up the duration and note values for every note in their composition.*

The Commodore 64 music program discussed by Lee Silvan in the May issue of *Commodore* magazine demonstrates admirably some of the great potential of the Commodore 64's sound chip. The single number code developed by Cyndie Merten really simplifies control of the computer. But converting each note into that code is a tedious task. The program below makes that work much easier.

With the Note Code program it is possible to work directly with the music score without having to look up the appropriate duration and note values. More importantly, the program creates music files that can be used in conjunction with the Silvan music program if slightly modified.

The Note Code program sets up a file that holds the data for the three voices. Hence it can be used as often as you like, but avoid using the same file name, because the previous data will be lost if you do. The program displays a menu that prompts input for the three bits of information required for the note data, namely note duration, octave and note. After the melody has been completed, end that data with zero. After data for the second voices have been coded, again end with zero. The same procedure is used with the third voice. With the data for the three voices coded, the program automatically stores your work. A new file can now be created, or you can exit the program. After a little practice you will be able to easily convert notes to code for the Silvan "Complex Rhythms" program.

Note Code

```
5 REM NOTE CODE PROGRAM
10 DIM D(11),N(50),DA(500)
15 FOR I=1 TO 11
20 READ D(I)
25 NEXT I
30 FOR I=1 TO 49
35 READ N(I)
40 NEXT I
45 K=0:C=0:XC=0
50 PRINT CHR$(147)
51 REM FILE NAME FOR MUSIC
52 PRINT"[SPACE3]USE[SPACE]'*'[SPACE]TO[SPACE]EXIT"
53 INPUT"ENTER[SPACE]DATA[SPACE]FILE[SPACE]NAME";F$
54 IF F$="*"THEN 600
55 PRINT CHR$(147)
60 PRINT"(1)[SPACE]1/16"
```

```
61 PRINT"(2)[SPACE]1/8"
62 PRINT"(3)[SPACE]1/8+1/16[SPACE]HOLD"
63 PRINT"(4)[SPACE]1/4"
64 PRINT"(5)[SPACE]1/4+1/16[SPACE]TIE"
65 PRINT"(6)[SPACE]1/4+1/8[SPACE]HOLD"
66 PRINT"(7)[SPACE]1/2"
67 PRINT"(8)[SPACE]1/2+1/16[SPACE]TIE"
68 PRINT"(9)[SPACE]1/2+1/8[SPACE]TIE"
69 PRINT"(10)[SPACE]1/2+1/4[SPACE]HOLD"
70 PRINT"(11)[SPACE]1/1"
80 PRINT"---------------------------------------"
149 PRINT"**0[SPACE]TO[SPACE]STORE[SPACE]DATA[SPACE]**
    [SPACE]-1[SPACE]RESTART":PRINT"***[SPACE]-2[SPACE]TO
    [SPACE]EXIT[SPACE]PROGRAM"
150 INPUT"DURATION[SPACE]OF[SPACE]THE[SPACE]NOTE";DR
151 IF DR=0 THEN 200
152 IF DR=-1 THEN 45
153 IF DR=-2 THEN 600
154 PRINT CHR$(147)
155 PRINT"OCTAVE[SPACE]1[SPACE]G-B[SPACE](LOW[SPACE]BASE)"
156 PRINT"OCTAVE[SPACE]2[SPACE]C-B[SPACE](BELOW[SPACE]
    MIDDLE[SPACE]C)"
157 PRINT"OCTAVE[SPACE]3[SPACE]C-B[SPACE](MIDDLE[SPACE]C)"
158 PRINT"OCTAVE[SPACE]4[SPACE]C-B[SPACE](ABOVE[SPACE]
    MIDDLE[SPACE]C)"
159 PRINT"OCTAVE[SPACE]5[SPACE]C[SPACE3](HIGH[SPACE]C
    [SPACE]ONLY)"
160 PRINT"---------------------------------------"
161 INPUT"OCTAVE[SPACE]OF[SPACE]NOTE";O
162 PRINT CHR$(147)
165 PRINT"C=1[SPACE]C#=2[SPACE]D=3[SPACE]D#=4[SPACE]E=5
    [SPACE]F=6[SPACE]F#=7"
166 PRINT"G=8[SPACE]G#=9[SPACE]A=10[SPACE]A#=11[SPACE]
    B=12"
167 PRINT
168 INPUT"NOTE";NT
170 J=((O*12)-12)+NT
171 IF J<7 OR J>49 THEN 550
172 PRINT CHR$(147):PRINT"PRESS[SPACE]RETURN[SPACE]TO
    [SPACE]CONTINUE[SPACE]OR[SPACE]R[SPACE]TO[SPACE]REDO"
173 PRINT"NOTE[SPACE]NUMBER"XC+1:PRINT"DURATION=";
    DR"OCTAVE="O;"NOTE="NT
174 GET L$:IF L$=""THEN 174
175 IF L$="R"THEN 55
176 K=K+1:XC=XC+1
```

```
180 DA(K)=N(J)+D(DR)
190 GOTO 55
200 C=C+1:K=K+1
201 XC=0
202 DA(K)=0
203 IF C<3 THEN 55
204 REM STORES NOTES FOR THREE VOICES
205 OPEN 5,8,5,"@0:"+F$+",S,W"
210 FOR I=1 TO K
220 PRINT#5,DA(I)
230 NEXT I
235 PRINT#5,"*"
240 CLOSE 5
250 GOTO 45
500 DATA 128,256,384,512,640,768,1024,1152,1280,1536,2048
505 DATA 0,0,0,0,0,0,0
510 DATA 39,40,41,42,43,48,49,50,51,52,53,54,55,56,57,58,
    59
520 DATA 64,65,66,67,68,69,70,71,72,73,74,75
530 DATA 80,81,82,83,84,85,86,87,88,89,90,91,96
550 PRINT CHR$(147)
551 PRINT"******ERROR[SPACE]IN[SPACE]DATA[SPACE]
    ENTRY******"
560 PRINT"ENTER[SPACE]DATA[SPACE]AGAIN"
570 FOR JN=1 TO 800:NEXT
580 GOTO 55
600 END
```

Before you can use your music files, it will be necessary to make a few modifications in the Silvan program (see pages 24 and 25 in the May, 1983, *Commodore* magazine or page 32, this issue). Simply add the changes below and then run the program. In a few seconds you will hear the coded music of whatever file you select.

Silvan Change

```
5 DIM NM(600)
6 INPUT"MUSIC[SPACE]FILE[SPACE]NAME";F$
70 OPEN 5,8,5,F$+",S,R"
71 P=P+1
72 INPUT#5,NM$:NM(P)=VAL(NM$)
73 IF NM$="*"THEN 76
75 GOTO 71
76 CLOSE 5
120 R=R+1
121 NM=NM(R)
```

Lee Silvan's Program

Use this program from the May issue as a reference for Robert Embree's "Note Code" article in this issue.

```
10 S=54272:FOR L=S TO S+24
   :POKE L,0:NEXT
20 DIM H(2,200),L(2,200),C(2,200)
30 DIM FQ(11)
40 V(0)=17:V(1)=65:V(2)=33
50 POKE S+10,3:POKE S+22,240
   :POKE S+23,244
60 FOR I=0 TO 11:READ FQ(I):NEXT
100 FOR K=0 TO 2
110 I=0
120 READ NM
130 IF NM=0 THEN 250
140 WA=V(K):IF NM<0 THEN NM=-NM
    :WA=1
150 DR%=NM/128:OC%=(NM AND 112)/
    16
160 NT=NM AND 15
170 FR=FQ(NT)
175 IF K=0 THEN OC%=OC%+1
180 IF OC%=7 THEN 200
190 FOR J=6 TO OC%STEP-1:FR=FR/2
    :NEXT
200 HF%=FR/256:LF%=FR AND 255
210 IF DR%=1 THEN H(K,I)=HF%
    :L(K,I)=LF%:C(K,I)=WA:I=I+1
    :GOTO 120
220 FOR J=1 TO DR%-1:H(K,I)=HF%
    :L(K,I)=LF%:C(K,I)=WA:I=I+1
    :NEXT
230 H(K,I)=HF%:L(K,I)=LF%
    :C(K,I)=WA-1
240 I=I+1:GOTO 120
250 IF I>IM THEN IM=I
260 NEXT
500 POKE S+5,60:POKE S+6,128
510 POKE S+12,10:POKE S+13,12
520 POKE S+19,10:POKE S+20,11
530 POKE S+24,31
540 FOR I=0 TO IM
550 POKE S,L(0,I):POKE S+7,L(1,I)
    :POKE S+14,L(2,I)
560 POKE S+1,H(0,I)
    :POKE S+8,H(1,I)
    :POKE S+15,H(2,I)
570 POKE S+4,C(0,I)
    :POKE S+11,C(1,I)
    :POKE S+18,C(2,I)
580 FOR T=1 TO 10 :NEXT
585 NEXT
590 GOTO 540
600 DATA 34334,36376,38539,40830
610 DATA 43258,45830,48556,51443
620 DATA 54502,57743,61176,64814
```

Just for you and your hi-res...

A Lightning-Fast Machine Language Joystick

By Danny Byrne

Write the capability for high-resolution joystick drawing into your programs using this machine language routine for the Commodore 64. It's FAST.

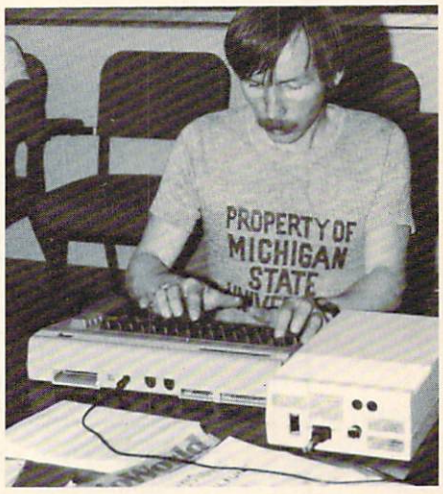
The Commodore 64 has awesome graphics capabilities. I have never had anyone disagree with that statement, no matter what type of computer they happened to own. All that jealousy can be very gratifying, and it's lots of fun to brag about 320 by 200 resolution, sprite graphics, 16 colors, screen splitting, and so on. But when I got my 64, in November of 1982, there wasn't a lot of information available on just *how* to utilize all those goodies. The high resolution (hi-res) capabilities were my first love, so I decided that was the place to start experimenting, and needless to say, it took me a while to get the hang of it!

First, let me give you some facts and figures pertaining to the way hi-res works on the Commodore 64. To program the 64 in hi-res, we use a method called bit mapping. Each individual dot (or pixel) has a corresponding bit in memory. With a resolution of 320 dots by 200 dots (320*200) this gives you a total of 64,000 dots to keep track of, which uses up a sizable chunk of memory—8K of the 39K available for programming in BASIC. But that's not all. The first time I tried one of my BASIC joystick routines—one that works just fine in low-res—I had enough time to take a nap while it drew a line from one side of the screen to the other!

I came to the frightening conclusion that I was going to have to learn machine language to be able to program effectively using my 64's hi-res capabilities! Why did I find this frightening? You have to remember that at the time, not even the *Programmer's Reference Guide* was available, and I thought that ML was a mystical secret code, understood only by those legendary programmers who had mastered their trade when a computer was something that took up an entire floor. I was sure that at the very least a master's degree in computer science was necessary!

Needless to say, I did find out that machine language programming *wasn't* that tough, and for anyone starting along the same road today (less than a year later) there are some excellent aids available, one of which, Commodore's assembler package, I have found indispensable.

The first month of my ML odyssey was spent peering red-eyed



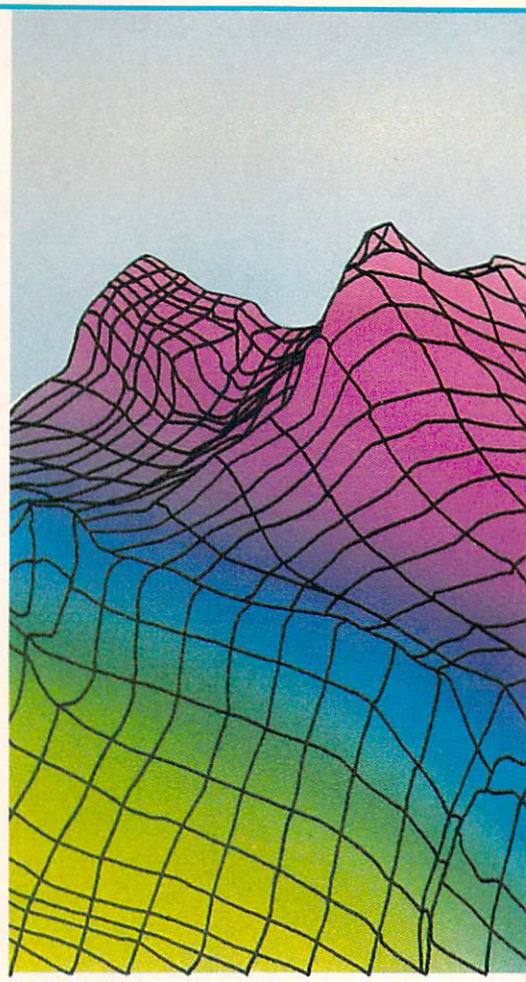
Danny Byrne

into a machine language monitor for more hours than I care to remember. No wonder, then, that I sing the praises of the assembler, one of the most difficult parts of which is thinking up nifty names for labels! The hi-res ML joystick routine that follows was begun on the monitor, but really came into its own when I learned assembly. I would like to thank Jim Butterfield for his patience in answering rather frantic long-distance questions on how the joysticks operate on the 64. Not too many people are willing to put up with beginners in the throes of the self-teaching experience!

There are two joystick ports on the Commodore 64 (56321=port A and 56320=port B) that receive data from the joystick, determined by the direction in which you

continued on page 60

SUPER EXPANDER 64



By Stephen Murri

Finally—the cartridge you've all been waiting for! Super Expander 64 *is an extension to BASIC that provides easy access to the many graphics and sound features of the Commodore 64. Even beginners can create exciting visuals and sound effects using sprite graphics, programmable characters and all three voices of the 64's SID chip.*

So you're the proud owner of a Commodore 64 and you've been told that it's the most graphics oriented home computer on the market. You've also been informed that the machine has a sound synthesizer chip so sophisticated that it can reproduce sounds ranging from a romantic violin to the thunderous collisions in a super galactic space battle.

You may think, however, that the use of these advanced features is limited only to the elite group of programmers who know how to use the bits and bytes of the special purpose SID and VIC chips. Well, you're wrong on that one! Commodore's new *Super Expander 64* cartridge puts all the sound and graphics power of the Commodore 64 at your fingertips —and easily, too. Within minutes after plugging the *Super Expander 64* cartridge into your machine, you will be able to experiment with sound, graphics and the many other powerful features of the Commodore 64 that previously took hours, even days, of programming time.

Super Expander 64, code-named VSP (for Video Support Package), was developed internally at Commodore's MOS Technology Division. It is a powerful extension to the BASIC language in the Commodore 64 computer. *Super Expander 64* provides the user with new BASIC commands that allow easy access to the advanced sound and graphics features of the Commodore 64. When the *Super Expander 64* cartridge is plugged into the computer, the user has the ability to:

- Create high resolution and multi-color high resolution displays
- Create and animate sprites
- Create and save shapes and images
- Fill these shapes with any of the sixteen colors
- Draw points, lines, arcs, circles and ellipses
- Draw polygons such as rectangles, triangles and octagons
- Combine text with high resolution displays
- Read game paddle, joystick and light pen positions
- Create music and game sounds
- Define programmable function keys
- Detect collisions between sprite-to-sprite and sprite-to-background data

Most impressive of all is the small amount of memory taken up by the *Super Expander 64* program. Through the use of efficient memory management techniques,

all of these features are provided at a loss of only 8192 of the free bytes available for BASIC programming. This is no minor accomplishment since it takes 8000 bytes alone to bit-map a high resolution screen on the Commodore 64.

Equally impressive is the ease with which these features are implemented. Even users with little artistic, musical or programming background will be able to incorporate bright, colorful, animated graphics and interesting sound effects into their BASIC programs.

Super Expander 64 Applications

The practical uses of Commodore's *Super Expander 64* package span a broad range of applications. First and foremost are the business applications. With business graphics the goal is communication and it has been proven that computer graphics are an extremely effective communication tool. The human eye can absorb the information in a diagram or perspective view much faster than it can scan a table of numbers.

Super Expander 64 allows for the easy creation of bar charts, pie charts, line graphs and just about any other business graphic that can get your message across with polish. Many of its features make it an ideal tool for developing specialized custom applications including logic schematics, mechanical engineering drawings and architectural drawings, to name but a few.

Super Expander 64 can also be used in many scientific applications, including the plotting of complex graphs, mathematical functions and other scientific data. Using *Super Expander 64*, your Commodore 64 can read in data from the outside world (through the use of an A/D converter) and graphically display the results. It can also

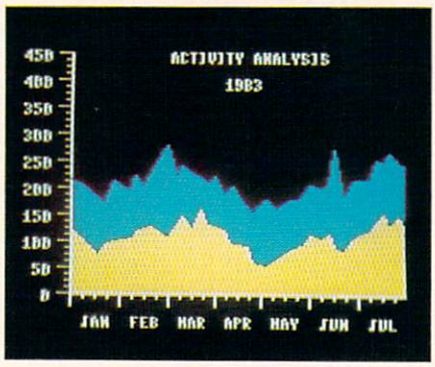
generate and manipulate contour maps and other three-dimensional graphic images similar to those developed on powerful mini-computer-based image-processing systems. This would normally be no easy task, since it means managing a complex set of data (using geometry and trig) to compensate for angle of view, hidden lines, image transformations, etc.

Another application, suited to home use, is the creation of interactive video games. Since *Super Expander 64* is essentially a graphics package that also provides for the generation of sound and the reading of input devices, it is an ideal medium for the development of interactive video games. With *Super Expander 64*, you can use sprites and high resolution graphics to create a cast of characters and colorful settings. The sprites can be moved around the screen based on the position of the user's joystick or paddle. You can also take advantage of the Commodore 64's unique ability to detect collisions between sprites and background. (This is obviously an extremely powerful feature in the production of video action games.) Finally, as a finishing touch, sound effects and music can be added to complement the game play.

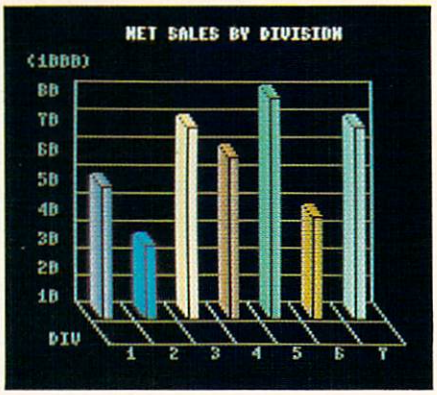
Super Expander 64 can also be used to create development tools, which are packages of functions or subroutines to support the writing of other application programs. Scientists Frank Covitz and Cliff Ashcraft, for instance, are using *Super Expander 64* to assist in the development of a Commodore product due to be released at the end of this year: the *Home Planetarium* for the Commodore 64. They are creating a graphics editor, which they will use to "fine-tune" high resolution screens generated on other graphic systems and converted to the Commodore 64. Their program will read the location of the joystick in order to move a small cursor around the screen. After the cursor is maneuvered to the desired location, the fire button is pressed to turn a pixel off or on. Other features may be added that will allow Covitz and Ashcraft to manipulate blocks of screen data, change colors, etc. You may recognize these features as those available in the basic "paint" packages for various microcomputers. As you can see, *Super Expander 64* can easily be used to create a powerful, full-featured paint package for the Commodore 64.

In the educational software arena, the *Super Expander 64* applications are almost limitless. It can be easily used to teach music fundamentals, generate geometrical images and otherwise enhance just about any educational program that an instructor can dream up.

In general, *Super Expander 64* can be used to improve the appearance and sophistication of ANY program on the Commodore 64, whether it be oriented toward education, entertainment or business. Any program with good graphics and sound always looks "slick" and it is this added degree of professionalism that always makes your good idea look better.



These screens demonstrate the various applications of Super Expander 64. This colorful line graph demonstrates use of the DRAW and FILL commands. It also demonstrates Super Expander's ability to mix text on a high resolution screen. Multi-Color mode was required in order to mix the various colors.



Multi-Color mode was also used in the generation of this bar chart. Programmable characters were used to build the vertical bars, and the DRAW command was used to generate the calibration lines.

Graphics on the Commodore 64

We are now ready to discuss how *Super Expander 64* is used, but before we continue, there are a few things you will need to know about how graphics are organized on the Commodore 64. There are a number of different graphics modes on the Commodore 64, but we will limit our discussion to the major four used in *Super Expander 64:* text mode, high-resolution mode, multi-color mode, and split-screen mode. A general understanding of these modes is required in order to fully utilize the power and flexibility of *Super Expander 64*. The concepts, however, are very simple.

In text mode, the screen is mapped as it is when you first power up your Commodore 64. There are 25 rows of 40-column ($40 \times 25 = 1000$) character "cells". Each cell consists of an eight-by-eight dot matrix and there can be only two colors in each character cell: foreground and background. Don't be confused by this two-color limit. All sixteen colors can be displayed on the screen at the same time, but only two colors can be in each eight-by-eight character cell.

In high-resolution mode the screen is mapped in 200 rows of

320 dots or "pixels" (short for picture elements). Each of the pixels is internally represented within the machine by a bit, which is either off or on. This is where the term "bit-mapped" graphics comes from. In high-resolution mode, the user has full control over each pixel on the screen. However, the color is handled similarly to text mode: each eight-by-eight character cell can have only two colors: foreground and background. Foreground color is displayed for the bits that are on and background color is displayed for the bits that are off.

In multi-color mode the screen is mapped in 200 rows of 160 pixels and, as with high resolution mode, the user has full control over the entire bit map. The only difference is that horizontal resolution is cut in half (from 320 to 160). The horizontal resolution is sacrificed for increased color capability; each eight-by-eight character cell can now have up to four colors: foreground, multi-color1, multi-color2 and background.

Split-screen mode is a combination of high resolution and text. When in split-screen mode, the top (320×160) portion of the screen is in high-resolution mode, which leaves room for a five-line text "window" at the bottom of the screen. This is a handy feature well suited for debugging programs or interactive applications requiring a high-res display and an area for text.

Resolution

You're probably asking what "a decrease in horizontal resolution" means to you and, also, when you should use high-resolution mode over multi-color mode. The answer depends on what you are trying to achieve. Resolution determines the precision of a display. It is the number of visibly distinct dots that can be displayed in a given area of the screen. The higher the resolution in a display the smoother your graphics will look. The lower the resolution the more your display will suffer from distortions known as the "jaggies"—the staircase-like effect you see in circles or lines that should be smooth but instead appear to zigzag.

Technically, the jaggies are an effect known as "aliasing". When straight lines cannot be drawn because of resolution limitations, they must be "approximated" as best as possible. Thus, when you draw lines in multi-color mode (160×200 resolution), the approximations or jaggies will be much more evident than if you were drawing in the much finer high-resolution mode (320×200 resolution).

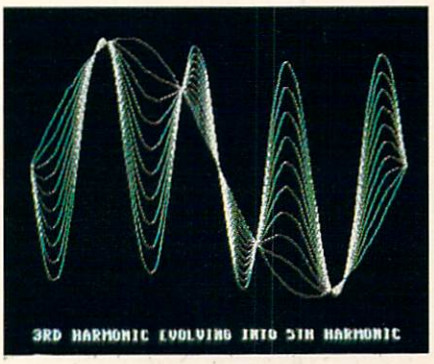
High-Resolution Versus Multi-Color

It would then seem logical that when we desire greater precision and fineness of detail, we should use high-resolution mode and when the emphasis of our display is on color or the mixing of colors, we should then switch to multi-color mode. This is the general rule of thumb but sometimes there is no easy answer to the high-resolution vs. multi-color decision. It all boils down to a matter of personal taste and judgement.

For example, when I attempted to produce a colorful pie chart using *Super Expander 64*, I initially thought that multi-color mode was the only way to go. In a multi-colored pie chart, all the colors of the pie pieces meet in the center and this obviously meant I would end up with more than one foreground color in the same eight-by-eight character cell. Multi-color seemed like the only viable option until I saw the display. The pie chart in multi-color mode was extremely distorted by the staircase effect of the jaggies. It just didn't look slick enough.

I finally solved the problem by re-creating my pie chart in high-resolution mode. I simply separated the multi-colored pie pieces so there were never more than two foreground colors in the same eight-by-eight character cell. The colors were vibrant and the precision was excellent. The added resolution indeed did the trick.

But, in spite of this high-resolution triumph, there are many situations in which multi-color mode is the better option. Whenever you want to accentuate or define your form using color, multi-color mode should be tried. Just use your own taste and judgement and take it from there.



In this picture, a three-dimensional sound waveform was created by analyzing sound data and plotting the results using the DRAW command. For maximum smoothness, High Resolution mode was used.

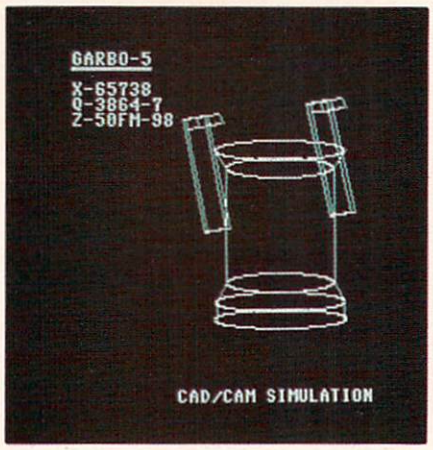
Sprite Graphics

So far, it would seem that *Super Expander 64* is quite an extensive package, but there's still more. How could Commodore release any graphics package without including those lovable little creatures we have affectionately come to know as sprites?

Sprites are 21 × 24 graphic images that you design and move anywhere on the screen. They are especially suited for video graphics and arcade-type animation. Up to eight sprites can be on the screen at any given time and they can be single- or multi-colored.

Super Expander 64 features a sprite-designer mode that allows for easy definition of sprite images. The user has full control over all sprite parameters such as color, expansion, multi-color mode, priorities, etc. After designing the sprite(s), the SPRITE, SPRSAV and MOVSPR commands are used to maintain characteristics, animate and save the sprites for later use in other programs.

There is even a feature that allows you to detect collisions of sprite-to-sprite and sprite-to-background data. This is accomplished through use of the COLINT command. The COLINT command references a line number in your program that refers to an "interrupt" subroutine. Whenever a collision is detected, your normal BASIC code is "interrupted" and program control is passed to the line number of your interrupt subroutine. This is the same theory used in programming the Commodore 64 video action games. The COLINT command can also be used to manage input from a light pen.



This screen was created for a recent Commodore 64 television commercial. The robot was created using the SCALE and DRAW commands. Again, High Resolution mode was used for maximum fineness and detail.

Game Controls and Function Keys

Another useful feature in the development of interactive programs or video games is *Super Expander's* ability to read the Commodore 64 game controls and function keys. Three functions are provided to let the user easily read the positions of one or two joysticks, up to four game paddles or light pen coordinates from the game control ports. These include the RJOY function for joysticks, RPOT for game paddles and RPEN for the light pen.

Super Expander 64 also lets you take full advantage of the eight function keys on the Commodore 64 through use of the KEY command. This command lets you program your own definitions for the various function keys (which are initially assigned useful commands such as CIRCLE, GRAPHIC, RUN, etc.). This can be accomplished in program mode or immediate mode and there is even an option to list all the current KEY assignments on the screen.

Programmable Characters

With *Super Expander 64* we also have the power to create our own characters. Let's say you're designing a bar chart or video game and the display requires custom characters that do not exist in *Super Expander's* character set. Simply lay out the characters on an eight-by-eight grid, compute the decimal values of each successive eight-bit row, and POKE those eight values into the area of memory containing the characters.

That area is laid out as follows: the first eight bytes (of memory location 50176) contain the definition for the letter "A", the second eight bytes contain the definition of the letter "B" and so on. Therefore if we alter the character definition of "A", everytime we use a CHAR command with "A" in it our new character will be displayed.

Programming With Super Expander 64

Now that we are familiar with Commodore 64 graphics and the differences in the two resolution modes, we can discuss how programs are written using *Super Expander 64*. As previously stated, *Super Expander 64* is an extension to the BASIC language on the Commodore 64. When the *Super Expander 64* cartridge is plugged in, you can use 21 new commands and 11 new functions in addition to the standard BASIC commands.

As with any other program on the Commodore 64, BASIC statements can be entered into the machine in program mode (with line numbers) or you can experiment with *Super Expander 64* by entering program statements in immediate mode (no line numbers preceding the command).

When you use *Super Expander 64*, programs are loaded and saved to disk or cassette normally. In fact, any straight Commodore 64 BASIC program can be loaded into the machine, "Super Expandified" and then saved back to disk or tape as an enhanced version of the original. Always make sure your *Super Expander* cartridge is plugged in, though. If you load in a *Super Expander* program without the cartridge, the normal Commodore 64 BASIC will not recognize the new BASIC "tokens" and errors will be generated.

Sound Effects and Music

Last but not least, is a feature that totally rounds out the entire package (is this Valley talk?): the ability to define and play your own sounds. *Super Expander 64* allows you to easily tap the resources of the powerful sound synthesizer (SID) chip in the Commodore 64.

The SID chip features three independent voices, nine octaves, four waveforms and programmable ADSR. (If you don't know what this means, don't worry about it; you will still be able to program music on your Commodore 64.) In order to use sound in your *Super Expander 64* programs, you must first "define" your sound or tune. This is accomplished by first setting the speed with a TEMPO command. The TUNE command is then used to define the waveforms and ADSR envelopes.

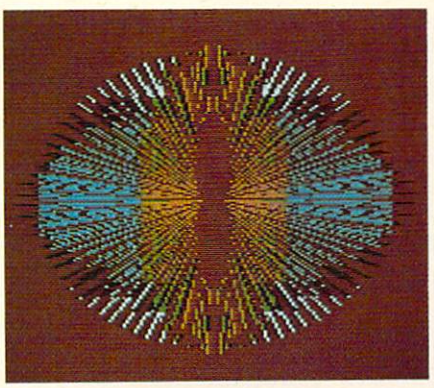
Super Expander 64 initially defines a list of ten "pre-sets" for you including a piano, accordian, guitar, xylophone and drum. You have the ability to alter these existing pre-sets or create your own unique sounds (and remember the Commodore 64 has direct output to a stereo, amplifier or 10,000-watt Marshall!!). The optional FILTER command can then be used to create resonance and to enhance or suppress selected frequency ranges in the sounds.

After you define the sound of the instrument or effect, the sound is played by entering a series of special characters that represent the value and duration of the music notes. These characters correspond to musical notation as closely as possible, i.e., C, C#, D, D#, E, etc. The character strings can be entered directly in the computer or they can be "played back" by executing print statements under program control. The user can copy sheet music into the machine or experiment by creating a unique sequence of notes. Using this easy method of composition you can create three-voice symphonies, each voice with a different sounding instrument.

Conclusion

This has been only a brief summary of the many powerful features of *Super Expander 64*. The documentation manual that comes with the product contains a thorough description of the *Super Expander 64* commands and parameters, complete with extensive program examples. With the *Super Expander* cartridge plugged into your Commodore 64, you will be able to experience computing power never before imagined: graphics, sound, color, animation... all at the tips of your fingers.

For me and the designers at Commodore, this completes the discussion of *Super Expander 64*. For you, the Commodore 64 owner, the fun is just beginning. Let's see what you can do with *Super Expander 64!!* C

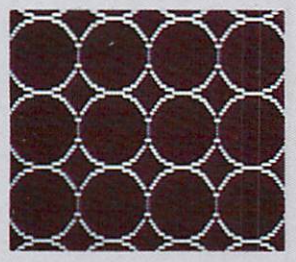


This picture demonstrates the ability to generate artistic screens using Super Expander's powerful graphic commands. This colorful montage was created using the COLOR and DRAW commands in High Resolution mode.

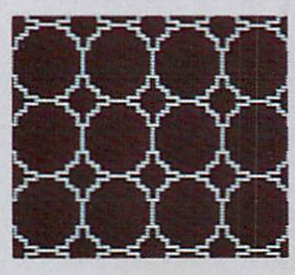
Picture credits: Waveform by Dr. Frank Covitz; Robot by Trip Denton; Montage by Dave Middleton; Business charts by Stephen Murri.

High Resolution Mode vs. Multi-Color Mode

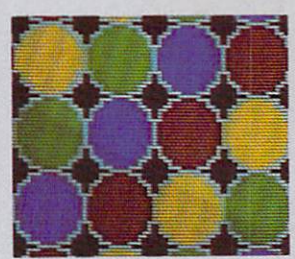
These screen enlargements demonstrate the differences between high-resolution mode and multi-color mode on the Commodore 64.



This picture is an enlargement of a design created with *Super Expander 64* in high resolution mode (320 × 200 pixels). This mode is used when maximum fineness and detail are required. Color is handled identically to standard text mode in that each eight-by-eight character cell can contain only two colors: foreground and background.



In this enlargement the same design was re-created using multi-color mode. In multi-color mode, screen resolution is cut in half to 160 × 200 pixels. This tends to exaggerate the distortion effect known as the "jaggies".



This frame demonstrates that in multi-color mode, the horizontal resolution is sacrificed for increased color capability. In multi-color mode, each eight-by-eight character cell can contain four colors. This mode is used when the emphasis of the display is on the mixing of various colors.

This new software package for the Commodore 64 puts 114—yes, you saw it correctly—114—additional commands at your disposal for creating programs, without interfering with the 64's resident BASIC. Both beginners and advanced programmers will love the convenience these added commands provide.

Simons' BASIC

By Jim Gracely

I heard a lot about *Simons' BASIC* for the Commodore 64 before I got my hands anywhere near it. One person told me that it was a great BASIC aid program with autonumbering, a trace command and a variable dump. Another person was overjoyed with the PRINT AT and PRINT USING commands. A third person flipped over the screen manipulation commands such as FLASH (a color) and INVERSE (the whole screen). What none of these people told me was that *Simons' BASIC* is still much more!!

The three areas mentioned above are just three out of 12 areas of BASIC enhancement that this program provides. In fact, *Simons' BASIC* adds 114 *additional* commands to the Commodore 64. This seemed to be a good number of commands, but as I worked with *Simons' BASIC* I realized just how many commands this is! The Commodore 64 has a basic set of 72 commands. *Simons' BASIC* adds over one and a half times more commands. The result of so many new commands was that just as soon as I tired of playing with one set of commands, I found a new set to play with. For weeks I was playing with new commands, just trying them out directly and in little programs.

The Materials

Simons' BASIC comes with a disk (backup protected) and a 120+ page manual. To my relief the manual contains descriptions of commands and not instructions. The program is very easy to load and once loaded it is in place until the 64 is turned off.

Simons' BASIC is an 8K machine language program that resides from 32768 to 40960. It is loaded into the 64 through a boot program on the disk. The load takes about two minutes and a changing screen color indicates that loading is occurring.

This program is easy to use from the start because it doesn't affect the standard BASIC commands in any way. The beginning programmer can write programs in normal BASIC and just use the programming aids. As your knowledge of BASIC increases your appreciation and use of *Simons' BASIC* will also increase.

Illustration—Jake Smith IV


on error
DUMP
INSERT
DISK
DIV
MOD

The Commands

The manual divides the 114 new commands into 11 groups (and chapters). This division seems logical to me and I will use it to provide a brief overview of the commands. This section is going to provide an overview only and does not include all of the commands or possible uses of the commands.

Programming aids: These commands aid the programmer in entering and debugging programs. The common commands such as AUTO (line numbering), RENUMBER, TRACE and OLD (reverse a new command) are all here. In addition, there are a number of uncommon commands such as DUMP (all non-array variables) and RESET (move data pointer to any selected DATA line). A KEY command allows permanent assignment of 16 function keys (that's right—16). There is also a set of two commands to make program lines disappear (for security of course). In addition to the commands mentioned, there are another ten commands in this group.

Input Validation and Text Handling: The commands in this group are used to simplify the use of strings and input statements. There are commands here for inserting one string into another (INSERT), writing one string on top of another (INST) and finding the offset of one string within another (PLACE). There is a useful command called DUP for creating a string of repeated characters. For example, to create a string SP$, which consists of 40 spaces, you can use DUP (" ",40). This group also contains the famous AT (print at) and USE (print using) commands along with a CENTRE—look-at-the-funny-spelling—command. Three new input commands allow you to: limit the type and number of characters input (FETCH), check for function keys (INKEY) and check for a key or one of a set of keys (ON KEY).

Extra Numeric Aids: Here are a number of commands that make number handling and conversions very easy. MOD, DIV and FRAC are commands used to find the remainder of a division, the whole number part of a division, and the fractional part of a number. There are commands for converting a binary number to decimal (%) and a hexadecimal number to decimal ($).

Diskette Commands: A single command (DISK) replaces OPEN15,8,15:PRINT#15 and the CLOSE15. Another single command (DIR) allows viewing of a part or all of the disk directory without destroying what is in memory (just like the directory command on the CBM machines).

Graphics: There are 20 com-

A Couple of Examples

I have selected three commands to discuss in more detail. I chose these commands because they are the ones that I tend to use the most. The three commands are KEY, DUMP and ON ERROR.

KEY

This commands allows you to assign any number or string of up to 15 characters to each of 16 function keys. The 16 function keys are:

Four function keys (1-4)
Four shifted function keys (5-8)
Four Commodore logo key function keys (9-12)
Four shifted + Commodore logo key function keys (13-16)

Fifteen characters are enough for a wide range of applications. I keep one defined as LIST, one as RUN and one as DIR"$". I put these assignments in a small program and run it each time I load *Simons' BASIC*. Single-keystroke listing and running of programs becomes very habit forming and I miss this command when I'm not running *Simons' BASIC*.

DUMP

This command will print out all variables and their present value. It can be used in both program mode and direct mode. It is an invaluable aid in debugging programs. At any point in a program the DUMP command can be added and at that point in program execution, all variables will be dumped to the screen.

Another nice application of the DUMP command is for program documentation. Once a large program is written and run, use OPEN4,4:CMD4 to specify the printer as the output device. A DUMP will then print the variable list to the printer. This printout can be saved as a list of variables used in the program.

ON ERROR

The ON ERROR command is a method of error trapping. This is a command that can save a lot of headaches by controlling the effect of an error on the program. The ON ERROR command is used at the beginning of a program (or the beginning of a section requiring error trapping) with any legal BASIC commands following it. The easiest way to use it is probably with a line such as ON ERROR: GOTO 1000. This command would send the program to line 1000 anytime an error occurred. The errors are then assigned a number from 1-23 and the numbers assigned to the variable ERRN. In addition, the line on which the error occurred is assigned to the variable ERRLN. Once an error is trapped a message can be printed and the program can restart or continue execution.

mands in this group including commands for going into high-resolution (HIRES) and multi-color mode (MULTI). There are single commands for turning on pixels (PLOT), drawing rectangles (REC), lines (LINE), circles (CIRCLE), arcs (ARC), angles (ANGL) and solid blocks (BLOCK). There are also special commands for creating designs (DRAW), printing text on high resolution screens (CHAR and TEXT) and filling in areas with colors (FILL).

Screen Manipulation: The high points in this group are the single commands for printing out a normal or high-resolution screen (HRDCPY and COPY) and saving and recalling a normal screen (SCRSV and SCRLD). There are also commands here for flashing the color of the screen (FLASH) and border (BFLASH), inverse any part of the screen (INV) and moving pieces of the screen around (MOVE). Four of the most fun commands in *Simons' BASIC* are for scrolling the screen up, down, left and right (UP, DOWN, LEFT, RIGHT) with your choice of wrapping around (W) or blanking (B).

Sprites and User-defined Characters: This group contains all the commands you need to define your own sprites (DESIGN, @, CMOB and MOB SET). It also contains all the commands to move the defined sprites around the screen (MMOB, RLOCMOB) and to check for collisions with other sprites and screen data (DETECT, CHECK). Similar commands are also here for user-defined characters (MEM, DESIGN and @).

Structured Programming: Here are all of the commands found in languages such as PASCAL, FORTRAN and COMAL. These are the commands that change Commodore's BASIC into a structured language. There are IF.. THEN.. ELSE and REPEAT.. UNTIL commands, procedure definitions and LOCAL and GLOBAL variables.

Error Trapping: This group contains the commands and variables for trapping just about all of the errors which may occur during a program.

Music: Parameters for making music are set with three commands: one each for the volume (VOL), the waveform (WAVE) and the ADSR envelope (ENVELOPE). The notes are defined as a string (MUSIC) and can be played (PLAY) either before or while continuing with the program.

Read Functions: The values for a light pen (PENX, PENY), a joystick (JOY) or a paddle (POT) can be read with a simple variable assignment. This means that a statement such as A=JOY will put the value of the joystick into variable A. Commands like these are a joy to the game writer!

Pros and Con

It appears that *Simons' BASIC* was well conceived and that most sources of problems eliminated. I have found very few programs that will not run when *Simons' BASIC* is in place. This is mildly surprising because many of the commands are interrupt-driven. The DOS program for the 64 will still run normally (which should make many people happy) but many of the commands are obsolete because of the features of *Simons' BASIC*.

The only area where I was a little disappointed was in music. There are some great features here such as a PLAY2 command, which allows the music to be interrupt-driven and therefore play as the program continues. However, the notes for any given piece of music must be entered as one long string variable. This limits the number of notes and makes entering notes difficult. Also, only one voice can be played at any time.

There is one last highlight I should be sure to mention and that is the manual itself. The manual does not contain any information on BASIC or how to program but instead concentrates on *Simons' BASIC*. There are 120+ pages of information here presenting each command in full detail. Each command is described individually with the format and purpose of the command presented first. In addition, each description is accompanied by at least one example and provides the expected result when the example is executed. Describing the result of each example is a great help because it insures you that you have correctly entered the example.

The manual also includes an index that is a real index! All the commands are indexed along with some cross references. The index also contains an entry for the full name of most of the commands ("HIRES" and "high-resolution graphics" are both in the index, for example).

One more nice thing about both *Simons' BASIC* and the manual is the addition and description of ten new error messages. It even has a new error message for too many nested loops or procedures (?STACK TOO LARGE). This same problem on the standard 64 would generate an ?OUT OF MEMORY error message.

Conclusion

Anyone who enjoys programming will enjoy this program. How could you *not* like it with all of the time-saving features? The fact that all normal BASIC programs will run is a big plus and will allow you to slowly ease into the fancier commands and options.

When you get *Simons' BASIC* be sure to read "About The Author" on page ii of the manual. You may be surprised to find out just who David Simons is!

My final words—Go out and Simonize your 64!! **C**

MAGIC DESK

A New Direction for Home Computer Software

By Diane LeBold

Make yourself comfortable in Commodore's animated office —where you can do the computerized office work of the future.

Illustration—Robert Neumann

Imagine having a full-color, animated office appear on your computer screen. You're at a large desk, on top of which are all the usual office accoutrements—typewriter, phone, card index, financial journal and calculator. Beside the desk is a three-drawer filing cabinet with a digital clock on top of it and under the desk is a small waste basket. On the other side of the "room" is a door. Just above the desk hovers (and this is the only odd thing about the scene) a disembodied hand with a pointing finger, dressed tastefully in blue serge.

The hovering hand, controlled by your joystick, is your willing slave. Time to type a letter to Aunt Flo? Move the dextrous digit down to point to the typewriter and press the fire button. *Voila!* Some fancy prestadigitation (that's short for magic) that conjures up a realistic-looking typewriter, paper already in place, margins set and ready for you to begin typing. Go ahead.

Clickety click. Sounds like a real typewriter, doesn't it? Look at the screen scrolling horizontally. Listen to the polite little bell that lets you know you're nearing the end of the (80-character!) line. Then press RETURN and, just as on a real typewriter (as opposed to a computer), the carriage returns to the beginning of the next line (with appropriate sound effects, of course). Press RETURN and go down as many lines as you wish. Scroll the "paper" up or down using your joystick. To reset margins simply point your finger to the "margin reset" picture at the bottom of the screen.

Is this work? It feels like play. Any typist with a fear of those great and awesome COMPUTERIZED WORD PROCESSORS will hardly notice that that's just what this little typewriter is. It doesn't have all the capabilities

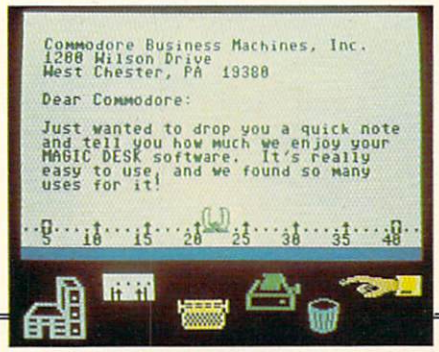
Commodore's Magic Desk *creates an office on your computer screen.*

of the sophisticated, full-featured word processors but it will take care of your home needs very tidily.

OK, so you've typed your letter. Now you need to get it from the screen into Aunt Flo's mailbox. Very simple. Point your trusty finger to the picture of the printer at the bottom of the screen and hit the fire button. What do you know, a printed copy!

Want to file the letter for future reference? Press RESTORE to get back to the desk. Look, there's a little sheet of paper that wasn't there before. That's your letter. If you move the finger to point to the filing cabinet, guess what will happen. Right, you'll get access to the disk drive so you can "file" your letter. You have a choice of three file drawers, each with ten files. Each file can have up to ten pages. You can move the pages from one file to another or copy the pages into several files if you wish.

On the other hand (so to speak) if you decide you don't really need to keep this particular correspondence, point to the waste basket



The Magic Desk *typewriter is a friendly little word processor.*

instead and... you guessed it. The best part of throwing a letter away is the sound it makes as it descends to the basket.

Meanwhile, the digital clock atop the filing cabinet has been faithfully keeping track of time. By the way, did you remember to set it? If not just get your frivolous finger up there and point to it, then enter the correct time. Press the fire button again and the clock will continue its vigil. That way no matter how involved you get playing with your *Magic Desk* you might still be able to get the chicken in the oven in time for dinner.

Please notice that everything we've talked about is done with pictures. That's because there are no language instructions in the *Magic Desk*. Rather, pictorial "computer metaphors" control all aspects of the program, from selection of features to individual menu items, which, in Commodore's estimation, makes it a truly multinational package.

This first package in Commodore's *Magic Desk* series contains only word processing and information filing but more functions are on the way, as you might suspect from the other items present on the desk. The next *Magic Desk* cartridge should contain calculating and home budget features. After that you can probably expect to see an artist's (graphics) package, accessed through an easel that would be placed in your "office", and perhaps some educational programs and some music programs —all of which would be represented by different items on the desk.

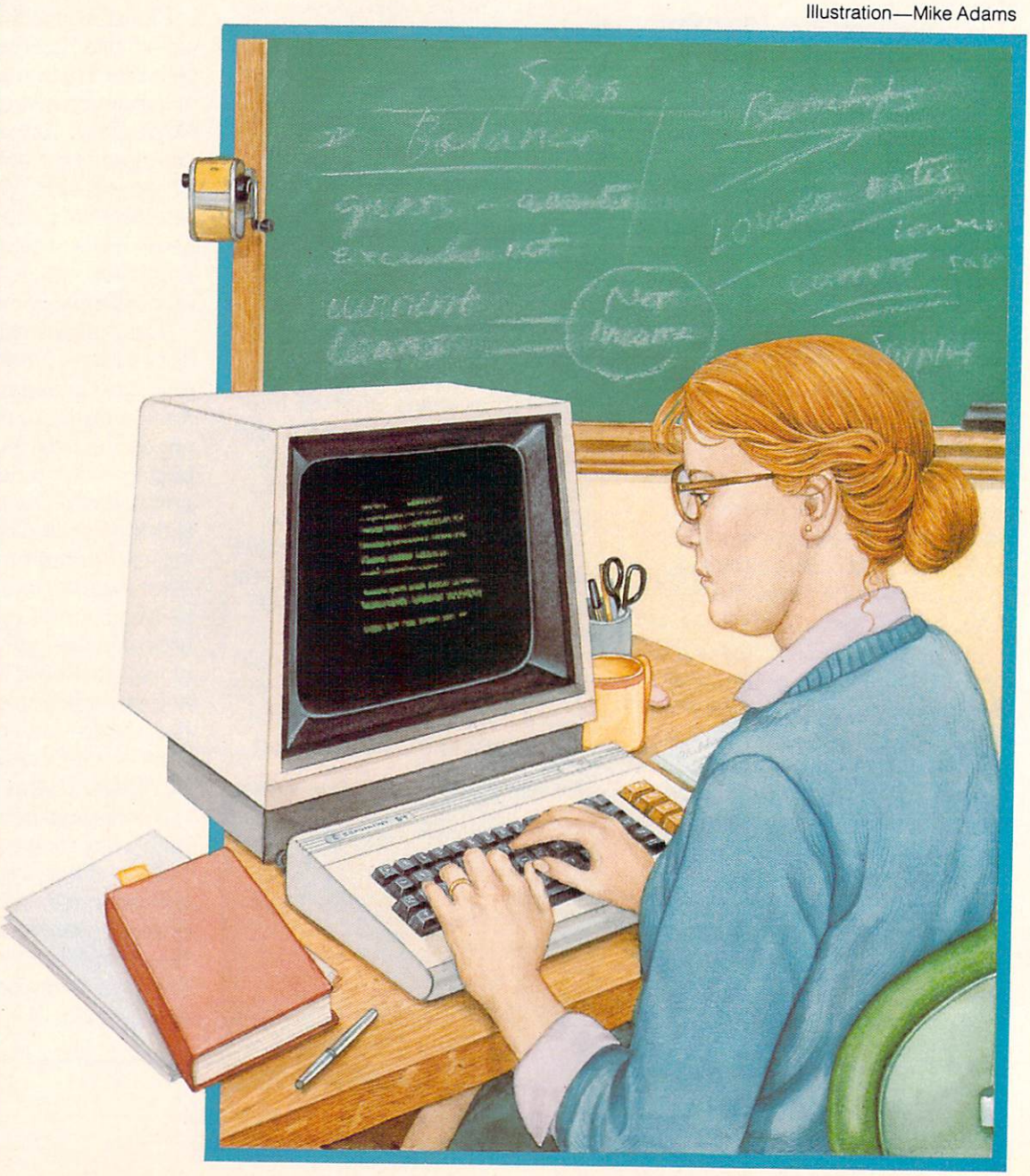
You should be seeing copies of this first *Magic Desk* sometime before winter. It's a unique approach that takes home applications software in an entirely new direction—blurring the lines a little between work and play, fact and fantasy. **C**

Commodore's

Educational Software Packages

Make Teaching—and Learning—Fun

Illustration—Mike Adams



Teachers can use Commodore's EasyLesson 64/EasyQuiz 64 *to take the drudgery out of creating and grading tests and the* Visible Solar System *cartridge to turn students on to the wonders of our planetary neighborhood.*

By Gail Austin

Teacher, Take it "EASY"

Oh, how easy life would be if you as a teacher had an identical twin, extra time or another pair of hands. Although Commodore could not clone you, create more hours in a day or give you more hands, they did do something to take some of the drudgery out of teaching. They created a software package for the Commodore 64 called *EasyLesson 64/EasyQuiz 64* to make writing and selecting test questions more fun. It even provides answer keys for printed tests and grades computer-generated ones.

Now, to be honest, no lesson preparation is totally easy. After all, you have had extensive training and experience in your subject area, in how to formulate good questions and in how to anticipate what students will answer. With these abilities already under your hat and an interest in computerizing some of your teaching tasks you can purchase this inexpensive program and feel relatively confident that you have entered the "computer age."

Let's take it nice and easy. Do not be dismayed to find that the manuals are not set up like a teacher's guide. After all, not everybody can be a teacher—not even computer experts who write manuals as though you have been working with computers for years. The first mini-manual congratulates you on your purchase and begins to explain the Directory Assistance and DOS Wedge programs. These will be helpful, but just glance over it for now. Then type: LOAD":*",8

RUN

After looking over the screen, type the number "one" and press RETURN.

You see that you skipped right to the next mini-manual for *EasyLesson 64*. There is a lot of good information in the first one and it is very valuable, but you do not want to get bogged down in advanced operations before you are ready. You might think that you should have spent your money on a new pair of shoes.

Forget the shoes. Just take the next step and read the preface to *EasyLesson 64*. It is important for you to know that you will:

1. need to think of categories for your multiple-choice questions. Seven categories is the maximum allowed. The computer uses these categories to generate different tests.
2. be able to type as many as five lines for each question.
3. want to have your textbook available for finding reference pages.
4. always type the correct answer next to answer "a". The computer will later place answers randomly.
5. need to type other good but incorrect answers.

This manual is fairly understandable. Only a few things need further explanation so you don't reconsider those shoes:

1. In Section 1.2, Create a Question, after the question and answers are typed, the question

   Set/Rest which?

   appears and you type the number(s) of each category the question relates to. After you press RETURN, each number is highlighted. To erase one just type it again.

2. When in 1.5, Generate an EasyQuiz, remember only five characters can be entered in a title. Make them easy to remember and keep a list of them since they are not listed on the Directory Assistance program. You can list them after you quit the program by typing:

   LOAD"$",8
   LIST

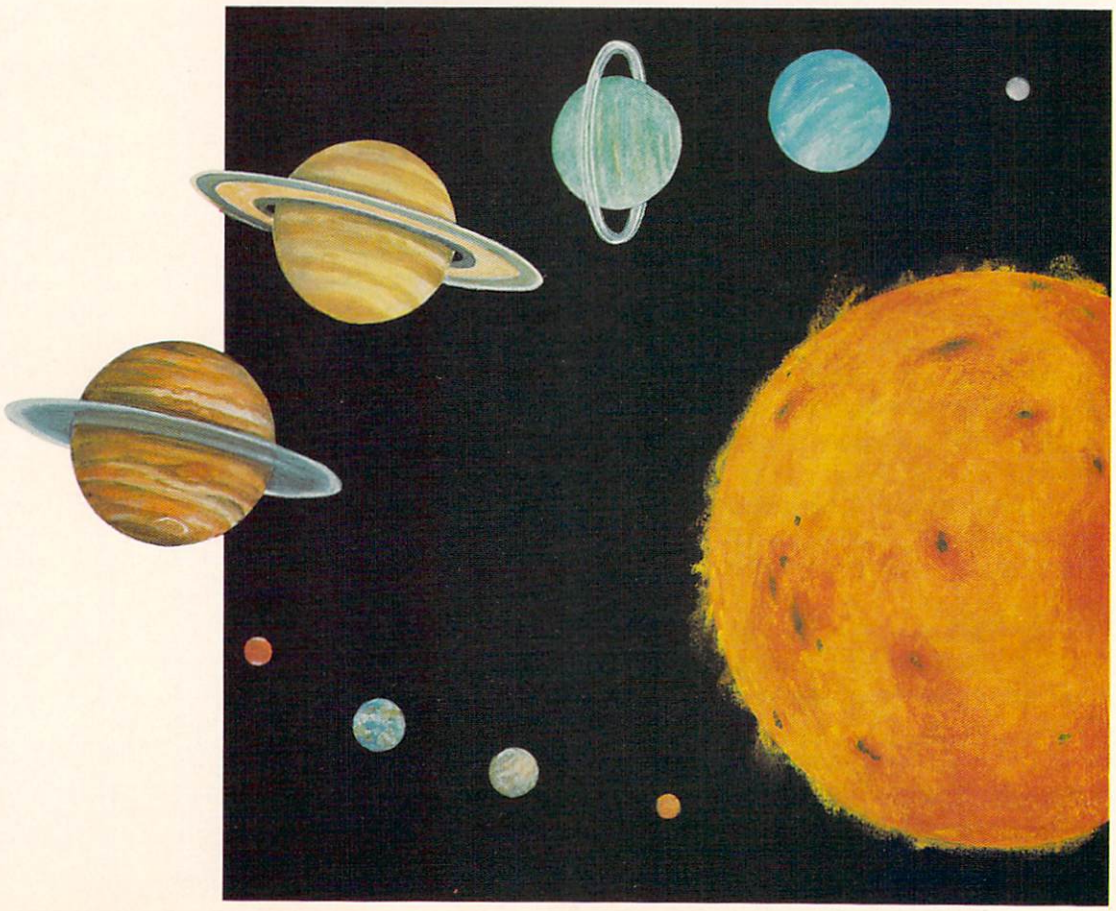
   or by using the DOS Wedge.

3. Remember that you save these quizzes on the disk. They can be used again at a later time.

It may seem as though this program has some faults—but what program doesn't? Sure, graphics would be nice to have. But that's a step above. You will have to find out more about Commodore's PILOT to have that—plus more.

In the meantime you will have fun with *EasyLesson 64/EasyQuiz 64* as it helps you:

- think through your teaching goals and make them into categories.
- create individual or group quizzes and answers.
- offer helpful hints by providing references.
- make both computer-generated or hard-copy tests easy to grade.
- enjoy your computer while doing your work—nice and EASY!



The teacher can use the* Visible Solar System *as a launch pad for class discussions, further research, films and projects about the solar system.

The Visible Solar System and The Visible Teacher

Commodore has produced a beautiful graphics program for the Commodore 64 to help students begin to understand our solar system. Called the *Visible Solar System*, this program was designed with a visible teacher in mind. The main goals of this software are to:

- create an interest in our closest planetary neighbors
- compare those planets in terms of distance, size, revolution and rotation times, number of moons, etc.
- help students understand spatial relationships.

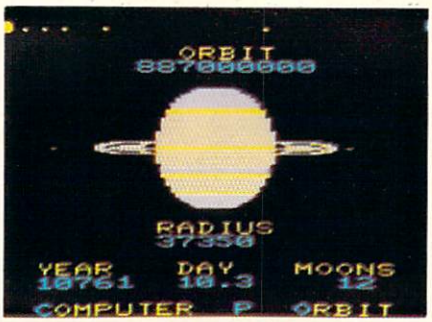
This program is best presented in a group where the teacher can explain what keys to use and what the displays show. The class can look at the solar system through the spaceship's camera. The group has control over the position and height of the spaceship as well as the camera angle.

The manual is helpful; however, there are some spaceship controls on the second page that need revision. The "up arrow" key moves the red target forward, the comma key moves the camera ahead and the period key moves the camera down. It is important to remember that the camera is always looking out a window that is located at the top of the spaceship.

The first section of the manual, Navigation, needs more explanation. Here is how it could be done by a class:

1. Decide where the spaceship should be moved. A red target will be moved to that position before the spaceship is sent:
   CRSR → moves target right
   CRSR ↓ moves target back
   ← moves target left
   ↑ moves target forward
2. Press "G", which makes a sound and shows that the spaceship moved.
3. Decide if the spaceship is at the altitude desired. Press "U" to move the spaceship to a higher altitude and "D" to move to a lower altitude. Notice the bar graph at the upper left of the screen.
4. Decide if the camera is at the desired angle. The blue target shows where the camera is pointing. Press the comma key to make a larger angle and the period key to make a smaller angle. Notice the bar graph at the lower left of the screen.
5. Press "V" to view the solar system. If no planets are within your view, the screen appears blank.
6. Press "O" to change the position of the spaceship and camera. See the third page of the manual for more views.

Since the entire solar system is not shown, this is where the visible teacher is needed. She or he can use the manual to get more information on how to access the close-ups of some of the planets and their comparisons. The teacher can use the *Visible Solar System* as a launch pad for class discussions, further research, films and projects about the solar system. **C**



Commodore's *Visible Solar* System

Easy Finance 64

A Finance Package for Everyone

Five new financial packages for the Commodore 64 help you plan your budget, decide on investments and generally get the most out of your money.

By
Ginger Bardi
and
Lynn Kachelries

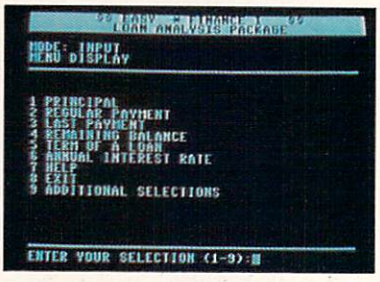
In today's fast-paced economy, you need to know how to get the most out of your hard-earned money. Your Commodore 64 can help you learn how. Commodore and Eagle Software of Wayne, Pennsylvania, have put together five exciting finance packages for the Commodore 64. The *Easy Finance* series shows you how to make the right financial decisions and plan for future expenditures.

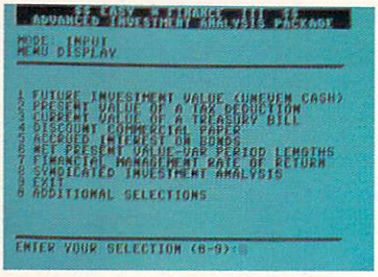
Easy Finance is completely pre-programmed; you need absolutely no programming knowledge to use this product. All the screens use several colors, which you can select so the information is very clear. Each screen is divided into four sections: the title area, where the title of the calculation is shown; the display area, where the input values are shown after they have been entered; the results/error message area; and the user input region, where the next value to enter is shown. To make the program even easier, *Easy Finance* has a tutorial you can request to see on your screen any time during the program.

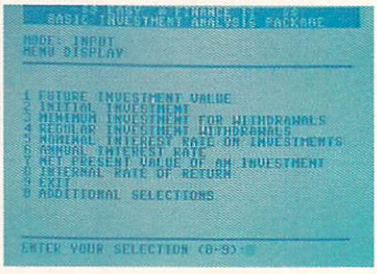
The five *Easy Finance* packages provide over 70 useful calculations covering loans, basic investments, advanced investments, business management and statistics. The programs prompt you for the necessary information. The manuals offer an example of every calculation to show you a situation where it would be useful. The results of the calculations are dis-

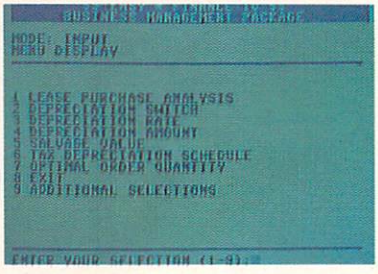
played on the screen or can also be printed. When there is more than one result—for example, an amortization schedule—the results are formatted in a table for easy reference.

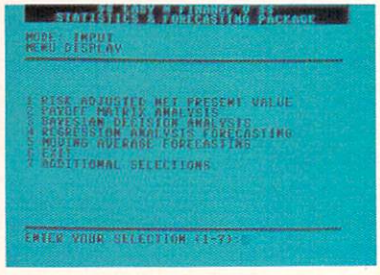
Figuring out the mortgage term that would best fit your budget or how much your savings account will be worth in a few years are just two of the calculations where *Easy Finance* can help you. To give you an idea of how easy the programs are to use, we will go through two sample problems. Once you know how to do one *Easy Finance* calculation you can do them all. You are prompted for the information in the same way for all the calculations.











Mortgage Comparison Analysis

Use this *Easy Finance I* calculation when you want to compare financial alternatives to decide how to set up a budget, choose a mortgage or select a certain type of loan. You can vary the amount borrowed, the interest rate or the term of the loan to determine which financial alternative will be best for you. You will first be prompted for the item you wish to vary. The number you enter here corresponds to the prompt number: principal=2, annual interest rate=3 and number of years=4.

Suppose you are considering a mortgage of $40,000.00 and you want to know how the monthly payments would differ if you borrowed the money over 15, 20, 25 or 30 years at an interest rate of 11.75%.

Prompts	**Information you must enter**
1. Variable item #	4 (No. of years is the 4th prompt)
2. Principal Amt.	40000
3. Annual Interest Rate	11.75
4. No. of years	15
5. Interval Value	5 (no. of years between each sample loan period)
6. Maximum Value	30

Results:

# YRS	PRINCIPAL	INT RATE	MO. PAYMENT	TOTAL INTEREST
15.00	40000.00	11.75	473.65	45257.46
20.00	40000.00	11.75	433.48	64035.88
25.00	40000.00	11.75	413.92	84175.78
30.00	40000.00	11.75	403.76	105355.00

By looking at the difference in your monthly payment between 15 and 30 years, you can decide which would be the optimum time period to match your budget. Since the difference in the monthly payment is only $70.00, you may decide to pay your mortgage off in 15 years and save $60,000 in interest.

Regular Deposits

Use Regular Deposits *(Easy Finance II)* when you want to reach a certain savings or investment account value by a specific date. This will tell you how much your periodic deposit must be to reach your goal.

Suppose that in three years you will need $5,000.00 for your initial college expenses. What monthly deposit must you make to attain $5,000.00 if the interest is 8.5%?

Prompt	Information you must enter
1. Final Total Value	5000
2. No. of years	3
3. No. of months	0
4. Interest Rate	8.5
5. No. Deposits/Year	12 (one each month)

Results:
Regular Deposit Amount = $122.42

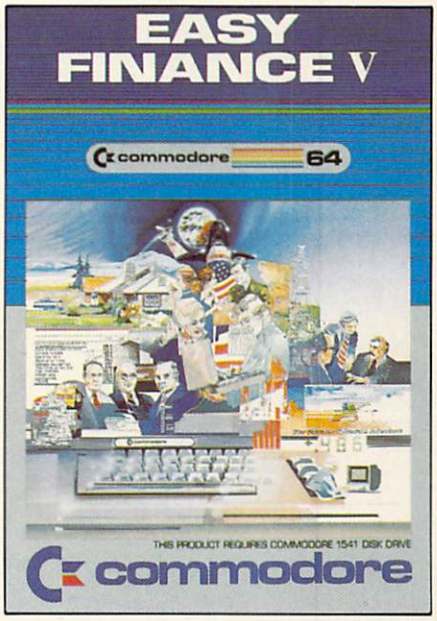
So, if you deposit $122.42 each month for the next three years you will have the $5,000 needed for college.

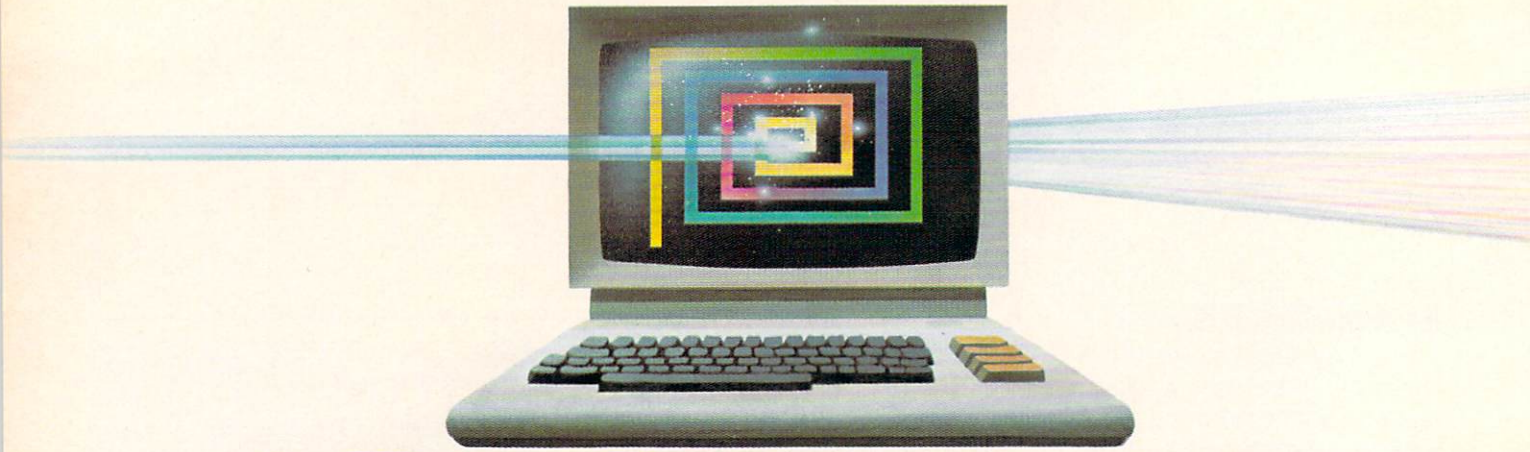
There is an *Easy Finance* package for everyone. *Easy Finance I and II,* which cover loans and basic investments, are applicable to almost everyone. You will find many uses both at home and in business to determine such things as the remaining balance on a loan, the monthly payment for a car loan or the future value of an investment. The business management calculations in *Easy Finance IV* provide fast reliable answers to determine, for example, whether to lease or purchase an asset or to find out when a depreciation switch from declining balance to straight line would allow larger depreciation amounts in the later years of an asset's lifespan.

For business investments use *Easy Finance III.* This advanced investment package will determine the current value of a treasury bill, the financial management rate of return or the maximum price for an acquisition. For those of you interested in statistics, *Easy Finance V* will do Bayesian decision analysis or can help you with regression analysis forecasting. Because of the clear, step-by-step manner in which the calculations are entered, even the most complicated problems are easy to understand.

These five packages are designed to be used as tools to help you clearly lay out what the alternatives are in your money decisions. With the facts at hand you can be assured of making a wise decision when the time is right.

When deciding which *Easy Finance* package to purchase, look on the back of the box. Every *Easy Finance* box has a list of the calculations that it performs, so you can tell immediately which one you need. We think you'll find that you want them all!! C





VICTERM 40

By Barbara Karpinski

VICTERM 40 software, for use with your VIC 20 and VICMODEM, gives you a 40-column screen, the ability to upload and download and much more.

VICTERM 40 is an easy-to-use, powerful communications package for the Commodore VIC 20. This package, together with a modem, allows you to turn your VIC into a smart terminal that can transport you to a new and exciting world—the world of telecommunications. In this world you can visit the library, make travel arrangements, receive the latest news, weather and sports, and access a large variety of services such as CompuServe and Dow Jones News/Retrieval. You can also "talk" to another computer far away or to your mainframe at school. You can even receive online technical assistance using the Commodore Information Network on CompuServe. All this and more can be accomplished from the comfort of your own home. All you need is your VIC 20, a VICMODEM or AUTOMODEM, a modular telephone and VICTERM 40.

Many of you who have waited patiently, and sometimes not so patiently, for this package to be released will not be disappointed. VICTERM 40 incorporates a lot of sophisticated features that allow the VIC to be used for serious applications as well as for games and recreation. Some examples include:

—A choice of a 40-column by 20-line text screen, a 22-column formatted screen, or a normal 22-column screen.

—Error-free upload/download capability.

—Color graphics while in 22-column mode.

—Selectable intelligent terminal functions such as cursor positioning, graphics and flow control.

—Choice of two methods of file transfer: using the RAM capture buffer or a direct error-free transmission while on CompuServe.

Of course the most notable feature of VICTERM 40 is the ability to expand the VIC screen to 40 columns instead of the usual 22 columns we are used to seeing. Since most information services usually output at 32 columns per line, this feature makes reading the screen much easier.

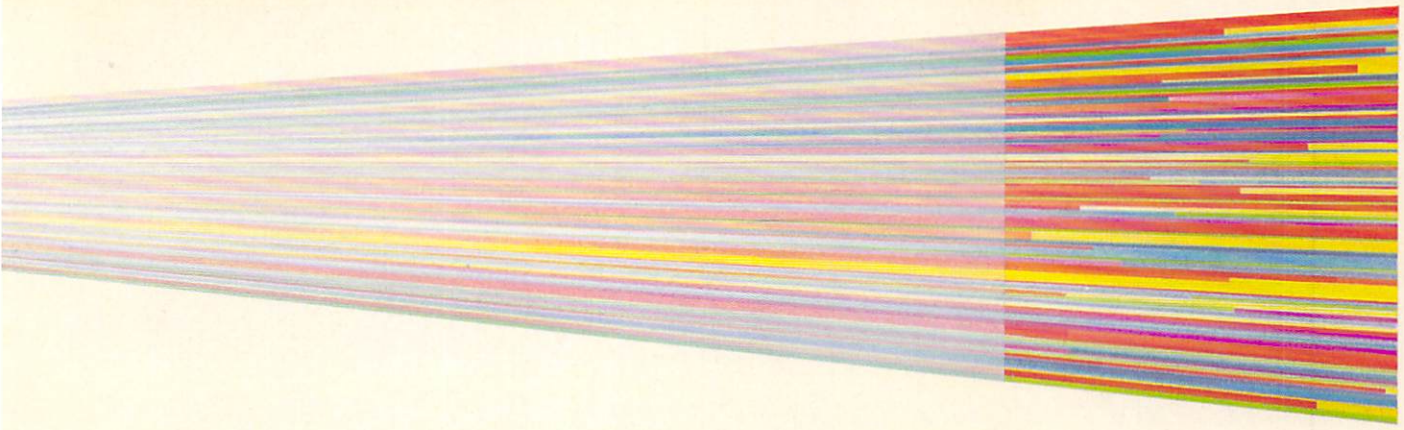
Most of the functions of VICTERM 40 are controlled by three menus and eight function keys. The two communication menus allow you to set control features such as selectable baud rate, parity, word length, etc. This gives you the flexibility to work with different computers. To switch between the two communications menus, type N (for next). Menu options are automatically preset to CompuServe standards but can be easily changed to be compatible with the system you are communicating with.

A special feature of VICTERM 40 is a separate options menu for file transfer and automatic dialing operations if you are using a modem with automatic dialing capabilities. This menu can be reached from any communications menu by typing F (for files). The first five features of the options menu control the RAM capture buffer where data you want to transmit, receive or hold for later display is temporarily stored. A small inconvenience in this program is that the RAM capture buffer is about 3K in 40-column mode and about 6K in 22-column mode, but if you use a memory expansion box to add additional RAM, VICTERM 40 will automatically use this space for the RAM buffer.

Eight function keys assure ease of travel between menus and terminal mode as well as define the following functions:

F1—By hitting the F1 key in terminal mode, you can open and close the RAM capture buffer.

F2—Transmits buffer contents without making changes or cor-



Illustration—Greg Purdon

Increasing Your VIC's Telecommunications Capabilities

rections. You can slow transmission by holding down the space bar.

F3—Changes background and border colors. Hold the key down until you find a combination you like.

F4—Displays a communications menu from which you can access the options menu.

F5—Changes the character color. Hold the key down until you find a color you like.

F6—Changes user color in two-color mode.

F7—Sends a control S (stop) to the other computer. (You can use this option when transmitting buffer contents.)

F8—Sends a control Q (resume) to the other computer. (You can use this option when transmitting buffer contents.)

Besides these keys, many more special function keys like backspace and control C are available. You can find a more detailed description of each function in the VICTERM 40 user's guide which, incidentally, has been designed to be easy to read and helpful.

Another helpful function of VICTERM 40 is the auto-dial option. Hooking up can be accomplished in one of two ways depending on the modem you are using. If you are using the VICMODEM you must manually dial the local CompuServe access number but if you are using the AUTOMODEM you can have the computer do this mindless task for you. First go into the options menu and choose option number eight. Type in the phone number. As each digit is dialed it is printed on the screen. When a connection is made VICTERM 40 automatically goes to terminal mode; otherwise it displays a NO CONNECTION message.

Perhaps the most exciting feature of VICTERM 40 is the uploading/downloading capability using CompuServe's error-free B protocol transmission. This allows you to save programs in three different formats: text, binary, and machine-specific or image form. A three-letter file extension determines the type of transfer that will occur. The valid extensions are:

1. .txt—A text transfer is used to transfer ASCII text files like untokenized BASIC programs and text files produced by word processing. When you omit an extension, this file type is assumed.
2. .bin—This transfer is used to transfer eight-bit files such as tokenized BASIC programs and machine language programs. No data is altered during a binary transfer.
3. .img—An image transfer is used to transfer machine-specific files. VICTERM 40 inserts all information necessary to recreate the file exactly as it originally existed. If you download an .img file uploaded by a non-Commodore computer, VICTERM 40 will issue an incompatibility warning. A great advantage of this form of file transfer is that when you download an image file to your computer as a VICTERM 40 image file, you don't need to re-type the entire program to have it run; just load the program in and RUN it. (Think of the hours of wasteful typing this saves!)

When on CompuServe you also have a choice of receiving or transmitting files via the RAM buffer or directly to or from disk. You can also transmit to tape, but you must use the RAM buffer in that case. Another advantage of VICTERM 40 worth mentioning is the two-color option in half duplex mode. This makes the text you type in one color and the text another computer sends a different color thus making it very easy to distinguish between who types what, and also easy on the eyes.

I have not by any means explained ALL the features of VICTERM 40, but I have highlighted enough of them to give you a small idea of the power of VICTERM 40 and its flexibility to support the needs of both the experienced user as well as the beginner. **C**

COMMODORE MARKETED SOFTWARE

an Overview of Products Available from Commodore's Software Division

These products are either already available or will be shortly. You can find them at your local Commodore dealer.

Order Number	Product Name
Telecommunications	
C1600	VICMODEM
C1605	Phone Adaptor
C1650	AUTOMODEM
VIC1610	VICTERM 40
Applications Software for PET/CBM	
400020	Assembler Development—8050
400021	Assembler Development—4040
400030	Integer Basic Compiler—8050
400031	Integer Basic Compiler—4040
400040	CMAR Record Handler
400050	UCSD Pascal (Without Board)—8050
400051	UCSD Pascal (Without Board)—4040
400060	PETSpeed Basic Compiler—8050
400061	PETSpeed Basic Compiler—4040
500010	OZZ—8050
500011	OZZ—4040
500039	Dow Jones Portfolio Management System (80 Col. RS232)
500041	BPI Accounts Receivable
500049	BPI Accounts Payable
500043	BPI General Ledger—8050
500045	BPI General Ledger—4040
500046	BPI Job Cost
500047	BPI Inventory
500048	BPI Payroll
500050	Legal Time Accounting
500051	Medical Accounting System
500052	Atlas 1200 Service and Maintenance
900040	I.R.M.A. II (Information Retrieval & Management Aid)
VIC 20 Software	
VIC Business Series	
VIC-2001	Simplicalc (Disk)
VIC-2002	VIC File (Disk)
VIC-2003	VIC Writer (Disk)
VIC-2004	Money Decisions I (Tape)
VIC-2005	Money Decisions II (Tape)

Order Number	Product Name
VIC Home Information Series on Disk	
VIC-3001	Quizmaster
VIC-3002	Know Your Child's IQ
VIC-3003	Know Your Own IQ
VIC-3004	Know Your Personality
VIC-3005	Menu Planner
VIC20 Recreation Games on Cartridge	
Video Arcade Series	
VIC1901	VIC Avenger
VIC1904	SuperSlot
VIC1906	Super Alien
VIC1907	Jupiter Lander
VIC1908	Draw Poker
VIC1909	Road Race/Midnight Drive
VIC1910	Radar Rat Race
VIC1913	Raid on Fort Knox
VIC1919	Sargon II Chess
VIC1920	Pinball Spectacular
VIC1921	Super Smash
VIC1922	Cosmic Cruncher
VIC1923	Gorf*
VIC1924	Omega Race*
VIC1925	Money Wars
VIC1931	Clowns*
VIC1937	Sea Wolf*
VIC1938	Tooth Invaders
VIC1939	Star Post
*Bally Midway Games developed under Commodore's licensing agreement with Bally Manufacturing Company.	
Scott Adams Adventure Games	
VIC1914	Adventure Land Adventure
VIC1915	Pirate Cove
VIC1916	Mission Impossible Adventure
VIC1917	The Count
VIC1918	VooDoo Castle
Children's Series	
VIC1911	The Sky Is Falling

Order Number	Product Name
VIC1912	Mole Attack
VIC1928	Home Babysitter
VIC1930	Visible Solar System
VIC1933	Bingo/Speed Math
VIC1935	Commodore Artist
VIC1941	Number Nabber

Home Improvement Cartridges

VIC1929	Personal Finance

VIC Programming Aid Cartridges

VIC1211A	VIC 20 Super Expander
VIC1212	Programmers Aid Cartridge
VIC1213	VICMon Machine Language Monitor

VIC Teach Yourself Programming Series

VL102	Introduction to BASIC Programming—Part I (Tape and book)
VL103	BASIC Programming—Part II (Tape and book)
VL110	GORTEK and the Microchips (Tape and book)
VIC1001	Waterloo BASIC (Disk)

VIC Application Programs on Tape

The following pre-recorded programs are designed for use with the Commodore Datassette tape recorder. Programs on tape come in several varieties and are color coded by category as follows: Recreation (red), Education (blue), Business Calculation (green), Home Utility (orange), and Computing Aid (black).

VT106A	Recreation Program Six Pack
VT107A	Home Calculation Program Six Pack
VT108	Math Improvement Six Pack (grades 2-6)
VT109	Sampler Six Pack
VT164	Programmable Character Set/ Gamegraphics Editor
VT232	Term 20/40—Terminal Emulator

Order Number	Product Name

Commodore 64 Software

Recreation Series on Cartridge

C64601	Jupiter Lander
C64602	Kickman*
C64603	Sea Wolf*
C64604	Bingo/Speed Math
C64605	Radar Rat Race
C64606	Clowns*
C64609	Visible Solar System
C64610	Tooth Invaders
C64612	Blueprint*
C64613	Lazarian*
C64614	Omega Race*
C64615	Wizard of Wor
C64616	Lamans
C64617	Pinball Spectacular
C64618	Gorf
C64621	Avenger
C64622	Super Smash
C64623	Star Post
C64624	Frogmaster
C64631	Star Ranger

*Bally Midway Games developed under Commodore's licensing agreement with Bally Manufacturing Company.

Educational Public Domain on Disk

C64700	Business "BA"
C64701	Geography "RA"
C64702	English "EA"
C64703	English "EB"
C64704	English "EC"
C64705	English "ED"
C64706	English "EE"
C64707	English "EF"
C64708	English "EG"
C64709	Math "MA"
C64710	Math "MB"
C64711	Math "MC"
C64712	Math "MD"

Order Number	Product Name
C64713	Math "ME"
C64714	Math "MF"
C64715	Math "MG"
C64716	Math "MH"
C64717	Comp Sci "CA"
C64718	Science "SA"
C64719	Science "SB"
C64720	Science "SC"
C64721	Science "SD"
C64722	Tech "TA"
C64723	History "HA"
C64724	Games "GA"
C64725	Games "GB"
C64726	Games "GC"

Mind-Challenging Games From INFOCOM on Disk**

C64625	Zork I
C64626	Zork II
C64627	Zork III
C64628	Suspended
C64629	Starcross
C64630	Deadline

** INFOCOM games developed under Commodore's licensing agreement with INFOCOM, INC. INFOCOM is a registered trademark of INFOCOM, INC.

EASY Business Series on Disk

C64200	Easy Calc 64
C64210	Word/Name Machine
C64202	Easy Finance I 64
C64212	Easy Finance II 64
C64213	Easy Finance III 64
C64214	Easy Finance IV 64
C64215	Easy Finance V 64
C64204	Easy Mail 64
C64207	Easy Script 64
C64208	Easy Spell 64
C64216	The Manager

Order Number	Product Name

Small Business System Series on Disk

C64220	General Ledger
C64221	Accounts Receivable
C64222	Accounts Payable
C64223	Payroll/Check Writing
C64224	Inventory Management
C64225	Codewriter

Learning Series

C64303	GORTEK and the Microchips (Tape)
C64310	Easy Lesson/Easy Quiz 64 (Disk)

Programming Series

C64101	Assembler 64 (Disk)
C64103	Bonus Pack (Disk)
C64104	C64 Super Expander (VSP) (Cartridge)
C64105	LOGO (Disk)
C64106	Pilot (Disk)
C64107	PET Emulator (Disk)
C64108	Simons' BASIC (Cartridge)
C64109	Screen Editor (Disk)
C64110	CPM 2.2 Operating System***
C64111	Nevada COBOL (Disk)
C64301	Intro to BASIC I (Tape)

Art and Music Series

C64402	Music Machine (Cartridge)
C64403	Music Composer (Cartridge)

Telecomputing Series

VT232	Term 20/40 Software Terminal Emulator (Tape)

*** CPM is a registered trademark of Digital Research, Inc.

SimplexSoft Ltd.

FINANCIAL RECORD SYSTEM
For VIC 20™—Commodore 64™

- Record, total, and/or print all income sources
- Record, total, and/or print all expenses
- Record, total, and/or print all the items on the most complex US tax forms (car, travel, entertainment, supplies, contributions, medical, local & state taxes, etc.)
- Review & Edit all entries by day, week, month, year, and/or by category (car, medical, taxes, etc.)
- Store & Retrieve all Data on tape or disk

TAX TIME WILL TAKE AN HOUR OR SO NOT DAYS OR WEEKS

IDEAL FOR:

- Individual & Multi-Income family financial & tax records
- Self employed & company Reps. on expense accounts
- Apartment owners and managers
- Small Contractors (Bldg., Plumbing, Heating, Elect., Etc.)
- Truck owners/operators
- Farmers
- Any other small businesses

NO KNOWLEDGE OF COMPUTER LANGUAGE NEEDED—ALL IN PLAIN ENGLISH

VIC 20 version requires 16K memory expander (Printer not required)
SimplexSoft 2 tape system only.....$29.95
Disk..... 34.95
Specify cassette or disk and computer model
Add $2.00 for mailing.

Send check or money order to:

SimplexSoft, Ltd.
617 N. Property Lane
Marion, Iowa 52302

VIC 20 and Commodore 64 are trademarks of Commodore Electronics Ltd.

DEALERS DON'T MISS OUT—ANYONE CAN DEMO THIS SYSTEM AND LOOK PROFESSIONAL

Reduction of an actual sign

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 Tape or Disk (Specify computer equipment)

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

Formulator A formula scientific calculator designed for tasks which require repetitive arithmetic computations. You can save formulas and numeric expressions. $39.95

Preschool Educational Programs ABC Fun; 123 Fun; and Ginger the Cat with: Addition and Subtraction, Number Hunt, and Letter Hunt. All programs have bright color, music, and action. Each $14.95

Microbroker Exciting, realistic and educational stock market simulation based on plausible financial events. $34.95 Tape or Disk

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.

Cardinal Software™

Distributed by
Virginia Micro Systems
13646 Jeff Davis Hwy
Woodbridge, VA 22191

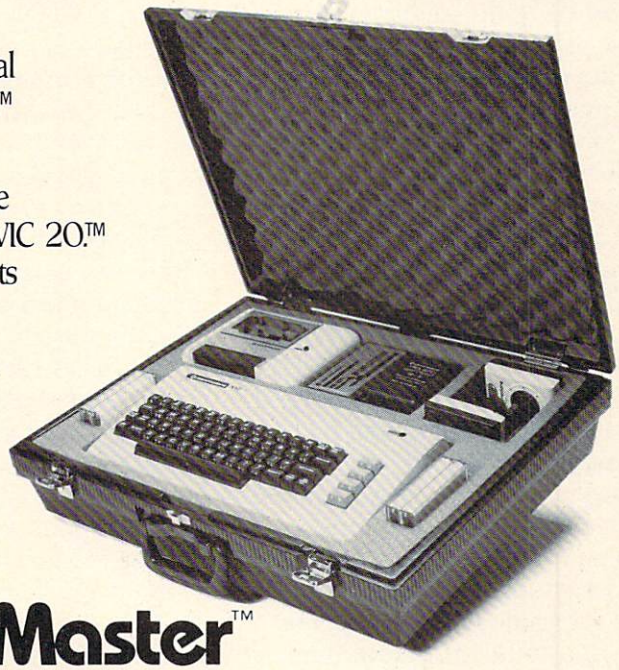
Commodore 64 and VIC-20 are registered trademarks of Commodore Electronics Ltd.

A Case of Convenience

Now that you can't live without your personal computer, why leave it behind? With TravelMaster,™ your Commodore™ goes wherever you do.

The foam interiors of TravelMaster cases are custom cut to cradle your Commodore 64™ or VIC 20.™ The rugged, double-wall, molded exterior protects each delicate piece.

Call your dealer today, he'll tell you how to travel with your computer safely, in style — and for a lot less than you might expect. What could be more convenient?



Commodore 64 and VIC 20 are trademarks of Commodore Electronics Ltd.

TravelMaster™

Manufactured by Southern Case, Inc.
TravelMaster Division • P.O. Box 28147 • Raleigh, NC 27611 • (919) 821-0877

A Lightning-Fast Machine Language Joystick

continued from page 33

move the handle. Inside the joystick are five switches. Four of the switches are for direction detection and one is for the fire button. Each switch corresponds to one of the lower five bits in the joy-port memory locations as mentioned above.

You may be wondering how you control eight different directions with only four switches. With four bits (one nybble!) of memory you can combine two different switch closures at once. For example, if you pushed the joystick toward the northeast, you would be pressing both the north and the east switches. This makes the diagonal positions possible. Another anomaly is that these bits are actually inverted, meaning that if we are pushing the joystick north we will return a zero in the bit that corresponds to north, or a one if we aren't pressing north. This also holds true for the fire button.

Now, on to the routine! First of all, we must load the accumulator with the value from the desired port, and shift it right one bit (LSR A). This will allow the rightmost bit to fall into the carry where we can test it for the proper weight, 0 or 1. If the carry becomes set or equal to one, we know we're not moving the joystick in that direction so we can branch to the next LSR A, and test another bit. If it is not equal to one, then we transfer the remaining value into the X register for safekeeping. This value will be the original value minus the shifted bit.

After doing this bit of housekeeping we are ready to gosub (JSR) the subroutine. Each of the small subroutines will handle the math and at the same time keep watch to see if we're at the screen limits. When we return from the subs we restore the accumulator with a transfer of the stored value we stuck in the X register. After testing all five bits we can return to the BASIC program that you were using to call this routine.

Now a little about the subroutines: since the X direction can be equal to 320 decimal or 140 hex we must set aside two memory locations for storing the values, updating them each time we call the joystick routine. The Y direction needs only one byte, which makes it a little easier than X. Knowing this we only need to check for a one in the high byte at any time.

To quickly go through this, let's say we are pushing left. We first check the low byte for "not zero". If it is not zero (or equal to one) we can then check the high byte for zero. This will tell us that we are at the edge of the screen so we put $140 into X to bring it back to the other side for wraparound. Note: If you don't want wraparound then you must put zeros back into the X coordinate, in both the high and low bytes.

If the low byte was greater than zero we know to decrement it by one and return to the main routine used to call this sub. If the low byte was zero and the high byte was one we zero out the high byte and decrement the low byte. In machine language when you decrement from zero you return the value of 255 or $FF hexadecimal. In other words, we took care of any carry that would have been generated from the high byte.

The other directional subroutines work basically the same whether decrementing or incrementing. Be sure and study the disassembly a little bit.

You may have been using Bill Hindorff's routine out of the *Reference Guide*, and just adding or subtracting the values returned, but I think that combining the two together would be somewhat easier to follow and use. Of course the worst is to come. To really speed things up we will have to do the math required to plot the proper points. But I should save all for another time (another article?). So have fun and please do yourself a favor. EXPERIMENT!!!!! **C**

Danny Byrne is the Vice President and co-founder (with his wife Betsy) of the New Mexico Commodore Users Group. He taught himself BASIC programming on his first computer, a Commodore VIC 20, and four months later sold his first program to Commodore, a joystick drawing program called "Alpha Draw", that is currently available in the Sampler Sixpack. Danny now works mainly on the Commodore 64, in machine language and assembler. The Byrnes have four children, and are jointly working on several kids' educational programs. Danny and a collaborator, Tim Villaneuva, are putting together a book of routines and tutorials, for both machine language and BASIC users.

Hi-Res Joystick

```
LINE# LOC    CODE          LINE

00001  0000                ;*********************************
00002  0000                ;---------------------------------
00003  0000                ;     HIRES JOYSTICK REV3.83
00004  0000                ;             BY
00005  0000                ;          DANNY  BYRNE
00006  0000                ;    ALBUQUERQUE, NEW MEXICO
00007  0000                ;---------------------------------
00008  0000                ;*********************************
00009  0000                ;
00010  0000                ;
00011  0000                ;THIS ROUTINE READS THE CURRENT
00012  0000                ;DIRECTION ON EITHER JOYSTICK AND
00013  0000                ;WILL UPDATE THE X AND Y STARTING
00014  0000                ;VALUES. IT HANDLES THE SCREEN
00015  0000                ;LIMITS AND WRAPS AROUND FOR YOU.
00016  0000                ;
00017  0000                ;---- MEMORY LOCATIONS USED -----
00018  0000                ;
00019  0000                ;
00020  0000                *=$C100                 ;ASSEMBLE CODE THIS
                                                   ADDRESS
00021  C100                JOY1    =$DC00          ;JOYSTICK ONE PORT
00022  C100                JOY2    =JOY1+1         ;JOYSTICK TWO PORT
00023  C100                        XLO =$C500      ;TEMP LO BYTE X STORED
00024  C100                        XHI =XLO+1      ;HI BYTE X (255-320)
00025  C100                        YLO =XLO+2      ;Y ONLY HAS LO BYTE
00026  C100                        FBP =$C50F      ;FIRE BUTTON
00027  C100                ;
00028  C100                ;
00029  C100                ;*********************************
00030  C100                ;------- CODE STARTS HERE --------
00031  C100                ;*********************************
00032  C100                ;
00033  C100                ;--------------- CHECK FOR UP ----
00034  C100  AD 01 DC      UP      LDA JOY2        ;READ JOYSTICK HERE
00035  C103  4A                    LSR A           ;SHIFT FIRST BIT FOR UP
00036  C104  B0 05                 BCS DOWN        ;IF BIT IS ONE NOT
00037  C106  AA                    TAX             ;PRESSED SO BRANCH,
                                                   ELSE
00038  C107  20 32 C1              JSR JUP         ;STORE VALUE AND GOTO
00039  C10A  8A                    TXA             ;UP ROUTINE. COME
                                                   BACK AND
```

```
LINE# LOC   CODE        LINE

00040  C10B             ;--------------- CHECK FOR DOWN --
00041  C10B  4A          DOWN   LSR A            ;CHECK FOR LEFT
00042  C10C  B0 05             BCS LEFT          ;BRANCH IF SET
00043  C10E  AA               TAX               ;STORE VALUE AGAIN
00044  C10F  20 40 C1         JSR JDOWN         ;DO ROUTINE FOR DOWN
00045  C112  8A               TXA               ;RESTORE VALUE FOR
                                                 NEXT BIT
00046  C113             ;--------------- CHECK FOR LEFT --
00047  C113  4A          LEFT   LSR A            ;SHIFT AND CHECK
00048  C114  B0 05             BCS RIGHT         ;NOT LEFT SKIP THEN
00049  C116  AA               TAX               ;TO RIGHT, ELSE SAVE
00050  C117  20 50 C1         JSR JLFT          ;VALUE AND DO ROUTINE
00051  C11A  8A               TXA               ;RESTORE VALUE AND
                                                 CHECK
00052  C11B             ;--------------- CHECK FOR RIGHT -
00053  C11B  4A          RIGHT  LSR A            ;NEXT BIT IF
00054  C11C  B0 05             BCS BUTT          ;NOT RIGHT CHECK FIRE
00055  C11E  AA               TAX               ;BUTTON ELSE DO
                                                 ROUTINE
00056  C11F  20 6E C1         JSR JRHT          ;THEN CHECK FOR FIRE
00057  C122  8A               TXA               ;BUTTON NEXT
00058  C123             ;--------------- CHECK FIRE BUTTON-
00059  C123  4A          BUTT   LSR A            ;LAST BIT TO SHIFT
00060  C124  B0 06             BCS NFB           ;IF BUTTON NOT PRESSED
00061  C126  A9 01             LDA #1            ;THEN BRANCH TO NFB.
00062  C128  8D 0F C5         STA FBP           ;ELSE STORE ONE
                                                 IN MEMORY
00063  C12B  60               RTS               ;RETURN TO MAIN
                                                 PROGRAM
00064  C12C             ;----------------- NOT PRESSED ---
00065  C12C  A9 00        NFB    LDA #0           ;STORE ZERO FOR
00066  C12E  8D 0F C5         STA FBP           ;NO FIRE BUTTON
00067  C131  60               RTS
00068  C132             ;
00069  C132             ;=================================
00070  C132             ;
00071  C132             ;THE FIRE BUTTON CAN BE USED TO
00072  C132             ;SIMULATE RAPID FIRE OR ANY USER
00073  C132             ;DEFINED FUCTION LIKE ERASE MODE
00074  C132             ;
00075  C132             ;=================================
00076  C132             ;
```

```
LINE# LOC   CODE        LINE

00077  C132             ; JOYSTICK SUBROUTINES HERE
00078  C132             ;=================================
00079  C132             ;NOW THE SUBS TO HANDLE UPDATING
00080  C132             ;THE JOYSTICK VALUES.  DIRECTION &
00081  C132             ;SCREEN WRAPAROUND ARE STORED IN
00082  C132             ;THE ORIGINAL X AND Y LOCATIONS
00083  C132             ;=================================
00084  C132             ;---------------------- UP --------
00085  C132  AD 02 C5   JUP    LDA YLO          ;LOAD Y AND CHECK
00086  C135  D0 05             BNE DCY          ;FOR NOT ZERO
00087  C137  A9 C9             LDA #$C9         ;IF IT IS THEN MAKE IT
00088  C139  8D 02 C5          STA YLO          ;PUT IT ON BOTTOM
00089  C13C  CE 02 C5   DCY    DEC YLO          ;ELSE DECREMENT
00090  C13F  60                RTS              ;RETURN FOR NEXT BIT
00091  C140             ;---------------------- DOWN ------
00092  C140  EE 02 C5   JDOWN  INC YLO          ;NOW DOWN SO CHECK
00093  C143  AD 02 C5          LDA YLO          ;TO SEE IF AT BOTTOM
00094  C146  C9 C9             CMP #$C9         ;IF NOT THEN ALREADY
00095  C148  90 05             BCC ICY          ;INCREMENTED SO RETURN
00096  C14A  A9 00             LDA #0           ;ELSE ZERO THEN RETURN
00097  C14C  8D 02 C5          STA YLO          ;STORE FOR TOP OF
                                                 SCREEN
00098  C14F  60         ICY    RTS
00099  C150             ;---------------------- LEFT ------
00100  C150  AD 00 C5   JLFT   LDA XLO          ;GET LOW BYTE AND
00101  C153  D0 0A             BNE DCXL         ;CHECK FOR ZERO
00102  C155  AD 01 C5          LDA XHI          ;NOW HIGH CHECK FOR
                                                 ZERO
00103  C158  F0 09             BEQ DCX          ;IF SO, AT EDGE, GO AND
00104  C15A  29 00             AND #0           ;SET FOR RIGHT EDGE
                                                 ELSE
00105  C15C  8D 01 C5          STA XHI          ;SET HI BYTE ONLY AND
00106  C15F  CE 00 C5   DCXL   DEC XLO          ;DEC LOW BYTE
00107  C162  60                RTS              ;RETURN
00108  C163  A9 01       DCX   LDA #1           ; THIS SETS TO OTHER
00109  C165  8D 01 C5          STA XHI          ;SIDE OF SCREEN
00110  C168  A9 40             LDA #$40
00111  C16A  8D 00 C5          STA XLO
00112  C16D  60                RTS
00113  C16E             ;---------------------- RIGHT -----
00114  C16E  AD 00 C5   JRHT   LDA XLO          ;THIS ROUTINE IS
00115  C171  C9 3F             CMP #$3F         ;BASICALLY THE SAME BUT
```

```
LINE# LOC   CODE        LINE

00116  C173  90 0E           BCC ICX          ;INCREMENTING AND IF
00117  C175  AD 01 C5        LDA XHI          ;LOW AND HIGH BYTES ARE
00118  C178  F0 09           BEQ ICX          ;EQUAL TO 320 DECIMAL
00119  C17A  A9 00           LDA #0           ;WE RESET TO ZERO FOR
00120  C17C  8D 01 C5        STA XHI          ;LEFT SIDE OF SCREEN
00121  C17F  8D 00 C5        STA XLO          ;THEN RETURN
00122  C182  60              RTS
00123  C183  EE 00 C5  ICX    INC XLO          ;ELSE WE INCREMENT
00124  C186  D0 05           BNE DUN          ;BEING CAREFUL OF A
00125  C188  A9 01           LDA #1           ;CARRY TO THE HIGH BYTE
00126  C18A  8D 01 C5        STA XHI          ;THEN RETURN
00127  C18D  60           DUN RTS
00128  C18E                ;
00129  C18E                ;================================
00130  C18E                ;$C500 WILL HOLD LOW BYTE OF X
00131  C18E                ;DIRECTION AND $C501 THE HIGH BYTE
00132  C18E                ;$C502 CONTAINS Y VALUE.  THE FIRE
00133  C18E                ;BUTTON IS STORED IN $C50F (=1 IF
00134  C18E                ;PRESSED ).
00135  C18E                ;================================
00136  C18E                ;
00137  C18E                ;
00138  C18E                ;
00139  C18E                ;**********************************
00140  C18E                ;       CODE FOR LEFT-RIGHT
00141  C18E                ;           AND UP-DOWN
00142  C18E                ;**********************************
00143  C18E                ;
00144  C18E                ;JUST INSERT THIS CODE INTO YOUR
00145  C18E                ;PROGRAM AND USE THE ENTRY POINTS
00146  C18E                ;FOR LEFT-RIGHT OR UP-DOWN MOVE-
00147  C18E                ;MENT TO SUIT YOUR REQUIREMENTS
00148  C18E                ;
00149  C18E                ;--------------------- UP-DOWN ---
00150  C18E                ;------------- UP ----------------
00151  C18E  AD 00 DC  UP2    LDA JOY1
00152  C191  4A               LSR A
00153  C192  B0 05            BCS DWN2
00154  C194  AA               TAX
00155  C195  20 32 C1         JSR JUP
00156  C198  8A               TXA
00157  C199                ;----------- DOWN ---------------
```

```
LINE# LOC   CODE        LINE

00158  C199  4A          DWN2   LSR A
00159  C19A  B0 05               BCS BUTT2
00160  C19C  AA                  TAX
00161  C19D  20 40 C1            JSR JDOWN
00162  C1A0  8A                  TXA
00163  C1A1                ;---------- FIRE BUTTON ---------
00164  C1A1  4A            BUTT2  LSR A
00165  C1A2  B0 06               BCS NOFB
00166  C1A4  A9 01               LDA #1
00167  C1A6  8D 0F C5            STA FBP
00168  C1A9  60                  RTS
00169  C1AA                ;---------- NOT PRESSED ----------
00170  C1AA  A9 00         NOFB   LDA #0
00171  C1AC  8D 0F C5            STA FBP
00172  C1AF  60                  RTS
00173  C1B0                ;--------------------LEFT-RIGHT--
00174  C1B0                ;---------- LEFT ----------------
00175  C1B0  AD 00 DC      UP3    LDA JOY1
00176  C1B3  4A                  LSR A
00177  C1B4  4A                  LSR A
00178  C1B5  4A                  LSR A
00179  C1B6  B0 05               BCS RGT2
00180  C1B8  AA                  TAX
00181  C1B9  20 50 C1            JSR JLFT
00182  C1BC  8A                  TXA
00183  C1BD                ;------------ RIGHT -------------
00184  C1BD  4A            RGT2   LSR A
00185  C1BE  B0 05               BCS BUTT3
00186  C1C0  AA                  TAX
00187  C1C1  20 6E C1            JSR JRHT
00188  C1C4  8A                  TXA
00189  C1C5                ;---------- FIRE BUTTON ---------
00190  C1C5  4A            BUTT3  LSR A
00191  C1C6  B0 06               BCS NTFB
00192  C1C8  A9 01               LDA #1
00193  C1CA  8D 0F C5            STA FBP
00194  C1CD  60                  RTS
00195  C1CE                ;---------- NOT PRESSED ----------
00196  C1CE  A9 00         NTFB   LDA #0
00197  C1D0  8D 0F C5            STA FBP
00198  C1D3  60                  RTS
```

```
LINE# LOC   CODE        LINE

00199  C1D4             ;
00200  C1D4             ;
00201  C1D4             ;
00202  C1D4             .END
```

```
ERRORS = 00000


SYMBOL TABLE

SYMBOL VALUE
 BUTT      C123    BUTT2     C1A1    BUTT3     C1C5    DCX       C163
 DCXL      C15F    DCY       C13C    DOWN      C10B    DUN       C18D
 DWN2      C199    FBP       C50F    ICX       C183    ICY       C14F
 JDOWN     C140    JLFT      C150    JOY1      DC00    JOY2      DC01
 JRHT      C16E    JUP       C132    LEFT      C113    NFB       C12C
 NOFB      C1AA    NTFB      C1CE    RGT2      C1BD    RIGHT     C11B
 UP        C100    UP2       C18E    UP3       C1B0    XHI       C501
 XLO       C500    YLO       C502

END OF ASSEMBLY
```

```
.:C100 AD 01 DC 4A B0 05 AA 20
.:C108 32 C1 8A 4A B0 05 AA 20
.:C110 40 C1 8A 4A B0 05 AA 20
.:C118 50 C1 8A 4A B0 05 AA 20
.:C120 6E C1 8A 4A B0 06 A9 01
.:C128 8D 0F C5 60 A9 00 8D 0F
.:C130 C5 60 AD 02 C5 D0 05 A9
.:C138 C9 8D 02 C5 CE 02 C5 60
.:C140 EE 02 C5 AD 02 C5 C9 C9
.:C148 90 05 A9 00 8D 02 C5 60
.:C150 AD 00 C5 D0 0A AD 01 C5
.:C158 F0 09 29 00 8D 01 C5 CE
.:C160 00 C5 60 A9 01 8D 01 C5
.:C168 A9 40 8D 00 C5 60 AD 00
.:C170 C5 C9 3F 90 0E AD 01 C5
.:C178 F0 09 A9 00 8D 01 C5 8D
.:C180 00 C5 60 EE 00 C5 D0 05
.:C188 A9 01 8D 01 C5 60 AD 00
.:C190 DC 4A B0 05 AA 20 32 C1
.:C198 8A 4A B0 05 AA 20 40 C1
.:C1A0 8A 4A B0 06 A9 01 8D 0F
.:C1A8 C5 60 A9 00 8D 0F C5 60
.:C1B0 AD 00 DC 4A 4A 4A B0 05
.:C1B8 AA 20 50 C1 8A 4A B0 05
.:C1C0 AA 20 6E C1 8A 4A B0 06
.:C1C8 A9 01 8D 0F C5 60 A9 00
.:C1D0 8D 0F C5 60 00 00 00 00{CU}
.
.
```

Joystick Loader

```
10 R=0:FOR X=49408 TO 49619
20 READ A:R=R+A:POKE X,A:NEXT
30 IF R<>23350 THEN PRINT"ERROR[SPACE]IN[SPACE]DATA[SPACE]
   STATEMENTS":END
35 REM ***LINES 40 AND 50 ARE FOR DEMONSTRATION***
36 REM ***ONLY AND MAY BE OMITTED***
40 SYS 49408
50 PRINT PEEK(50432)+256*PEEK(50433);:PRINT PEEK(50434)
   :GOTO 40
100 DATA 173, 1, 220, 74, 176, 5, 170, 32, 50, 193
110 DATA 138, 74, 176, 5, 170, 32, 64, 193, 138, 74
120 DATA 176, 5, 170, 32, 80, 193, 138, 74, 176, 5
130 DATA 170, 32, 110, 193, 138, 74, 176, 6, 169, 1
140 DATA 141, 15, 197, 96, 169, 0, 141, 15, 197, 96
150 DATA 173, 2, 197, 208, 5, 169, 201, 141, 2, 197
```

Be Careful When You Dial Us

When you call Commodore, please double check the numbers before you dial, and be sure you dial carefully. Several people in our area who have similar phone numbers have been receiving a distressing number of calls intended for us. Our numbers are:

Executive Offices:	215-431-9100
Customer Support:	215-436-4200
Magazine Subscriptions:	800-345-8112
(In Pennsylvania 800-662-2444)	

Butterfield's "Bits & Bytes" TV Series Aired in the U.S.

"Bits & Bytes", a Canadian TV series put together under the watchful eye of Jim Butterfield, has been purchased by the Public Broadcasting System for airing in the United States this fall.

The series is designed to help beginning computerists—or people thinking about buying a computer—understand their equipment and its potential. Set up as a tutorial, each half-hour program covers a specific subject, ranging from the most rudimentary discussion of what RAM and ROM are to more sophisticated topics like modems and programming languages.

Although the program is not specifically Commodore-oriented, you do get to see a lot of Commodore equipment—and can pick up some interesting information as well, even if you're an experienced hacker.

Too bad PBS didn't buy the other part of this series, titled "The Academy", which on Canadian TV ran immediately following "Bits & Bytes". "The Academy" is a review of and commentary on the information presented in "Bits & Bytes"—and features Jim Butterfield in the flesh. (He does not actually appear in "Bits & Bytes".)

"Bits & Bytes" was produced by TV Ontario in association with the Ontario Teachers Federation. Watch your local listings for its appearance.

```
160 DATA 206, 2, 197, 96, 238, 2, 197, 173, 2, 197
170 DATA 201, 201, 144, 5, 169, 0, 141, 2, 197, 96
180 DATA 173, 0, 197, 208, 10, 173, 1, 197, 240, 9
190 DATA 41, 0, 141, 1, 197, 206, 0, 197, 96, 169
200 DATA 1, 141, 1, 197, 169, 64, 141, 0, 197, 96
210 DATA 173, 0, 197, 201, 63, 144, 14, 173, 1, 197
220 DATA 240, 9, 169, 0, 141, 1, 197, 141, 0, 197
230 DATA 96, 238, 0, 197, 208, 5, 169, 1, 141, 1
240 DATA 197, 96, 173, 0, 220, 74, 176, 5, 170, 32
250 DATA 50, 193, 138, 74, 176, 5, 170, 32, 64, 193
260 DATA 138, 74, 176, 6, 169, 1, 141, 15, 197, 96
270 DATA 169, 0, 141, 15, 197, 96, 173, 0, 220, 74
280 DATA 74, 74, 176, 5, 170, 32, 80, 193, 138, 74
290 DATA 176, 5, 170, 32, 110, 193, 138, 74, 176, 6
300 DATA 169, 1, 141, 15, 197, 96, 169, 0, 141, 15
310 DATA 197, 96
```

Computers in the Church: Journeys and Visions

by Jim Strasma

Although many of you know Jim Strasma and his wife, Ellen, as authors of several books about Commodore computers and editors of The Midnite/PAPER, *a Commodore-oriented magazine, you may not have known that Jim is also a United Methodist pastor.*

In this article Jim accomplishes two tasks: first, he provides valuable information on how computers can help relieve a pastor's daily burden of "administrivia". (This information, by the way, can certainly be applied to any administrative function, whether religious or secular.) Second, and equally important, he also gives you some rare insights into the earliest days of micros (way back in 1977), when 4K Altairs sold in kits were state-of-the-art technology.

One thing you have to understand is that I'm a gadget freak. I've always had a warm place in my heart and pocketbook for electronic toys. We bought our first calculator in seminary, in February of 1971. At $100 for a simple four-function unit, it was nearly out of reach financially. Even so, I had to have it. Likewise with watches... I still had a perfectly good Omega when I bought my first electronic watch. And I still have that Timex now, three electronics later. With my background, it shouldn't surprise anyone that I bought the first church computer in Illinois.

The Need

Much of the daily work of a pastor has little to do with what we learned in seminary. At least a quarter of a typical work day is taken by "administrivia"—the dull routine of administration. Some of it can't be avoided, like the once a year chore of finding a sucker, er volunteer, to put the Christmas star on top of the steeple again in the snow. But other tasks are so mechanical it's a shame humans have to do them.

The chore that first turned my thoughts to computers was the newsletter. At the time I was pastor of a small-town church of about 200 members. For the first time in some years we had a newsletter—a one-pager I wrote once a month. I considered it a good tool for evangelism and keeping our message before the community, but the mechanics of preparing it were a constant frustration. Mailing labels were especially painful. People had this awful habit of moving or dying just after we retyped the master list....

First Contact

In January of 1977 I enjoyed my first pastoral junket—a week studying world hunger in New York City. It was a good seminar, but the best part was my one afternoon off. I knew New York was famous for gadgets and I was determined to find some. But even I wasn't prepared for what I found. While strolling through Manhattan, I happened past the second computer store in the country, barely a month after it opened.

Back in high school I'd had three days of IITran computer programming, followed by a semester course in FORTRAN at DePauw University. In those days computers were room-filling monsters fed by keypunched cards. I spent most of that semester standing in line for the only two keypunch machines on campus.

The computer store in Manhattan was different. Its computers were as small as a bread box and there wasn't a keypunch in sight. That $26,000 monster had been replaced by a simple keyboard-and-TV combination and results were saved on ordinary cassette tapes. The best part was the

price—only $3,495 for an Altair system with 4K RAM. Why that was nearly as much storage as DePauw's IBM computer!

Of course there were disadvantages to the new units. They were only available as kits and no one seemed sure where to get some parts. Nor would anyone promise it would work after I put it together. But that didn't matter. I didn't have $3,495 anyway. So I bought two books and left.

I didn't sleep much that night. Instead I read about microprocessors—the tiny integrated circuit chips that made cheap computers possible. It turned out choosing a computer was no easy task. There were three competing chips, each incompatible with the others. There were also a dozen or more makers of computer kits, each largely incompatible with the others.

Next I read about BASIC, the language used by microcomputers. It's like FORTRAN but simpler, the book said. Invented at Dartmouth College in 1964 as a Beginners' All-purpose Symbolic Instructional Code, its primary virtue was that it was small enough to fit in most any computer.

You Can't Go Home Again

Back home in Illinois I kept reading about computers, but didn't think about buying until May of 1977. That was the month *Popular Science* magazine featured a home computer on its cover. According to the article, the Commodore PET was a real computer, selling for the unheard of price of $495! Better yet, it was fully assembled and worked as soon as it was plugged in! And its BASIC was built in on special chips called ROMs.

Unfortunately, I couldn't afford the PET, either. But that winter I moved to a huge church as Minister of Evangelism. And immediately I knew I had to get a computer. Otherwise I would be buried by paperwork when I should be visiting newcomers. I was responsible for monitoring the attendance of nearly three thousand members and an equal number of visitors. Of these, seven hundred members and one hundred visitors could be counted on signing the attendance register every Sunday. As you might expect, my desk was covered with card files, and I gave thanks daily for my secretary.

Normally I'd have pushed strongly for the church itself to buy a computer. But my new senior pastor, much though I love him, is as adverse to gadgets as I am fond of them. And then as now, the sole budget priority of that church is finding a way to add parking and enlarge the sanctuary. Since I agreed with that goal, I decided to buy my own computer.

The method I chose is open to most pastors. I put all gifts from weddings and funerals into a computer fund. I've always accepted such offerings, using them for things the church needs but can't bring itself to buy. Fortunately, Grace Church has lots of weddings....

While waiting for my savings to grow I also did something else most pastors can do. I borrowed a computer owned by a parishioner, which was primarily used by his teen-age son. That meant it was free during school hours.

The Plunge

During the summer of 1978, by a process of elimination, I found myself the owner of a Commodore PET computer. It took about one month to teach myself BASIC and three more months to finish a program to monitor church attendance. Since that first PET had only 8K of memory, I divided my data into groups of 100 names, keeping each group on a separate cassette. The program told me when to put in each cassette. That method worked—and well—but shuffling a dozen data cassettes was time-consuming and inconvenient.

One partial solution to the tape shuffling was to add memory. By June of 1979 three different companies offered plug-in cards to increase the program workspace in the computer. That month Commodore also introduced a new model that already included 32K of memory and a real keyboard. Surprisingly, the net cost of selling my old PET and buying the new model was the same as keeping the old model and adding memory. Needless to say, I traded up.

My next upgrade came that November. A computer is not much use to a church without a printer, especially when the computer is at home and the office at the church. I'd thought about keeping the computer at the office, but couldn't for two reasons. Grace Church, which was suffering about one break-in a month at the time, didn't have insurance to

cover my equipment. Second, computers are very addictive and my wife would have resented my staying at the office as late as I would have if the computer were there.

Instead, I bought a cheap printer. Though it's now possible to buy a daisy-wheel printer for $800, it only bought me a dot-matrix printer in 1979. Even so, the printer gave me more usable attendance reports and allowed me to begin using the computer as a typewriter for sermons. This feature alone saved two hours per week and gave me a far better-looking manuscript on Sunday morning.

Beyond BASIC

Then I hit another barrier. BASIC simply couldn't maintain 3,000 names in alphabetical order. Each sort of the list took 48 hours of computer time, and it had to be done monthly. This is a typical problem. Any computer can seem fast until it has a full load of data to juggle. Anyone shopping for a computer should test it with as much information as possible before buying. To solve the sort problem I learned to write programs in machine language—the computer's native tongue. It wasn't easy, but it did work. The resulting program arranges the 3,000 names in under a minute.

A Full System at Last

There are advantages to being first. It turned out that many other people needed my sort and would pay for a copy. That provided enough money to buy a dual disk drive and complete my system. Having two drives is a safety measure. With two, it is easy to back up disks with a spare copy of important information.

Anyone with my money problems in buying a computer now has one wonderful option not open to me years ago—leasing. Buying as I did, a bit at a time, is far more expensive in the long run than getting what you need at the start. It also makes the system less useful until completed. Most reputable computer dealers now gladly lease computers by the month. The terms are typically for three years, with a cheap option to buy the computer at the end. Therein lies a dilemma: by then your computer will still be in perfect working order, but hopelessly obsolete. Each year for the past ten seems to have brought a doubling of computer memory and disk storage without an increase in price. In fact, there may never be a good time to buy a computer, because there will always be something better and cheaper just over the horizon.

The solution is to buy a model that is immediately useful—one that already has the programs you insist on using. Then it won't matter that it becomes obsolete; you'll still use it for the chores it has always done, even if you buy another computer for other chores.

I've done this myself, keeping my now-old computer while adding another for my wife to use in her work of editing. We each have a computer and disk drives, all connected to work together. This is an important consideration for larger churches, where several people share a common disk drive and printer. If this matters to you, be sure systems you consider provide for it.

The Rest of the Story

I've left out some of my story: the move to my own church, where I used the computer for everything from budgets to bulletins with the help of a new daisy printer; and the further move to a special appointment, freeing me to work full-time with this exploding new technology. It has been estimated that interest in computers among pastors doubled every month during 1982. Based on my mail, 1982 may be remembered as the year churches discovered computers.

The Church Market

Many churches have the mistaken idea that they need a special church package in order to use a computer. In most cases this is not true. The daily needs of a church office are more like those of other businesses than most pastors comfortably admit. Custom church packages do exist—some good and some awful. Good ones tend to be expensive—my home church of 1,700 members just happily paid $22,000 for a computer and a package of custom church programs. It works well, and requires almost no skill in the user. But it does little that commonly available secular packages couldn't do as well for $15,000 less, providing some skilled help is available. Few churches are now so small or remote as to be without such help. These secular packages include five major areas of utility: word processing,

data management, accounting, education and networking. Let's look briefly at each.

Word Processing

Word processing covers everything churches do now with a typewriter and more. The computer is ideal for weekly sermons, bulletins and newsletters. Need I remind anyone how little in a typical church bulletin changes from week to week? In churches where having a letter-perfect bulletin is considered essential, a computer should save four hours a week. Then too, having a computer gives a whole new meaning to the concept of the sermon barrel. If all your sermons are in computer files, adapting one for re-use is simplicity itself. It even improves the sermon the first time around. How often have you wanted to move an idea to a different part of the sermon but didn't because it would mean retyping? On the computer such moves take only a few seconds.

However, the word processor shines best when combined with mail list information. At Grace Church I sent offset-printed welcome letters to each new visitor. After I got a daisy-wheel printer, I sent personalized letters instead. It didn't take a minute longer, and the resulting letters were far warmer than the printed ones. Using essentially the same technique, it is possible to send individualized letters to every family in the church instead of the usual canned Christmas, Easter and Stewardship campaign letters.

Do be sensible in this. If you write more than thirty lines or make the results too professional-looking, everyone will know you used a computer, no matter what you say. And read the results before sending them. We've all received examples of computer-generated gibberish: "Good Morning Mrs. Methodist, and how are things there in Box 66?"

Data Management

The congregational mail list can become the basis of a member and visitor information system using any of the better database manager packages sold today—with several resulting benefits. The first of these is more up-to-date mail labels. Correcting a computer mail list is usually faster than updating a manual system so it tends to be done more promptly. Since the postal service charges for every returned newsletter, updating addresses weekly can mean real savings. It also protects against causing offense in mailings to recently bereaved, divorced or married families.

Nearly all common data management packages can also keep records of such information as last attendance, pledge, birthdays, church jobs and groups, neighborhood, workplace and emergency contacts. Once such information is readily available, sending a postcard to everyone in the choir or printing the names of members with April birthdays is a trivial chore. And because the information is all together in one place, it is easy to update.

Additionally, all pastors know the value of having a new visitor contacted by a member of the church who lives nearby, or works in the same plant or shares a common interest. Once such information is in the computer it is simple to have the computer search the records for appropriate persons to contact newcomers. A database can also help by making sure no qualified volunteer is overlooked in recruiting people to fill church jobs.

One other benefit of such information is as an early warning system for personal difficulties or dissatisfaction. Typically, families in crisis and families that are upset will change their patterns of attendance and giving. If this is noticed within three weeks or so, effective help can usually be given. This is especially true when the database lists another person in the congregation trusted by the family. On the other hand, if the problem goes unnoticed for six weeks or more, such families may rightly conclude that no one in the church cares much about them.

Accounting

Financial record-keeping is one of the main interests churches have in computers at the moment. A good accounting package can keep track of donors and donations to various funds, as well as expenses in any number of budget categories. This, however, is one task that needs special programming. It would never do, for example, to send members a bill instead of a pledge report. And most sizeable churches need separate reports for a multitude of special groups and activities (known as fund accounting).

Personally, I consider this one of the last areas most churches should computerize. People will forgive a misspelled name or a mis-directed letter, but too few forgive a financial error. On the other hand, a good accounting package can provide a church with a great deal of information that may help it be a better steward of this world's resources.

Similarly, a budget-planning spreadsheet program can greatly aid budgeting and provide early warning about congregational progress toward budget income and expense goals, particularly when it includes corrections for seasonal variations in budgeted income and expense items.

Education

This area is only beginning to be explored with Bible games and other computer-assisted instruction (called CAI). One exciting use of this technology presently in operation is in public settings such as malls and fairs. Here suitably programmed computers are carrying on "conversations" about God with people who would rather die than speak to a pastor. This same technique is now being extended to two-way TV systems such as Telidon in Canada.

Networking

In addition to Telidon, computers are now being linked in a multitude of other ways. We will shortly see local church computers phoning in annual church reports to other computers in area offices. These in turn will forward area summaries to still other computers in national offices. This will avoid at least two typings of all such information. Likewise, when the national church needs to get news out in a hurry, the task need only take hours instead of weeks once each church is able to receive electronic mail.

For More Information

If you have a specific question about computers in the church, you may write Rev. Strasma at 1238 Richland Avenue in Lincoln, IL 62656. If you need an answer, please include a stamped self-addressed reply envelope. C

How to Start Working With Your New Computer System

by Donald E. Hassler

Some advice on how to get started with a new Commodore business system.

You've all read articles on how to choose a system. And you've all talked to specialists *ad nauseum*. I am assuming that you've all done the groundwork, the system is ready to go in your office and you have the programs on hand.

The first rule in getting started with your new Commodore system is: don't try to do everything at once. Go slowly. Test everything and *don't stop your present accounting system*—at least not until everything is working properly on the computerized system. Start with just one accounting module. In our operation we started with the payroll because payroll is one of the most time-consuming procedures accounting people do. So it lends itself to having the data handled by your Commodore system.

Enter all your employees in the payroll file (master file, data file—whatever they call it). Run off the list and see how it looks. Your computer entry clerk will find this a good way to learn how to use the Commodore system's keyboard. After every employee is on the list, do a sample payroll. You should even run a payroll on the computer in tandem with whatever other system you have. In other words, test, test, TEST!!

The second rule for getting started is: keep one person primarily responsible for all computer keyboard work. The best choice is a good typist that has some bookkeeping background. Also, that person should be fast on a ten-key number pad. (Don't buy a Commodore computer for your business that doesn't have a ten-key pad. That is very important!!)

Start the payroll for a period after the end of a quarter. That'll make it much easier for 941 reports. When you have the payroll running to your satisfaction it's time for the next step: accounts payable. Start the A/P system just the way you did with payroll. Enter the accounts, run lists, try entries and generally get familiar with the system's personality. (What's this? Computer programs with personalities? Yes, Virginia, they have them. They are basically the personalities of the system designers.)

Stop now and be sure both packages are running well. Any bugs? Talk to your supplier or consultant to work them out. Be sure every bug has been taken care of. Now you can consider putting your general journal and general ledger on the Commodore system. Once again, start with a fiscal period, preferably the fiscal year.

Review the chart of accounts carefully. Plan for too many accounts if you're not sure. Change your chart of accounts as necessary. Which brings us to rule three: don't try to make the business accounting software fit your books exactly. Be flexible. This is the time to make improvements. No system will do exactly what you do by hand or with an outside accountant. And be creative. The Commodore system can handle all the numbers you can ever throw at it.

After the general ledger is successful for a quarter you can consider other refinements to the system. You may wish to add accounts receivable, order entry or even an inventory package. Most systems include these options. With inventory, beware!! There is no simple unit inventory program that will work for every type of business, so for inventory you might even consider a good database manager with reports designed to emulate a system you are already familiar with.

If you are running a cash-intensive retail business you need one more item. You need a cash entry system that will take all your daily transactions and organize them. There are several good systems available. We publish a sales analysis and accounts receivable package that works well for this. Others do too.

Above all, don't expect the Commodore system to make a set of bad books look good. You still need good bookkeeping practices and close cash control. A Commodore computer WILL make things faster and will almost eliminate clerical errors. You will soon be getting statements by the tenth of each month. And you will save a significant amount each year in more productive personal time usage. C

Children of the Future
A Natural Affinity.....
Children And Computers!!!

by Jane G. Reh

Photo by David McCuen

Eric Scott (at computer), Bill Reh and Heather McCuen, Pennsylvania school children who quickly became computer literate.

These fourth- and fifth-grade children took to their school's PET computer with great enthusiasm—and by the end of eight weeks were all computer literate.

Children have an instinctive and immediate rapport with computers. They approach them in the same way that they approach a new puppy—with a smattering of awe and a strong desire to touch! And touch leads to immense joy and excitement. Our children are eager for today's and even tomorrow's technology. They do not fear something new, because "new" has been the basis of each day of their young lives. It's a very special experience for an adult to be a part of a child's introduction to and growing experiences with computers. As adults—educators and parents alike—become less fearful and more comfortable with this new technology, they will find great satisfaction and mutual delight by entering this new and exciting future together with their children.

Let me try to show you how exciting this experience is by sharing with you a program that a Commodore PET, 100 fourth- and fifth-grade students and I participated in for four months this year.

As the school grapevine began to hum, students, teachers and parents asked, "Where is it? In the closet??" Indeed the PET had just arrived, was temporarily living in the office closet and decisions had to be made. Who would be the lucky ones to use

the COMPUTER? What exactly is computer literacy? How much could be taught to these eager young children about this high technology? These were not easy questions for the school district to answer.

The demonstration program started with 40 fourth- and fifth-grade students. These students watched two hours of films covering the manufacture, current uses and possible future uses of both microcomputers and mainframes, giving them a background knowledge of computers and a small technical vocabulary. The students were very patient because, you see, they knew that hands-on computer time was coming soon.

The children were paired and assigned one-half hour of computer time per week to learn BASIC (Beginner's All-purpose Symbolic Instruction Code) programming. Quite a challenge for these youngsters! With the instruction units ranging from the simple PRINT command to writing programs using loops and the decision-making IF-THEN statement, we all wondered, "How far will the children go?"

Finally the great day arrived—the first hands-on session. The children cleverly devised team methods so that not a precious second of computer time would be wasted. Each pair had decided before arriving who would be on the computer first. Then one child read the instructions aloud while the other typed at the console. At the end of 15 minutes they simply switched roles. This way, neither child lost time reading and typing at the same time. It was interesting to see two initial types of computer users. First, there were the aggressors, who would bounce into the computer chair and confidently begin to type. Then there were the wide-eyed, tempted, but unsure types, who would softly approach the computer, slide into the computer chair and gingerly touch the keys. They were assured that they couldn't hurt our PET and within two sessions all were self-assured computer users.

Since the children were able to complete the early, smaller units in less than 30 minutes, they had time to try both of our computer game programs, mainly the ever-popular "Guess The Number", and some of our CAI (Computer-Assisted Instruction) tapes.

In the "Guess The Number" game the computer picks a random number in a given range, which the player must guess. The computer helps by telling you if your guess is too high or too low. This game started with a range of zero through ten. This soon became too easy and the children quickly learned to change the correct program line (the RND statement) to increase the range, with zero through 500 becoming the most popular. Some might say that the children were just playing a game. Not so. This program was used with great intensity and a record kept of the number of guesses needed to get the correct answer. The children correctly reasoned that they should begin at the middle of the range and continue to pick subsequent mid-range numbers as the computer responded to each guess. They also knew how to LIST the program, find the program statement controlling the range and change it to suit themselves (EDIT the program). The development of logical reasoning and programming skills is definitely learning and this computer made it fun!

The CAI tapes had children who did not like math computation spending much time—sometimes including recess—using multiplication, factoring and fraction programs. Rather than grind it out with pencil and paper, which they normally tried to avoid, they would do the programs repeatedly until they could score 100. These were the same students who resented correcting their math papers in class!

At the end of eight weeks all students were computer literate, meaning they understood the basic workings of a computer—how a program can be written to solve a problem, be read by the computer and run to obtain the answer and how to operate the tape drive to use programs written by others—and they were loving it. Never before had a new curriculum been introduced that was not simply accepted or liked, but *loved,* by the children. They did their best to *never get sick on computer day!*

With such success another 60 students were added to the program, while those who could not yet be included prayed for next year to come quickly. Fortunately, computers never tire and our PET worked very hard indeed to accommodate as many teachers and students as the school day would allow. The computer had caught everyone's imagination and teachers now competed with the children for hands-on computer time. Everyone

wanted to experience what this magical machine could do for them.

Needless to say, the children who had a computer at home did have an advantage over the other children. They were able to use what they were learning in school on their home computer. They were also able to introduce their parents and other members of their family to BASIC programming.

And on the children went, using their computer time even on the very last day of school. How far did they advance? Except for the few who missed time because of serious illness, all the students from the original group were able to learn the first half of the BASIC programming language—quite an achievement!! Are they experienced super-programmers? No, of course not, nor was that our intent. Will they be able to use computers, even as computers change and new and ever more useful functions and peripherals are added, throughout their lives, without fear or hesitation? An emphatic YES!

What exactly did the children of the future accomplish? They honed their logical reasoning skills through learning a programming language, while becoming familiar and at ease with today's and tomorrow's technology. Their reasoning abilities were challenged automatically by the programming itself and by programs such as "Guess The Number". The computer made learning through CAI tapes more enjoyable—in fact, fun. It provided a much higher volume of practice in math, social studies and grammar than would have been possible, or endured, with pencil and paper.

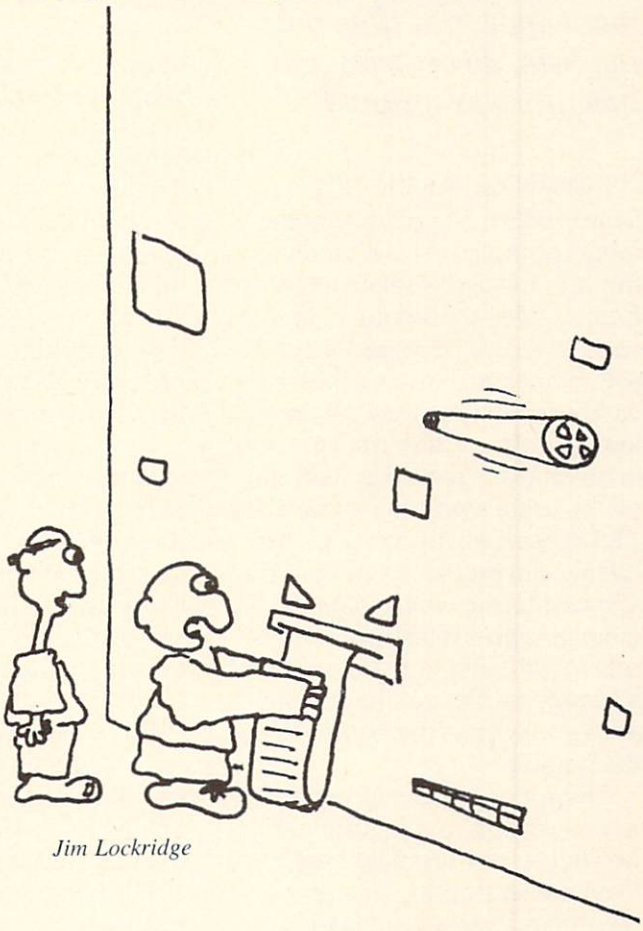
It is truly hard to believe, even for a computer professional, that one—just one—Commodore PET could accomplish so much—a remarkable beginning for these children. And best of all, if our children are computer literate and have mastered one programming language before high school, they will be able during their high school years to use their programming skills, married to the current math and science curriculum, to learn much more science and math and learn it in far greater depth than was ever possible before. And this is something that is sorely needed in our country today.

Needless to say, this one PET is rapidly multiplying, so that all of the children in our school district can gain the joy and rewards of computer literacy.

We all know that computer literacy will be a must for everyone as these machines continue to assist us in more and more ways in both our jobs and our everyday lives. Whatever career is chosen by your child, it will include computer usage in one form or another.

I hope that some day soon all parents and educators will join the exciting and rewarding computer revolution and embark on the computer journey with the children of the future. C

Mrs. Reh acted as a volunteer consultant to her Montgomery County, Pennsylvania, school district for six months this year, developing and implementing this computer curriculum.



Jim Lockridge

"It says 'Worship me or 2+2 will equal 5'"

Creating Relative Files on the 1541 Disk Drive

by Larry Greenley

Many readers have asked us about creating and using relative files on the 1541 disk drive, so we bribed one of our customer support representatives (with promises merely of fame—not fortune) into explaining the technique for creating data management-type files (like mailing lists, directories and inventory programs).

Using either the VIC 20 or Commodore 64 you can implement sophisticated file handling through the use of relative files. Relative files are unique in that a specific record or a specific field within a record can be accessed without searching through an entire file. Instead, the user can zero in on an exact record or field simply by using a position command. This is because the structure of a relative file provides a pointer that allows direct access to data. So even though a relative file stores one record after another just like a sequential file, you have faster access to what is there, thanks to the pointer.

The program below constructs, as an example, a relative file with two fields. The first field begins in character position one of the record and the second field begins in character position 25. The file is created by pressing the F1 key. Line 8 in the program actually creates the file. In line 8, Z$ acts as a file name variable for the file the user defines. The "L" tells the disk drive that the file to be constructed is a relative file and distinguishes it from a sequential or random file. The CHR$(50) in line 8 allocates space for records of 50 characters in length.

In line 9, OPEN 1,8,15 opens the command channel. Whenever relative files are used two channels must be opened: the command channel (exactly as it appears in line 9) and the channel that is reserved for the creation of the file (line 8). Logical file 1 is then reserved for use with the position command, where the number one must always follow PRINT# (as in line 30). The position command determines the channel, the number of the record and the position of the data in each record.

Understanding the position command is the next step in mastering relative files. Line 30 of the program is the first position command, which points to the first character position of the file. In line 30 the "P" tells the disk drive to point to the position in the record that corresponds to the fourth character string code (CHR$(1)).

The first character string code in line 30 contains the channel number, which is the third number in the relative file OPEN statement. The second and third character string codes are the low-byte high-byte format for the record number, where RECORD NUMBER=low-rec+(high-rec*256). The low-byte record number keeps track of records between one and 255 and the high-byte record number keeps track of records above 256. For example if record number 300 were to be accessed the position command would read:

```
PRINT#1,"P"CHR$(2)
CHR$(44)CHR$(1)CHR$(1)
```

In most cases you want each record to have more than one field. In this program, line 65 is a position command that creates a second field by telling the disk drive to position the record pointer to the twenty-fifth character position (CHR$(25)).

The position command only points to a specific character position. It does not write anything to or read anything from the file. In order to write to a relative file a PRINT# statement must be executed after the position command, as in line 60. Here the pointer to character position one was first specified in line 30. Line 60 then allows the user to write to disk. To read from a file that already exists on a disk first specify the position, as in line 130, then follow with an INPUT# statement, as in line 160.

In this program after the F1 key is pressed and a relative file has been constructed on a disk the user can type RUN and press the F3 key to read the newly constructed file. If an error condition

exists (flashing red light on the disk drive) the user can hit the RUN/STOP key, type RUN again and press the F5 key. This reads the error channel from the disk drive and displays the error number, the error message, the track and the sector of the error and then resets the disk drive again. Later on in the program the user is asked if the file should be printed. To print the file the user presses the F7 key when prompted or presses "C" to continue.

Here are a few tips for using relative files. 1) Before a file can be written to or read from, a position command must be executed. 2) The number immediately following the PRINT# statement in the position command must be a one (or the first number in the command channel OPEN statement, which in this case is OPEN 1,8,15). 3) The first character string code in the position command must always correspond to the third number in the relative file OPEN statement. 4) For programming convenience it is appropriate to use the FOR-NEXT loop variable in the second character string code of the position command (CHR$(I)), where "I" equals the record number.

In order to modify this program for personal needs additional position commands may be added to create more fields within each record. As the program now stands the position commands are pointing to character positions one and 25. To add fields simply include the character position numbers of your choice as the last character string codes in additional position command lines.

To increase the length of each record increase the value of CHR$(50) in line 8 to the desired record length. At present the program processes five records (the FOR-NEXT loop variable is one to five). But it can be modified to process over 700 records. To create up to 256 records, simply change the FOR-NEXT loop variable to "FOR I=1 TO 256". In order to create more than 256 records insert the number one into the third character string code of the position command. C

Relative File Program

```
1 REM ***RELATIVE FILE PROGRAM***
2 DIM A$(5),C$(5): PRINT"[CLEAR]"
3 PRINT "HIT[SPACE]F1[SPACE]TO[SPACE]CONSTRUCT[SPACE]A
  [SPACE]RELATIVE[SPACE]FILE"
4 PRINT "HIT[SPACE]F3[SPACE]TO[SPACE]READ[SPACE]A[SPACE]
  RELATIVE[SPACE]FILE"
5 PRINT "HIT[SPACE]F5[SPACE]TO[SPACE]READ[SPACE]THE[SPACE]
  ERROR[SPACE]CHANNEL": GOSUB 5000
6 PRINT "ENTER[SPACE]RELATIVE[SPACE]FILE[SPACE]NAME"
  :INPUT Z$
8 OPEN 2,8,2,Z$+",L,"+CHR$(50):REM CREATE THE RELATIVE FI
  LE
9 OPEN 1,8,15
10 GOSUB 1000
20 FOR I=1 TO 5
30 PRINT#1,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
   :REM POSITION THE RECORD POINTER
40 PRINT "ENTER[SPACE]A[SPACE]NAME"
50 INPUT A$(I)
```

```
60 PRINT#2,A$(I)
63 PRINT "ENTER[SPACE]ADDITIONAL[SPACE]INFO":INPUT C$(I)
65 PRINT#1,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(25)
   :REM POSITION POINTER TO 25TH CHAR.
67 PRINT#2,C$(I)
70 NEXT I
75 PRINT "DO[SPACE]YOU[SPACE]WISH[SPACE]TO[SPACE]REPLACE
   [SPACE]A[SPACE]RECORD":INPUT D$
76 IF D$="N"THEN 80
77 GOSUB 8000
78 GOTO 75
80 PRINT "THE[SPACE]RELATIVE[SPACE]FILE[SPACE]IS[SPACE]
   CONSTRUCTED"
82 FOR DE=1 TO 2500:NEXT DE:GOSUB 6000
85 CLOSE 1:CLOSE 2
90 END
100 PRINT"ENTER[SPACE]DESIRED[SPACE]FILE[SPACE]TO[SPACE]
    READ":INPUT Z$
105 OPEN 2,8,2,Z$:OPEN 1,8,15
106 PRINT "READING[SPACE]"Z$
110 FOR I=1 TO 5
115 REM FORI=5TO1 STEP-1
130 PRINT#1,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
160 INPUT#2,A$(I)
170 PRINT "RECORD#("I")=",A$(I)
175 K=6-I
177 PRINT#1,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(25)
179 INPUT#2,C$(I):PRINT "ADDITIONAL[SPACE]INFO:";C$(I)
180 NEXT I
181 PRINT "DO[SPACE]YOU[SPACE]WISH[SPACE]TO[SPACE]REPLACE
    [SPACE]A[SPACE]RECORD":INPUT D$
182 IF D$="N"THEN 185
183 GOSUB 8000
184 GOTO 181
185 GOSUB 1000
190 PRINT "END[SPACE]OF[SPACE]READ":FOR DE=1 TO 1500
    :NEXT DE:GOSUB 6000:CLOSE 1:CLOSE 2:END
1000 INPUT#1,A,B$,C,D:IF A<20 THEN RETURN
1001 IF A<>50 THEN PRINT A,B$,C,D::CLOSE 1:CLOSE 2:END
1999 RETURN
2000 OPEN 15,8,15
2001 INPUT#15,A,B$,C,D
2002 PRINT A,B$,C,D
2003 CLOSE 15:END
```

```
5000 GET S$:IF S$=""THEN 5000:REM SCAN KEYBOARD FOR FUNCT
     ION KEY CHR$ CODES
5001 IF S$=CHR$(133)THEN 6:REM ASSIGN F1 FUNCTION KEY
5002 IF S$=CHR$(134)THEN 100:REM ASSIGN F3 FUNCTION KEY
5003 IF S$=CHR$(135)THEN 2000:REM ASSIGN F5 FUNCTION KEY
5004 RETURN
6000 PRINT "[CLEAR]HIT[SPACE]F7[SPACE]FOR[SPACE]HARDCOPY
     [SPACE]OR[SPACE]C[SPACE]TO[SPACE2]CONTINUE"
6001 GET P$:IF P$<>CHR$(136)AND P$="C"THEN RETURN
6002 FOR DE=1 TO 500:NEXT DE:IF P$=""OR P$<>CHR$(136)THEN
      6000
6003 OPEN 4,4:CMD 4
6004 PRINT#4,"THE[SPACE]"Z$"[SPACE]FILE[SPACE]CONSISTS
     [SPACE]OF:"
6005 FOR I=1 TO 5
6010 PRINT#4,"RECORD[SPACE]#[SPACE]";I;"=";A$(I)
6012 PRINT#4,"ADDITIONAL[SPACE]INFO";I;"=":C$(I)
6015 CLOSE 4:RETURN
7000 REM TO READ RECORDS IN REVERSE ORDER,
      REMOVE THE REM IN LINE #115 AND
7002 PUT A REM BEFORE LINE #110
7005 REM TO READ THE 1ST FIELD OF THE 1ST RECORD AND THE
     2ND FIELD OF THE LAST
7007 REM REPLACE THE CHR$(I) IN LINE #177 WITH CHR$(K)
8000 PRINT "WHICH[SPACE]RECORD#[SPACE]DO[SPACE]YOU[SPACE]
     WANT[SPACE]REPLACED":INPUT I
8001 INPUT"ENTER[SPACE]NEW[SPACE]RECORD";A$(I)
8002 PRINT#1,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(1)
     :REM POSITION THE RECORD POINTER
8003 PRINT#2,A$(I)
8004 PRINT "ENTER[SPACE]NEW[SPACE]RECORD[SPACE](FIELD
     [SPACE]2)":INPUT C$(I)
8005 PRINT#1,"P"CHR$(2)CHR$(I)CHR$(0)CHR$(25)
     :REM POSITION THE RECORD POINTER
8007 PRINT#2,C$(I)
8009 PRINT "RECORD[SPACE]#";I;"HAS[SPACE]BEEN[SPACE]
     REPLACED"
8010 RETURN
```

Self-Modifying Programs for the Commodore 64 or VIC 20

by Dave Whomsley

A handy regenerator for getting a program to add to itself or delete from itself.

If you've ever asked the question, "How can I get a program to add to itself or delete from itself?" the program below will provide your answer. This program uses a little-known nook in the computer's memory known as the keyboard buffer queue. Any key can be put into this buffer for safekeeping, so a program can't touch it. After the program ends, the "hidden" keys come out of hiding and appear on the screen.

In this program we want the computer to think we typed something on the screen and then hit RETURN. This "something" can be a line number (which would be deleted when RETURN is hit) or a whole line of information (which would be entered when RETURN is hit). It can also be a command, such as GOTO50 or LIST.

To get the computer to do what we want it to do, we first print our information at the proper position on a blank screen. Next we "hide" some carriage returns in the buffer until we're ready to use them. Finally, we move the cursor to the home position and end the program.

Next the carriage returns come out of the buffer to accomplish the desired task. Eight come out on top of our line numbers and delete them from memory. One comes out on top of the variables and re-initializes them and one comes out on top of the GOTO line, which starts the whole procedure over.

The program, as is, will automatically delete any range of line numbers. It can also be easily changed to insert any number of REMark and DATA lines or input a DATA line then READ and use the data so you can instantly see (or hear) the results.

If you are thinking it might be easier to manually delete 30 lines or so rather than type in this long program, you can shorten the program by cutting out all the error-checking lines. The program works fine using just the 17 lines that are multiples of ten. But make sure you input your information properly!

Program Explanation

Line 10 first clears the screen then resets all the variables so you can start from scratch.

Lines 20-50 make up the input section. Here we input all the variables and check for errors.

Lines 60-100 print the lines we want deleted. This section would be changed if we wanted to add REMarks, DATA statements or whatever.

Lines 130-180 cause the program to "come back to life." Without these lines the program would print the proper line numbers, then stop. C

Regenerator

```
10 PRINT"[CLEAR]":RESTORE
11 REM *******   INPUT SECTION   *******
20 INPUT"STARTING[SPACE]LINE[SPACE]#";SL
21 IF SL<=300 THEN PRINT"[RVS]OVERWRITING[SPACE]ROUTINE
   [UP2]":GOTO 20
22 IF SL<>INT(SL)THEN PRINT"[UP2]":GOTO 20
25 PRINT"[DOWN2]"
30 INPUT"ENDING[SPACE]LINE[SPACE]#";EL
31 IF EL<=SL  THEN PRINT"[RVS]PLEASE[SPACE]INPUT[SPACE]
   AGAIN[UP2]":GOTO 30
```

```
32 IF EL<>INT(EL)THEN PRINT"[UP2]":GOTO 30
35 PRINT"[DOWN2]"
40 INPUT"INCREMENT";IN
41 IF IN<>INT(IN)THEN PRINT"[UP2]":GOTO 40
42 IF EL<1 THEN PRINT"[UP2]":GOTO 40
43 IF EL-SL<IN THEN PRINT"[UP2]":GOTO 40
45 INPUT"[RVS,DOWN2]OK";Q$:IF Q$<>"Y" THEN 20
46 REM ******  END INPUT SECTION  ******
50 PRINT"[CLEAR,DOWN3]"
60 FOR PL=1 TO 8
70 IF SL>EL THEN PRINT"LIST":GOTO 240
80 PRINT SL
90 SL=SL+IN
100 NEXT PL
110 PRINT"SL="SL":EL="EL":IN="IN
120 PRINT"[DOWN2]GOTO[SPACE]50
121 REM *******   POKE KEYBOARD   ******
130 FOR X= 1 TO PL+2
140 POKE 630+X,13
150 NEXT X
160 POKE 198,PL+2
170 PRINT"[HOME]"
180 END
```

Random Thoughts
Part 2: Building a Distribution

by Mark Zimmermann

Here are some more tools for you to use in dealing with random numbers in your computer programs. Specifically, this part will deal with how to build a particular distribution of random numbers. We'll get involved in other topics along the way, of course, and we won't come anywhere close to exhausting the subject, but we will make a good start. (Part 1 appeared in Issue 24.)

Distributions, Discrete and Continuous

As Part 1 described, random numbers generated by computer always have some sort of controlled distribution over some range. The distribution tells you how likely it is to get any particular number (or set of numbers) in each random draw. The range over which the random numbers are distributed tells you all the possible outcomes of a draw.

Thus, for example, rolling a die (or calling the BASIC function 1+INT(6*RND(1)) and getting the result back) produces a result from the range 1, 2, 3, 4, 5, 6, with all outcomes in that range equally likely (probability 1/6). At least, the outcomes are equally likely if the die is fair, you throw it hard enough and your RND function is good. This distribution of random numbers is "discrete", meaning the outcomes are separate, distinct values.

On the other hand, if you take a one meter-long piece of string and cut it at a random location (or call the BASIC function RND(1) itself), the length of the remaining string as a fraction of a meter is a "continuously" distributed random variable. It doesn't seem to come out of a finite set of separate possibilities, but rather could be any number between zero and one. Again, if you want to quibble, you could point out that there are a finite number of atoms in the string, and you can only measure the length to finite precision, so the set of possible outcomes really isn't infinite. Similarly, your computer's RND(1) result is only given to nine- or ten-decimal digits, so there are only ten billion or so possible outcomes of that function call. True—but for most practical purposes, that's close enough to infinity!

So it's very convenient to divide up the world of random numbers into "discrete" and "continuous" distributions and to treat each separately. If you're looking at a computer progam, it's usually easy to recognize a discrete distribution: somewhere in the function producing the random result, there's very likely an INT statement or an equivalent series of IF statements. The INT function in BASIC turns a continuous result into a discrete one by throwing away any fractional part and leaving you with an integer, discrete result. (Another digression: for negative numbers, a good INT function gives you a result one less than you might expect; it's "the greatest integer less than" function.)

Building a Discrete Random Distribution

It's straightforward, in most cases, to build any discrete distribution of random numbers that you need. The discrete case tends to be simpler than the continuous one, because there are fewer possibilities and you can frequently cover them all with a series of IF statements, if nothing else.

For example, suppose you want to decide what kind of monster to attack in a game you're designing; you may want to have a 10% chance for it to be a troll, a 17% chance for a demon, a 3% chance for a manticore and the rest of the time you'll settle for a simple dragon. The following program section will do the job nicely:

```
4630 X = RND(1)
4640 IF X < 0.1 THEN GOTO 9000:
   REM TROLL
4650 IF X < 0.27 THEN GOTO 9100:
   REM DEMON
4660 IF X < 0.3 THEN GOTO 9200:
   REM MANTICORE
4670 GOTO 9300: REM DRAGON
```

It may be obvious, but a couple of features of the above are worth pointing out if you haven't seen them before. First, we know that the result of line 4630 is to set X to some number between zero and one; more precisely, if the RND(1) function does its job, X will be greater than or equal to zero, but less than one and equally likely anywhere within that zone. So, 10% of the time X will lie between zero and .1 (troll), 17% of the time X will lie between .1 and .27 (demon), etc. The probabilities of the various outcomes all add up to one—just as the total piece of string was one meter long before we cut it. If the first IF statement (line 4640) fails, then X is not less than .1 and we don't have to waste time testing for that again in lines 4650 or 4660. On the other hand, if instead of GOTO statements following the IF test we had GOSUB statements, which return the program execution to the place that called the subroutine, then it would be crucial to write line 4650 as

```
4650 IF X< 0.27 AND X> 0.1 THEN GOSUB
     9100: REM DEMON
```

and so on... since otherwise a value of X which was less than .1 would also be less than .27 and you'd be generating the wrong monster.

A final point about the above simple example. You really don't need to worry much about whether or not the test in the IF statement is "less than" or "less than or equal to", since there's only one chance in a billion or so that it will make any difference!

So to make the rule more explicit. To generate an arbitrary discrete distribution, start by listing the possible outcomes you want to see, each with its own probability. The probabilities should be fractions between zero (it never happens) and one (it always happens). Percentages are converted to fractions by dividing by 100. Then generate a random number between zero and one using RND(1), and test it to see whether it's less than the chosen probability value for your first event. If it is less, make the first choice; if not, test it to see if it falls between the first event's probability and the sum of the first and second events' probabilities. (This zone is the width of the second event's probability.) Repeat the procedure for each event.

This procedure will always work, but sometimes it may become tedious, especially if there are many possible outcomes which fall in some regular pattern. In that case, you should look for a mathematical rule to take the place of all those IF statements. That's where the INT function usually shows up. If a large number of outcomes are equally likely (for example, you want to choose with equal probabilities from monster types 1 through N) you can generate a random integer from the set 1, 2, ..., N by stretching the line interval between zero and one by a factor of N, so that it is N units long, and then chopping it up into N pieces using the INT function. Thus:

```
5550 I = INT(N*RND(1))+1
```

gives you an integer between one and N inclusive. It simply takes the result of RND(1), stretches it by N, uses INT to turn it into integers from the set 0, 1, 2, ..., N−1 and then adds one to give the final value for I from among the digits one through N. You can then take advantage of this value of I in a BASIC statement of the form:

```
5552 ON I GOTO 9500, 9520, 9540, etc.
```

where the first choice is the one taken when I=1, the second choice for I=2, etc.

If all your choices aren't equally likely, you can try to play tricks with the "stretching" part of the above operation. For example, think about the following:

```
2340 X = RND(1)
2345 I  = INT(4*X*X)+1
```

What are the possible outcomes? Well, since X lies between zero and one, so does X*X, and so 4*X*X must lie between zero and four (actually, between zero and 3.99999...). So, as in the earlier example line 5550, with N=4, the result of line 2345 will be to set I to an integer from the set 1, 2, 3, 4.

So is the result of line 2345 the same as the result of 5550 (when N=4)? Not on your life! The set of possible outcomes (the range of the random variable I) is the same in both cases, but line 2345 does

not give all four results with equal likelihood! The trick, of course, is in the multiplication X*X. When X is small, X*X is *much* smaller (for example, if X=.1, then X*X is .01, one tenth as large). But if X is near one, then X*X isn't much different from X. The stretching of our string is no longer uniform. The string is more like a rubber band, which has gotten compressed at the end near zero and stretched thinner at the end near one.

If you work it out, you'll find that all values of X between zero and .4999999... give a result of I=1. Thus, I=1 half the time! For X between .5 and .7071... (which is SQR(2)/2, by the way), I comes out to be 2; so I=2 happens about 20.7% of the time. For X between .7071 and .866 (approximately), I=3, and for X between about .866 and one, I=4. You could make up a table:

I	Probability (approx.)
1	.50
2	.21
3	.16
4	.13

So, after the funny stretching that X*X caused in line 2345, the low end of the string ended up with more than its usual share of fiber and the high end got thinned out considerably.

How do you design such a "stretch" for your own particular application? One way is to experiment and look at the outcomes, trying one kind of stretch and then another until you get what you want. A useful and very general form is:

```
6780 X=RND(1)
6790 I=INT(N*X↑K)+1
```

This gives one all sorts of stretches for various values of K. When K=1, it's just like example line 5550 and all results for I in the range one through N are equally likely. When K=2, there's a stretch like the one in example lines 2340-2345, which biases things toward the low end. If K is less than one (but greater than zero) things go the other way, toward the higher numbers. Trying K=0 gives extreme bias toward the high end—I is *always* equal to N then!

One final suggestion: When generating numbers randomly for a physical simulation of some sort, let yourself be guided by the actual physical system you're worrying about. It will frequently suggest some particular distribution or way of generating a distribution. Rolling a six-sided die and a 20-sided die and adding the results, for example, gives one an outcome in the range two through 26, with not all of those values equally likely. To simulate it on the computer, the easiest approach is to simulate each die separately, generate equally likely numbers in the ranges of one through six and one through 20 separately and then add them.

Building a Continuous Random Distribution

We're almost out of space for this month, so we'll have to continue the discussion in later columns, but as promised before, here's how to build a continuous distribution: don't use an INT statement!

That's not a joke, either. The essential concept of building a totally controlled discrete random distribution was to stretch a string and then chop it into pieces. For a continuous set of outcomes, you could think of the pieces as being arbitrarily tiny; or better yet, don't cut the string into discrete pieces at all. The stretching you do can be linear, uniform, such as the result of multiplying RND(1) by some constant and adding another constant. Thus, if you have to generate a number that is equally likely to lie anywhere between 17 and 21 (an interval of length four), just use 4*RND(1)+17. If you want to stretch with some bias toward one end or the other, you can use something like N*X↑K+J to get numbers between J and J+N, where X=RND(1).

Other sorts of stretches may be better for particular purposes. Next time, I'll discuss some standard distributions for random variables, such as the famous Gaussian bell-shaped curve, and the sorts of stretches needed to get them. Sometimes really strange stretches are desirable, such as the ones that logarithm or tangent functions provide. They can produce random numbers that are arbitrarily large by stretching the interval zero to one an infinite amount at one end. We'll learn how to design such stretches, when appropriate. C

PETSpeed Review

by Joe Rotello

This column will sometimes be devoted to a short but concise review/user test of PETSpeed compiled programs that are commercially sold. This issue we consider the Typro Data Manager and Word Processor from Input Systems, Inc. (25101 S.W. 194th Ave., Homestead, FL 33031, 305-245-3141).

These programs may be used either alone or integrated together. They are both inexpensive alternatives to the much costlier data base/word processors currently available for the Commodore product line. We tested the version designed for the CBM 8000 series computers.

Data Manager

Users format their own fields within the record. Maximum records are limited only by the capacity of the disk; data files are in relative file format. User input is very friendly, with comments, reverse fields and the like inserted in order to keep confusion to a minimum. One unique feature is that for any changes in data to become permanent, *two* shifted returns are required. I like that feature since it keeps mistakes to a minimum.

The SORT and SEARCH functions are well thought out. The user can search each field and search by pattern matching if desired. I especially like the SEARCH feature that allows you to selectively print records or just certain fields in each record.

Address insertion from the Word Processor program can be accomplished into the Data Manager with a minimum of hassle. All in all, a well thought out product with an informative, sometimes dry, but basically well written manual.

Word Processor

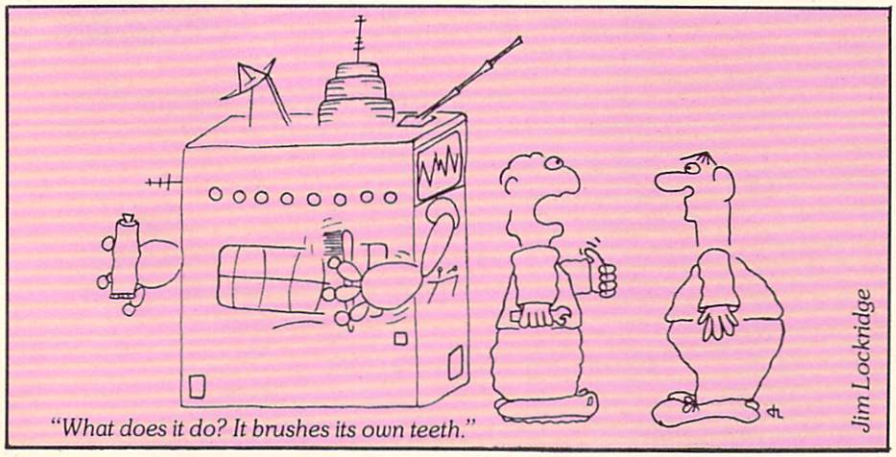
The Typro Word Processor is a very nifty bit of programming. Not quite as easy to learn as the Data Manager but very powerful for the price. And when was the last time *you* saw your dot matrix printer *underline!* Yep, it's in this one.

Features like screen line editing, line insert/delete, justification and global edit (moving text from one place to another) are included and they are simple to use and understand. I would judge that the user can enter about 250 or so lines of 45-55 characters each in the memory space provided for the 4032/8032 computers.

Other features such as page numbering (top or bottom) and headings are available and add much to the effectiveness of Typro. Best of all, since the program is compiled under (you guessed it) PETSpeed, the program is fast enough even for the best typist.

The most powerful feature I found was the ability to append files together. So if you have a big manuscript or the gift of gab you can chain up to the maximum disk space available and still have it all print out as if it were one big happy file.

The price is only $89.00 for *both* programs or $55.00 each. Even though they are not on a par with Wordcraft or Silicon Office they are very powerful and useful for the money spent. The factory people are friendly and their service is very good. Remember, the mark of a good company is how they react when you have a problem or just want to ask a question. C



"What does it do? It brushes its own teeth."

Public-Key Cryptography for Commodore Microcomputers

by Craig R. Hessel

This article presents a package of six cryptographic subroutines written in 6502 machine language and a calling program for the package written in PET/CBM BASIC. The subroutine package is relocatable and should work on any 6502 or 6510 microcomputer. The calling program, which should be adaptable to VIC or Commodore 64 computers, drives a cassette-file-based public-key cryptographic system. The program and 6502 package were developed and tested on an original ROM 8K PET.

Cryptography is the science of safeguarding information by the use of secret codes. The computer age has spurred an explosion of research in this field for several reasons. First of all, fast computers can easily break older and simpler cryptographic schemes. At the same time, this speed has broadened the avenues of research available to cryptographers. And finally, computers today have become storehouses for vast amounts of vulnerable, confidential data.

A focal point of this research has been the novel concept of public-key cryptography. In a typical cryptographic system, two or more people share a single key—usually a number or password—which both locks and unlocks data. Such a system has drawbacks. Before coded communication can begin, the key must be distributed secretly. This requires face-to-face meetings or the services of reliable couriers. In addition, the risk of key theft increases as more people share the same key. In a public-key cryptosystem, each person has two keys—a secret decoding key and a public encoding key. Everyone's *encoding* keys are shared openly, for example, in a public directory. However, each person's *decoding* key is known only to themselves. By using the directory and a publically known encoding method, individuals may exchange confidential messages without the drawbacks just mentioned. In essence, public-key cryptography permits private communication over public channels without requiring secret preliminary exchanges of keys or coding methods.

In any public-key cryptosystem, there is a known relationship between the encoding key and the decoding key. For the system to be secure, however, this relationship must be unusual enough so that it is practically impossible to deduce the decoding key from the encoding key. Reliable public-key systems are very rare for this reason.

In 1978, three MIT researchers—R. L. Rivest, A. Shamir, and L. Adleman—published a landmark paper announcing their discovery of the first public-key cryptosystem. In the original RSA system, each encoding key includes the product of two large prime numbers. The corresponding decoding key involves the two primes themselves. Prime numbers (2, 3, 5, 7, 11, and so on) are numbers which cannot be written as the product of smaller numbers. Although it is easy to see, for example, that 247 is the product of primes 13 and 19, it turns out that even with the aid of a high speed computer it is virtually impossible to factor a large product—roughly 100 digits or more—into its two prime components. As a bonus feature, the RSA system allows private messages to be "signed" digitally to prevent forgery.

In the program here, a modified version of the original RSA cryptosystem is combined for speed with a typical cryptographic system. The slower RSA system is used in the "signed" encoding/decoding of a disposable key at the start of every message. The body of the message is then encoded/decoded more quickly with this one-time-use key.

Program Explanation

As your first task, type in and save a copy of CRYPTOCODE 5/83. You will need a machine

language monitor and the subroutine package hex listing which accompanies this article. The package occupies $16EC-$1DBC, with $1DBD-$1FFF reserved for storage.

Next, type in and save a copy of the BASIC calling program listed here. These extra program comments may be helpful.

Line 145 The two POKEs change the top-of-memory pointer and CLR adjusts other pointers to protect CRYPTOCODE 5/83 from BASIC. For later ROMs, replace pointer 134-135 with 52-53. S is the start of the 6502 package and PF is the printer flag. If you set PF=1 to get hard copy, then turn on the printer before running the program.

Lines 150-155 B is the buffer pointer and E is the SYS entry point. The single byte strings are cursor left, carriage return, cursor character, clear screen, reverse and reverse off.

Lines 180-190 The first test checks that CRYPTOCODE 5/83 is properly in place. The next test checks for enough storage area above the package. A storage error would probably be due to an improper relocation.

Line 410 PRINT#5,B$; also works here but does not restrict hard copy to 40 columns.

Line 490 The OPEN is for cassette #1, with T=0 for read and T=1 for write.

Line 510 Status value of 64 flags normal end-of-file.

Line 530 The cassette print is followed by a software fix to lengthen the inter-record gap. Eliminate the POKEs and the delay loop for later ROMs.

Line 550 The POKE here indexes the appropriate 6502 subroutine: GETKEY, RSAENC, RSADEC, GETSEED, SEEDENC, or SEEDDEC. The SYS does the calling.

Lines 575 and 595 In these two lines, Z=0 flags buffer #1 and Z=40 flags buffer #2 of the subroutine package.

Lines 650-700 The keyboard input routine has three input parameters—maximum number of characters N, type of character flag TF and return flag RF.

Line 670 The DEL key is allowed as a rub-out key for editing.

Line 690 The quote character is printed an extra time and then erased to take the PET/CBM out of quote mode.

Testing the Program

To begin testing, turn your machine off and back on again, load your copy of CRYPTOCODE 5/83 next (the loading order is important), then load and run your copy of the calling program. If the message CHECKSUM ERROR appears, then you must correct an error in your copy of the subroutine package. Otherwise, you will see a menu list of commands.

Test the MENU, EXIT, WRITE FILE, and READ FILE commands next. The WRITE FILE command allows you to use any of 64 unshifted ASCII keyboard characters in writing to a file in blocks of 80 characters. Each block is automatically sent to file after the eightieth character is entered. The block may be edited earlier with the DEL key. The RETURN key is recognized only at the beginning of a block, when a null string is used to signal end of data. This command may be used, for example, to write ordinary introductions to coded messages, to add keys to the key cassette, or to transfer coded message printouts to cassette prior to decoding. The READ FILE command reads and displays any files produced by the other commands. It may be used, for example, to read the uncoded introduction to a coded message, to advance the key cassette past old key files prior to inserting a new key or to dump a coded message.

The remaining three commands call the cryptographic subroutine package. During such calls interrupt requests are ignored, so you can stop the program only by turning off your machine. To test these commands, refer to the SAMPLE OUTPUT that accompanies this article. The GENERATE KEYS command takes about 31 minutes with "random" hex input of all 0's and about 74 minutes with all F's. You should at least test this command with all 0's input. This will put D-KEY #1 and E-KEY #1 on your key cassette. Then either test the command

```
                SAMPLE OUTPUT

*** 6502 PUBLIC-KEY CRYPTOGRAPHY ***

WAIT ABOUT TEN SECONDS

0=MENU
1=GENERATE KEYS
2=ENCODE MESSAGE
3=DECODE MESSAGE
4=WRITE FILE
5=READ FILE
6=EXIT

COMMAND? (0-6) 1=GENERATE KEYS
ENTER RANDOM HEX STRING:
00000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000
WAIT ABOUT ONE HOUR
INSERT KEY CASSETTE FOR WRITE
PRIVATE D-KEY:
000000000000A400000000004800000000000B88000
000000000000580000000000036000000000000252
ENTER FILE NAME FOR PRIVATE D-KEY:
D-KEY #1
PUBLIC E-KEY:
00000000000290B00000001202 2F0000002E3D75
0A500000275 9BE6D000013817 9A00000000D684F
ENTER FILE NAME FOR PUBLIC E-KEY:
E-KEY #1
```

again with all F's input, or else, as a short cut, use the WRITE FILE command twice to add D-KEY #2 and E-KEY #2 to your key cassette.

Now test the ENCODE MESSAGE command. It is a good idea to keep key files and message files on separate cassettes, even during testing, since this will reduce the chance of accidentally overwriting valuable files. The same ASCII characters are allowed here as in the WRITE FILE command, but in 40- rather than 80-character blocks. Note that each line of 40 message characters is encoded as 80 hex characters and that an extra 80 hex characters—the encoded disposable key—precedes the coded message.

As a final test, call the DECODE MESSAGE command to insure that the original message is recovered.

Using the Keys

You are now ready to step into the realm of public-key cryptography. First create your personal encoding and decoding keys. Make sure your key cassette is positioned past the four sample keys already there, and then call the GENERATE KEYS command for the last time. For this call, enter truly random hex input.

Once you have your key pair, you will need to swap encoding keys with other PET/CBM owners or with VIC and 64 owners who have adapted the calling program to their machines. One way to do this is to have encoding keys posted on computer club bulletin boards or in club newsletters. Add to your key cassette the encoding keys of individuals you

would like to correspond with. One side of a C-10 cassette holds 16 key files, so you should have room yet for ten such encoding keys. Even though these encoding keys are not secret, your decoding key is, so remember to keep your key cassette secure from prying eyes.

After encoding a message, you may transmit the message file to its destination in any convenient way. If you own a modem, you may send the file over phone lines. If you have a printer, you may dump the file and send the printout by mail. Or you may just send the message cassette itself.

Before decoding a message, you must have the coded message on file. If the message came as a printout, use the WRITE FILE command to transfer the hex data to cassette. Be especially careful when entering the first 80 hex characters, since a mistake here will garble the entire message. A later typo will garble a block of ten letters. One side of a C-10 cassette will hold a file of about 9K hex characters, which is normally equivalent to 700-800 English words.

How the Keys Work

This section covers the 6502 package and its encoding/decoding methods. If you do not care for technical details, then skip ahead to the last paragraph. Before continuing here, see the accompanying memory map and I/O specifications.

As stated before, CRYPTOCODE 5/83 is a package of six cryptographic subroutines for 6502/6510 microcomputers. The package is relocatable, with any relative shift of the program area applying also to the storage area. In particular, the entry point and the location of the I/O buffers are changed accordingly. The package is called by the instruction JSR ENTRY, with the contents of the Y-register indexing the desired subroutine. An alternate SYS entry point at ENTRY-2 is used by the BASIC calling program, with the contents of POKE location ENTRY-1 as the index. Interrupts are inhibited during calls to the package, which itself makes no external calls and contains no external references. Use is made of the 6502 stack to a depth of 41 bytes. The top 5/8 of page zero is also used, but the original contents there are restored. Register contents are not preserved.

The three subroutines GETSEED, SEEDENC, and SEEDDEC comprise a typical cryptographic system. GETSEED generates a 40-byte SEED, which is the disposable key used by the calling program for encoding/decoding one message. The SEED has four parts: a prime modulus P, a decoding multiplier D, an encoding multiplier E and an initial pseudo-random number R. Each of these is ten bytes long, except for P, which has an implied extra bit. The subroutine generates P, D, and R randomly. E is computed so that 1 = E*D mod P. This means that 1 is the remainder when E*D is divided by P. For example, 1 = 8*15 mod 17.

SEEDENC and SEEDDEC operate on text data ten bytes at a time in a two-step process. At the start of encoding and at the end of decoding, each text block is exclusive-ored with the current R value. This insures that identical message blocks will be encoded differently. For each new block, R is set equal to the low order ten bytes of (R+1)*(4*R+1). In the main encoding/decoding procedure, each ten-byte text block T is set equal to T*E mod P for encoding or T*D mod P for decoding. This transformation is repeated until T fits into ten bytes. Generally, there are no repetitions, since the GETSEED subroutine makes sure that prime P just exceeds ten bytes in length.

The GETKEY, RSAENC, and RSADEC subroutines make up the public-key cryptosystem. GETKEY generates a 40-byte E-KEY and a 40-byte D-KEY. The E-KEY, when preceded by an extra set bit, forms the 97-digit product N of primes P and Q. The D-KEY contains P and Q themselves in a packed format. The subroutine chooses P and Q randomly while assuring that neither P-1 nor Q-1 is a multiple of 257.

The RSAENC subroutine encodes a 40-byte block T by the rule T = T ↑ E mod N where the encoding power E is 257. This transformation is repeated until T fits into 40 bytes. GETKEY insures that N is only slightly larger than 256 ↑ 40, so only one transformation is usually needed.
RSADEC unpacks P and Q from the D-KEY, sets

N = P*Q and X = (P−1)*(Q−1), and then computes decoding power D so that 1 = E*D mod X. Finally, the 40-byte text block T is decoded by the rule T = T ↑ D mod N. As above, this step is repeated until T fits into 40 bytes, although repeats are seldom necessary.

Both GETKEY and GETSEED use the same probabilistic method to find large primes. The algorithm cannot assure valid output. However, its chances here of mistaking a composite number for a prime are less than one in a billion.

With the RSA cryptosystem, data blocks may be decoded before being encoded. This is the basis for the digital "signing" of messages in the system. To both "sign" and encode a message, the sender first decodes the message with his own secret decoding key and then encodes the result with the receiver's public encoding key. To decode and "unsign" this data, the receiver first decodes the data with his decoding key and then encodes the result with the sender's encoding key. The calling program uses this method in the "signed" encoding/decoding of the initial SEED at the start of each message.

For More Information

As a final note, it should be pointed out that the RSA cryptosystem is part of a controversy between the academic community and the National Security Agency. At issue are the traditional notions of academic freedom, private industry's need for confidentiality of computer data, and the government's desire to keep unbreakable ciphers out of enemy hands. For a discussion of the controversy, read the article "The Crypto-Censors" by Paul Hoffman in the July, 1982, issue of *Science Digest* or see the episode "Privacy" from the *Nova* television series on public television. For more technical information on the RSA system, refer to the original Rivest-Shamir-Adleman article "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" in the February, 1978, issue of *Communications of the ACM* or see chapter 4.5.4 of the second edition of Donald Knuth's *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* C

Memory Map

Program area:	$16EC-$1DBC	Storage area:	$1DBD-$1FFF
Parameters:	$16EC-$17EB	6502 storage:	$1DBD-$1E3C
Interpreter:	$17EC-$1B19	LIAL storage:	$1E3D-$1FFF
LIAL code:	$1B1A-$1DBC	Buffer #1:	$1E3D-$1E64
ENTRY:	$17EC	Buffer #2:	$1E65-$1E8C

(LIAL = Large Integer Arithmetic Language)

I/O Specifications

Subrtn	********* Input *********			****** Output ******		Approx
name	Y-reg	Buffer #1	Buffer #2	Buffer #1	Buffer #2	time
GETKEY	1	RANDOM	—	E-KEY	D-KEY	1 hr
RSAENC	2	PLAINTEXT	E-KEY	CIPHERTEXT	E-KEY	10 sec
RSADEC	3	CIPHERTEXT	D-KEY	PLAINTEXT	D-KEY	6 min
GETSEED	4	RANDOM	—	SEED	—	2 min
SEEDENC	5	PLAINTEXT	SEED	CIPHERTEXT	NEXTSEED	0.4 sec
SEEDDEC	6	CIPHERTEXT	SEED	PLAINTEXT	NEXTSEED	0.4 sec

Driver Program

```
100 REM ************************************************
105 REM THIS CASSETTE-BASED PUBLIC-KEY CRYPTOGRAPHIC PROG
    RAM FOR OLD ROM PET
110 REM CALLS THE 6502 SUBROUTINE PACKAGE 'CRYPTOCODE 5/8
    3', WHICH IS ASSUMED
115 REM TO BE IN PLACE (5868-7612).  WITH MORE THAN 8K RA
    M, THE PACKAGE MAY BE
120 REM RELOCATED HIGHER, BUT THEN RESET POINTER IN LINE
    145.  FOR HARD COPY
125 REM OUTPUT, SET PF=1 IN SAME LINE.  FOR LATER ROMS,
     FIX LINES 145 & 530.
130 REM ************************************************
135 REM
140 REM INITIALIZE
145 POKE 134,236:POKE 135,22:CLR:S=PEEK(134)+256*PEEK(135)
    :PF=0:IF PF THEN OPEN 5,4
150 B=S+1872:E=S+254:CL$=CHR$(157):CR$=CHR$(13)
155 CC$=CHR$(166):CS$=CHR$(147):RV$=CHR$(18):RF$=CHR$(146)
160 M$(0)="MENU":M$(1)="GENERATE[SPACE]KEYS"
    :M$(2)="ENCODE[SPACE]MESSAGE"
165 M$(3)="DECODE[SPACE]MESSAGE":M$(4)="WRITE[SPACE]FILE"
    :M$(5)="READ[SPACE]FILE"
170 M$(6)="EXIT":M$(7)="READ":M$(8)="WRITE"
    :M$(9)="OF[SPACE]":M$(10)="FOR[SPACE]"
175 PRINT CS$;:B$="***[SPACE]6502[SPACE]PUBLIC-KEY[SPACE]
    CRYPTOGRAPHY[SPACE]***":GOSUB 385:GOSUB 390
180 B$="TEN[SPACE]SECONDS":GOSUB 425:X=-PEEK(E+1)
    :FOR I=S TO S+1744:X=X+PEEK(I):NEXT
185 I=S+2323:POKE I,0:Y=PEEK(I):IF X<>179140 THEN B$="CHE
    CKSUM[SPACE]ERROR":GOTO 215
190 POKE I,255:T=0:B$="":IF Y+PEEK(I)<>255 THEN B$="STORA
    GE-IN-ROM[SPACE]ERROR":GOTO 215
195 REM
200 REM COMMAND LOOP
205 GOSUB 385:ON T+1 GOSUB 230,245,270,310,350,350
    :GOSUB 390:B$="COMMAND?[SPACE](0-6)[SPACE]"
210 GOSUB 400:N=1:TF=2:RF=0:GOSUB 650:T=VAL(B$)
    :B$="="+M$(T):IF T<6 THEN 205
215 GOSUB 385:CLOSE 5:PF=0:END
220 REM
225 REM MENU
230 FOR I=0 TO 6:B$=CHR$(48+I)+"="+M$(I):GOSUB 385:NEXT
    :RETURN
```

```
235 REM
240 REM GET KEYS
245 GOSUB 565:B$="ONE[SPACE]HOUR":GOSUB 425:Y=1:GOSUB 550
    :Z$="KEY":T=1:GOSUB 440
250 Z$="PRIVATE[SPACE]D-KEY":Z=40:GOSUB 255
    :Z$="PUBLIC[SPACE]E-KEY":Z=0
255 GOSUB 435:GOSUB 595:GOSUB 620:GOSUB 400:T=1:GOSUB 485
    :B$=T$:GOSUB 530:GOTO 370
260 REM
265 REM ENCODE
270 X$="SEND":Y$="RECEIV":GOSUB 455:IF B$=""THEN RETURN
275 GOSUB 565:B$="EIGHT[SPACE]MINUTES":GOSUB 425:Y=4
    :GOSUB 550:GOSUB 595:T$=B$:GOSUB 545
280 B$=T$:GOSUB 575:Z=0:GOSUB 595:GOSUB 620:Z$="MESSAGE"
    :T=1:GOSUB 440
285 GOSUB 485:B$=T$:GOSUB 530:GOSUB 430
290 N=40:TF=0:RF=1:GOSUB 650:IF B$=""THEN 370
295 Y=5:GOSUB 590:GOSUB 620:GOSUB 530:GOTO 290
300 REM
305 REM DECODE
310 X$="RECEIV":Y$="SEND":GOSUB 455:IF B$=""THEN RETURN
315 Z$="MESSAGE":T=0:GOSUB 440:GOSUB 485:GOSUB 505
    :IF B$=""THEN 370
320 GOSUB 610:Z=0:GOSUB 575:B$="SIX[SPACE]MINUTES"
    :GOSUB 425
325 GOSUB 545:Z=0:GOSUB 595:Z=40:GOSUB 575:GOSUB 435
330 GOSUB 505:IF B$=""THEN 370
335 GOSUB 610:Y=6:GOSUB 590:GOSUB 630:GOSUB 400:GOTO 330
340 REM
345 REM WRITE/READ FILE
350 T=5-T:Z$="DATA":GOSUB 440:GOSUB 485
    :ON T+1 GOSUB 435,430:IF T THEN 365
355 GOSUB 505:IF B$=""THEN 370
360 GOSUB 630:GOSUB 400:GOTO 355
365 N=80:TF=0:RF=1:GOSUB 650:IF B$<>""THEN GOSUB 530
    :GOTO 365
370 CLOSE 9:RETURN
375 REM
380 REM MULTIPLE ENTRY PRINT SEQUENCE
385 GOSUB 400
390 PRINT:IF PF THEN PRINT#5,CR$;
395 RETURN
```

```
400 PRINT B$;
405 L=LEN(B$):IF PF=0 THEN RETURN
410 PRINT#5,LEFT$(B$,40)CHR$(-13*(L>39))MID$(B$,
    41)CHR$(-13*(L>79));:RETURN
415 REM
420 REM PROMPTS
425 B$="WAIT[SPACE]ABOUT[SPACE]"+B$:GOTO 385
430 B$="ENTER[SPACE]"+Z$+":[SPACE](END[SPACE]WITH[SPACE]
    NULL[SPACE]STRING)":GOTO 385
435 B$=Z$+":":GOTO 385
440 B$="INSERT[SPACE]"+Z$+"[SPACE]CASSETTE[SPACE]FOR
    [SPACE]"+M$(7+T):GOTO 385
445 REM
450 REM READ KEYS
455 Z$="KEY":T=0:GOSUB 440:Z$=X$+"ER'S[SPACE]D-KEY"
    :GOSUB 470:IF B$=""THEN RETURN
460 GOSUB 610:X$=B$:Z$=Y$+"ER'S[SPACE]E-KEY":GOSUB 470
    :IF B$=""THEN RETURN
465 GOSUB 610:Y$=B$:RETURN
470 GOSUB 485:GOSUB 505:GOTO 370
475 REM
480 REM OPEN TAPE FILE
485 B$="ENTER[SPACE]FILE[SPACE]NAME[SPACE]"+M$(9+T)+Z$+":"
    :GOSUB 385
490 N=16:TF=0:RF=2:GOSUB 650:GOSUB 390:OPEN 9,1,T,B$
    :RETURN
495 REM
500 REM TAPE READ
505 B$="":FOR I=1 TO 80:GET#9,A$:IF ST=0 THEN B$=B$+A$
    :NEXT:RETURN
510 I=80:IF ST=64 AND B$=""THEN NEXT:RETURN
515 B$="TAPE[SPACE]READ[SPACE]ERROR":GOSUB 385:B$="":NEXT
    :RETURN
520 REM
525 REM TAPE WRITE
530 PRINT#9,B$;:POKE 59411,53:FOR I=0 TO 99:NEXT
    :POKE 59411,61:RETURN
535 REM
540 REM SIGN/UNSIGN & 6502 CALL
545 B$=X$:Z=40:GOSUB 575:Y=3:GOSUB 550:B$=Y$:GOSUB 575:Y=2
550 POKE E+1,Y:SYS E:RETURN
555 REM
560 REM RANDOM INPUT & BUFFER POKE
```

```
565 B$="ENTER[SPACE]RANDOM[SPACE]HEX[SPACE]STRING:"
    :GOSUB 385:N=80:TF=1:RF=0
570 GOSUB 650:GOSUB 610:Z=0
575 FOR I=1 TO 40:POKE Z+B+I,ASC(MID$(B$,I)):NEXT:RETURN
580 REM
585 REM POKE-CALL-PEEK & BUFFER PEEK
590 Z=0:GOSUB 575:GOSUB 550
595 B$="":FOR I=Z+B+1 TO Z+B+40:B$=B$+CHR$(PEEK(I)):NEXT
    :RETURN
600 REM
605 REM HEX TO ASCII & ASCII TO HEX & INSURE 6-BIT ASCII
610 T$="":FOR I=1 TO 79 STEP 2:X=ASC(MID$(B$,I))-48
    :Y=ASC(MID$(B$,I+1))-48
615 T$=T$+CHR$(255 AND 16*(X+7*(X>9))+Y+7*(Y>9)):NEXT
    :GOTO 635
620 T$="":FOR I=1 TO 40:X=ASC(MID$(B$,I)):Y=X AND 15
    :X=(X AND 240)/16
625 T$=T$+CHR$(X+48-7*(X>9))+CHR$(Y+48-7*(Y>9)):NEXT
    :GOTO 635
630 T$="":FOR I=1 TO LEN(B$):T$=T$+CHR$((63 AND ASC(MID$(
    B$,I))-32)+32):NEXT
635 B$=T$:RETURN
640 REM
645 REM KEYBOARD INPUT
650 PRINT RV$;:Y=N:GOSUB 705:B$="[SPACE]"
655 GET A$:IF A$<>""THEN 655
660 PRINT CC$CL$;:Y=LEN(B$):PRINT"[SPACE]"CL$;:GET A$
    :IF A$=""THEN 660
665 X=ASC(A$):IF X=13 THEN IF RF=2 OR RF=1 AND Y=1 THEN 7
    00
670 IF X=20 THEN IF Y>1 THEN PRINT CL$"[SPACE]"CL$;
    :B$=LEFT$(B$,Y-1):GOTO 660
675 IF TF=0 THEN IF X<32 OR X>95 THEN 660
680 IF TF=1 THEN IF X<48 OR X>70 OR X<65 AND X>57 THEN 660
685 IF TF=2 THEN IF X<48 OR X>54 THEN 660
690 PRINT A$;:IF X=34 THEN PRINT A$CL$RF$"[SPACE]"RV$CL$;
695 B$=B$+A$:IF Y<N THEN 660
700 B$=MID$(B$,2):PRINT RF$;:Y=N-LEN(B$)
    :ON(Y=0)+1 GOSUB 705:GOTO 405
705 FOR I=1 TO Y:PRINT"[SPACE]";:NEXT:FOR I=1 TO Y
    :PRINT CL$;:NEXT:RETURN
```

Cryptocode 5/83

```
$16EC  07 0D 14 28 FF FF 00 0A
$16F4  32 00 10 00 05 10 01 01
$16FC  00 00 00 00 00 00 00 00
$1704  00 00 00 00 00 02 02 FF
$170C  01 01 01 01 01 02 02 02
$1714  02 02 FF FF FF FF FF FF
$171C  01 21 4C 68 86 8C 92 B2
$1724  B7 C1 CB D2 E4 83 95 FF
$172C  3E 6A 8A 8E D3 17 1B 40
$1734  46 58 FF FF FF FF FF FF
$173C  01 02 03 03 03 04 03 01
$1744  03 03 03 02 02 02 02 02
$174C  F3 27 A6 A9 B3 28 3B AE
$1754  D7 DF E7 78 7C 80 84 88
$175C  02 02 01 01 01 03 03 03
$1764  03 03 03 03 03 03 03 03
       CHECKSUM = $262F

$17EC  78 D8 A2 E0 B5 00 48 E8
$17F4  D0 FA 98 48 A9 60 85 E8
$17FC  20 E8 00 BA CA CA 9A 68
$1804  38 E9 12 85 E8 68 E9 01
$180C  85 E9 A0 E0 A2 EA 18 B1
$1814  E8 65 E8 95 00 C8 E8 B1
$181C  E8 65 E9 95 00 C8 E8 D0
$1824  ED 18 A0 00 B1 EA B6 60
$182C  99 60 00 8A 91 EA C8 10
$1834  F3 B0 58 A0 B0 A2 E8 A9
$183C  4C 95 E8 E8 B1 E8 18 65
$1844  E8 95 E8 C8 E8 B1 E8 65
$184C  E9 95 E8 C8 E8 D0 E8 A2
$1854  E0 B1 E8 95 D0 C8 E8 D0
$185C  F8 A4 ED A6 EC D0 01 88
$1864  CA 86 B6 86 C4 86 CA 84
       CHECKSUM = $51C4

$18EC  C0 02 A0 00 B1 69 B0 18
$18F4  20 D3 00 48 B5 90 85 6B
$18FC  B5 A0 85 6C 68 AA B5 90
$1904  85 C7 B5 A0 85 C8 A4 60
$190C  AA E6 69 D0 02 E6 6A 60
$1914  B1 FE 65 E8 AA B1 FC 65
$191C  E9 48 8A 48 A4 60 18 A5

$176C  90 44 AE AE AE F5 3D 8C
$1774  5C 60 64 69 6E 73 78 82
$177C  01 03 03 04 04 04 03 03
$1784  02 03 03 02 04 02 02 03
$178C  98 9B FD 05 0F 1A 27 30
$1794  EC 40 4F E9 25 BD B2 8D
$179C  C6 01 3A 02 FC 00 BE 02
$17A4  73 02 CD 03 C1 03 00 00
$17AC  00 00 76 B5 53 3E 00 00
$17B4  CA D0 FA 60 7E 00 00 E8
$17BC  D0 FA 60 B9 00 00 00 00
$17C4  00 99 00 00 88 D0 F4 60
$17CC  D1 06 51 07 2E 04 10 00
$17D4  30 00 50 00 60 00 70 00
$17DC  80 00 90 00 A0 00 FF FF
$17E4  FF FF FF FF FF FF A0 00
       CHECKSUM = $2D77

$186C  B7 84 C5 84 CB A5 60 20
$1874  D9 00 A5 63 29 0F 85 63
$187C  68 D0 1D A5 B0 D0 2F F0
$1884  02 68 68 18 08 A5 62 48
$188C  38 B0 97 68 A8 28 A2 20
$1894  68 95 DF CA D0 FA 58 60
$189C  85 62 A5 EE 85 69 A5 EF
$18A4  85 6A A0 00 A2 F0 B1 E8
$18AC  95 90 C8 E8 D0 F8 A5 B0
$18B4  F0 06 20 D6 00 38 D0 CC
$18BC  A0 00 98 48 A9 CF 48 B1
$18C4  69 20 D3 00 85 6F B1 F6
$18CC  18 65 E8 AA B1 F4 65 E9
$18D4  48 8A 48 A5 6F C0 08 B0
$18DC  24 A8 90 2C B1 FA 65 E8
$18E4  AA B1 F8 65 E9 48 8A 48
       CHECKSUM = $4423

$196C  0A A9 39 D0 06 A9 F9 D0
$1974  02 A9 79 85 C6 0A 4C C3
$197C  00 84 6D A9 08 85 6E B1
$1984  6B 4A 48 90 03 20 DC 00
$198C  A6 61 20 BC 00 68 6A C6
$1994  6E D0 EF A4 6D 91 6B 88
$199C  D0 DF 60 A2 03 A8 B5 65
```

```
$1924  62 60 48 4A 4A 4A 4A A8
$192C  AA 68 29 0F 60 A0 00 C8
$1934  84 6D A9 08 85 6E B1 6B
$193C  85 6F 06 6F A6 60 20 B5
$1944  00 20 DC 00 AA 30 02 E6
$194C  6F C6 6E D0 ED A4 6D A5
$1954  6F 91 6B C4 60 D0 D8 8A
$195C  30 17 60 A4 60 B0 12 90
$1964  0C A9 59 D0 0E A9 19 D0
       CHECKSUM = $3E2B

$19EC  19 18 A2 04 36 64 CA D0
$19F4  FB 90 0D A2 04 B5 64 55
$19FC  B0 95 64 CA D0 F7 B0 02
$1A04  86 6F 68 2A C6 6E D0 DA
$1A0C  A4 6D 91 BD C8 D0 CA 60
$1A14  A6 60 20 B5 00 90 0A B0
$1A1C  07 A6 61 20 BC 00 90 01
$1A24  CA 86 64 60 B5 80 85 62
$1A2C  60 A4 61 48 B1 BD AA 68
$1A34  91 BD 8A C8 D0 F5 F0 EE
$1A3C  48 B1 B6 AA 68 91 B6 8A
$1A44  88 D0 F5 F0 E1 A5 62 B0
$1A4C  0B A5 62 B0 02 A5 64 F0
$1A54  24 60 A5 64 D0 1F 60 B1
$1A5C  EC 10 1A 60 B1 EC 30 15
$1A64  60 A4 60 B1 B6 D0 0E 88
       CHECKSUM = $42A3

$1AEC  91 B6 88 D0 F9 60 B1 B6
$1AF4  E9 00 91 B6 88 D0 F7 60
$1AFC  38 B1 B6 69 00 91 B6 88
$1B04  D0 F7 60 38 A9 00 F1 B6
$1B0C  91 B6 88 D0 F7 60 E6 62
$1B14  60 A5 62 95 80 60 4D 05
$1B1C  01 80 82 90 81 13 0F 06
$1B24  12 00 10 01 13 80 94 16
$1B2C  84 0A 06 12 00 05 01 03
$1B34  80 92 16 0A E5 90 11 93
$1B3C  13 13 18 14 85 93 17 17
$1B44  0A FD 16 14 84 30 13 0F
$1B4C  18 93 17 82 90 81 14 E3
$1B54  0F 0F 92 17 82 0B 09 12

$19A4  94 65 CA 10 F8 60 AA D0
$19AC  01 E8 86 60 8A 18 65 EC
$19B4  85 BD A9 FF 65 ED 85 BE
$19BC  A9 00 E5 60 85 61 A2 F0
$19C4  A5 B6 A4 B7 18 65 60 90
$19CC  01 C8 95 A0 94 B0 E8 D0
$19D4  F3 60 C6 62 60 A4 61 84
$19DC  6F 84 6D A9 08 85 6E B1
$19E4  BD 0A 48 A5 6F F0 02 90
       CHECKSUM = $4168

$1A6C  D0 F9 60 A4 60 B1 B6 F0
$1A74  01 60 88 D0 F8 8A 10 02
$1A7C  C6 6A 18 65 69 85 69 90
$1A84  02 E6 6A 60 20 E2 00 85
$1A8C  6A 20 E2 00 85 69 60 69
$1A94  10 A8 A5 69 20 DF 00 A5
$1A9C  6A 20 DF 00 B1 F2 65 EE
$1AA4  85 69 B1 F0 65 EF 85 6A
$1AAC  60 A6 63 E8 8A 29 0F AA
$1AB4  85 63 B5 70 60 A6 63 95
$1ABC  70 CA 8A 29 0F 85 63 60
$1AC4  B1 B6 91 C7 88 D0 F9 60
$1ACC  B1 C7 91 B6 88 D0 F9 60
$1AD4  B1 C7 48 B1 B6 91 C7 68
$1ADC  91 B6 88 D0 F3 60 A4 60
$1AE4  AA D0 05 98 D0 02 A9 00
       CHECKSUM = $42B7

$1B6C  80 E8 0C 0A 93 81 12 01
$1B74  10 0E F2 91 13 11 69 31
$1B7C  0E 03 1B 08 FA 11 59 05
$1B84  00 16 16 16 88 93 F6 83
$1B8C  17 0A FA 97 0F 09 80 93
$1B94  13 81 12 01 10 0F EE 32
$1B9C  0E EB 93 11 80 16 35 12
$1BA4  00 01 0A 03 12 E0 80 11
$1BAC  12 83 84 14 82 85 12 01
$1BB4  67 A7 A6 81 93 34 93 A2
$1BBC  E0 83 95 34 95 A4 E0 85
$1BC4  97 0E EB 94 0C 01 F5 11
$1BCC  17 1B 08 FC 11 58 A1 1A
$1BD4  A1 1A 68 1B 08 F7 11 17
```

```
$1B5C  00 01 01 03 80 13 0E EC
$1B64  14 11 05 01 80 90 14 14
       CHECKSUM = $2E15

$1BEC  57 63 1D A9 1A A9 19 69
$1BF4  1B 59 08 F7 06 80 19 66
$1BFC  1D 11 54 09 08 61 59 1C
$1C04  56 65 1C 07 0C 61 56 1B
$1C0C  59 12 65 1A 81 05 08 E1
$1C14  19 3B 67 37 83 12 87 14
$1C1C  86 6C 33 64 09 0C 62 1C
$1C24  1C 56 60 59 1C 58 61 1C
$1C2C  07 07 62 56 60 58 1B 59
$1C34  61 3B 81 12 68 38 17 64
$1C3C  08 01 17 83 67 12 38 86
$1C44  05 01 00 36 3A 6D 33 17
$1C4C  16 0A FD 16 16 64 09 01
$1C54  16 63 1D 11 63 1D 89 05
$1C5C  07 D9 19 55 99 1A 89 1E
$1C64  1B 1E 1E 1E 06 01 3C 88
       CHECKSUM = $1FAC

$1CEC  11 67 1C 1D 17 06 00 19
$1CF4  06 01 19 83 1E 1B 1E 1E
$1CFC  1E 90 06 01 19 17 8A 91
$1D04  8C 12 87 14 86 6A 33 89
$1D0C  99 86 9A 13 8A 87 36 1A
$1D14  08 F6 19 8B 9C 1A 55 1A
$1D1C  54 1A 82 9B 1A 64 1A 54
$1D24  65 1A 55 81 9A 64 1A 65
$1D2C  1A 80 99 1A 11 06 00 09
$1D34  02 06 01 5A 68 1D 81 67
$1D3C  1D 9D 1A 06 00 19 17 16
$1D44  14 87 99 27 89 50 9A 27
$1D4C  8A 51 9B 27 8B 52 9C 27
$1D54  8C 53 68 1D 91 11 6A 09
$1D5C  12 81 94 29 81 96 69 08
$1D64  07 16 16 14 80 00 60 96
       CHECKSUM = $23BE

$1BDC  81 05 10 38 67 37 16 14
$1BE4  11 12 6F 1A 6E 1A 87 11
       CHECKSUM = $28DF

$1C6C  06 00 3C 81 62 1B 38 A8
$1C74  81 98 62 1C 38 81 12 65
$1C7C  19 16 16 16 16 16 C1 80
$1C84  82 05 08 83 12 00 23 83
$1C8C  05 09 E3 19 57 62 1B 92
$1C94  39 A3 14 19 57 62 1C 91
$1C9C  39 82 84 12 00 43 94 11
$1CA4  06 00 09 02 06 01 54 63
$1CAC  1C 1D 06 00 19 A0 19 88
$1CB4  90 89 3A 98 64 08 0E 16
$1CBC  16 16 16 06 04 37 1C 19
$1CC4  17 16 14 07 09 21 E2 E3
$1CCC  14 86 36 87 98 21 83 99
$1CD4  1A 5A 6B 1C 5B 09 09 6A
$1CDC  19 85 97 84 30 1A 08 F1
$1CE4  19 A8 1A A8 1A A8 80 98
       CHECKSUM = $22BA

$1D6C  B1 65 11 28 81 95 80 97
$1D74  67 1C 1D 06 01 19 A0 19
$1D7C  54 06 00 19 55 A1 64 19
$1D84  65 19 66 55 06 00 19 82
$1D8C  65 1C 55 09 08 12 00 21
$1D94  01 20 1A 08 F1 67 1D 65
$1D9C  11 1B 09 11 1B 09 10 1B
$1DA4  09 0F 1B 09 0E 1B 09 0D
$1DAC  1B 09 0C 10 4E 20 4E 22
$1DB4  4E 23 4E 24 4E 25 4E 26
$1DBC  4E

       CHECKSUM = $1032
```

Listing VIC and 64 Programs with a PET

With optional spelled-out cursor and color commands

by Bruce Jaeger

If you'd like to use your CBM printer to make a nice, clean hard copy of a graphics program that you've created on the VIC 20 or Commodore 64, here's how to get your PET to translate those special graphic commands.

I finished writing a little graphics program recently on the VIC 20 with the Super Expander. I needed a hard copy of the program and, not having a printer connected to my VIC, I loaded the program into my trusty PET to make the listing on the PET's CBM printer.

Everything worked smoothly—until the PET got to the Super Expander's special commands. If it could talk, I'll bet the PET would have been saying "Hey! What's this COLOR command? And c'mon! I dunno nothin' about this CIRCLE. Gimme a break!" Then the PET would convert all the special Super Expander commands to words it already knew, like INPUT, DATA and so forth.

This led to a pretty unreadable hard copy! I was going to write a BASIC program to list the Super Expander commands properly, but I stopped myself in time. Why write "COLOR" and so on, when the VIC already knows how?

Instead of LISTing the program to the screen in the normal way, LIST the program to the disk or cassette in the following way: (Direct Mode)

(CASSETTE):

```
OPEN 1,1,1:CMD1:LIST
```

(DISK):

```
OPEN 1,8,1,"PROGRAM NAME,S,W
":CMD1:LIST
```

The program will then LIST itself to tape or disk! When it is finished, you MUST close the file:

```
PRINT #1:CLOSE 1
```

Then LOAD and RUN the Lister program on your PET with the printer. It will read the sequential file created above and print the listing.

As an option, I've included a routine to spell out the cursor and color control commands between quotes. I used that option when I had the Lister program list itself. If you have a printer for your VIC or 64, you may wish to use Lister just for the spelled-out cursor commands. C

Lister

```
100 REM PROGRAM TO PRINT LISTINGS
110 REM FROM SEQUENTIAL FILES
120 REM
130 REM BRUCE JAEGER
140 REM ST. PAUL, MN
150 REM
160 L=0:QM=0
170 DIM C$(255):FOR X=0 TO 255:C$(X)=CHR$(X):NEXT X
180 READ C,C$:IF C=-1 THEN 200
190 C$(C)=C$:GOTO 180
200 INPUT"[CLEAR]FILE[SPACE]NAME[SPACE3,CMDR I,LEFT3]";F$
```

```
210 IF F$="[CMDR I]"THEN F$="":REM FOR CASSETTE ONLY!
220 PRINT"[DOWN]CASSETTE[SPACE]OR[SPACE]DISK?"
230 GET CD$:IF CD$<>"C"AND CD$<>"D"THEN 230
240 IF CD$="C"THEN 260
250 INPUT"[DOWN]DRIVE[SPACE]#[SPACE3]0[LEFT3]";D$
260 INPUT"[DOWN]TITLE[SPACE]FOR[SPACE]LISTING[SPACE2,
    CMDR I,LEFT3]";T$
270 INPUT"[DOWN]LINES[SPACE]PER[SPACE]PAGE[SPACE3]55
    [LEFT4]";LP
280 PRINT"[DOWN]DO[SPACE]YOU[SPACE]WANT[SPACE]THE[SPACE]
    CURSOR[SPACE]COMMANDS"
290 PRINT"[RVS]S[RVOFF]PELLED[SPACE]OUT,[SPACE]OR[SPACE,
    RVS]N[RVOFF]ORMAL"
300 GET CC$:IF CC$<>"S"AND CC$<>"N"THEN 300
310 OPEN 4,4:PRINT#4,CHR$(1);T$
320 PRINT#4:L=L+2
330 IF CD$="C"THEN OPEN 2,1,0,F$:GOTO 350
340 OPEN 2,8,2,D$+":"+F$+",S,R"
350 GET#2,A$:RS=ST:IF CC$="N"THEN 390
360 IF A$=CHR$(34)OR A$=CHR$(98)THEN GOSUB 440:GOTO 390
370 IF QM=1 AND A$=CHR$(13)THEN QM=0
380 IF QM=1 THEN PRINT#4,C$(ASC(A$));:GOTO 410
390 PRINT#4,A$;
400 IF A$=CHR$(13)THEN L=L+1
410 IF L>LP THEN L=0:INPUT"[DOWN]NEXT[SPACE]SHEET[SPACE3]
    Y[LEFT3]";N$
420 IF RS<>64 THEN 350
430 CLOSE 2:CLOSE 4:END
440 REM QUOTE MODE?
450 IF QM=0 THEN QM=1:RETURN
460 QM=0:RETURN
470 DATA 5,"<WHITE>",8,"<DISABLE[SPACE]SHIFT[SPACE]
    COMM.>",9,"<ENABLE[SPACE]SHIFT[SPACE]COMM.>"
480 DATA 14,"<LOWER[SPACE]CASE>",17,"<DOWN>",18,
    "<REVERSE>",19,"<HOME[SPACE]CURSOR>"
490 DATA 20,"<DELETE>",28,"<RED>",29,"<RIGHT>",30,
    "<GREEN>",31,"<BLUE>"
500 DATA 142,"<UPPER[SPACE]CASE>",144,"<BLACK>",145,
    "<UP>",146,"<REVERSE[SPACE]OFF>"
510 DATA 147,"<CLEAR/HOME>",148,"<INSERT>",156,"<PURPLE>",
    157,"<LEFT>"
520 DATA 158,"<YELLOW>",159,"<CYAN>"
530 REM OPTIONAL C-64 DATA
540 DATA 129,"<ORANGE>",149,"<BROWN>",150,"<LT.[SPACE]
    RED>",151,"<GRAY[SPACE]1>"
550 DATA 152,"<GRAY[SPACE]2>",153,"<LT.[SPACE]GREEN>",154,
    "<LT.[SPACE]BLUE>",155,"<GRAY[SPACE]3>"
560 DATA -1,END
```

BAUDOT 2

by Bruce Cameron and David Cameron

In January, 1983, we published a simple method for receiving RTTY (radioteletype) on the VIC 20. After a year of experience we now have an improved version that should please anyone who enjoys chasing press, amateur, Interpol and other RTTY signals.

The original version used a switched crystal to provide extra baud rates. This version does it all with software. The function keys select various baud rates from those most usually encountered. Additionally, typing "B" on the keyboard causes the machine to ask "Baud Rate?" and you can then input any other. (It sometimes says "redo from start" and asks again. At most, you may have to tell it twice.)

As with the original program you can force it to go into letters with "L" and force it into figures with "F". This is useful when you have poor receiving conditions.

The original article described a minimal terminal unit, but any standard one will do. It needs only to connect pins B and C of the user port to Ground when it sees a signal.

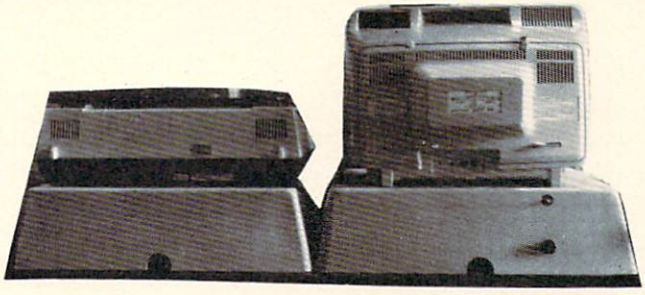
The program initializes with RS-232 configuration at 50 baud. It will run at this rate until you direct it otherwise. Most amateur stations use 45 and most commercial use either 50 or 75, but others are found here and there. If a station uses a non-standard rate you can find it merely by trying various rates in sequence, such as 45, 46, 47, 48, etc. Line 200 sets the function keys to the preselected rates in the order of F1, F3, F5, F7, F2, F4, F6, F8. (The even numbers are the shifted ones.) You can change any of these if you prefer. C

Baudot 2

```
5 REM BAUDOT
10 OPEN 2,2,0,CHR$(96+1)+CHR$(0)
15 FOR I=1 TO 8: READ BR(I):
   NEXT I
20 LS=-1
30 LF$=CHR$(10)
40 CR$=CHR$(13)
50 L$="E"+LF$+"A[SPACE]SIU"+CR$
   +"DRJNFCKTZLWHYPQOBG*MXV*"
60 F$="3"+LF$+"-[SPACE]'87"+CR$
   +"$4',!:(5')2#6019?&*./;*"
100 GET#2,C$:IF C$=""GOTO 160
110 C=ASC(C$):IF C<1 OR C>31
    GOTO 100
120 IF LS THEN C$=MID$(L$,C,1)
130 IF NOT LS THEN C$=MID$(F$,
    C,1)
140 IF C$<>"*" THEN PRINT C$;:
    GOTO 160
150 LS=(C=31)
160 GET X$:IF X$="" GOTO 100
161 IF X$="B" THEN INPUT"BAUD
    [SPACE]RATE";BR(0):A=132
    :GOTO 164
162 A=ASC(X$):IF A<133 OR A>140
    GOTO 170
164 CL=966667/BR(A-132)
166 HB=INT(CL/256):POKE 666,HB
168 POKE 665,INT(CL-HB*256)
170 IF X$="L" THEN LS=-1
175 IF X$="F" THEN LS=0
180 GOTO 100
200 DATA 45,50,57,60,67,74,100
    ,110
```

INNOVATIVE ORGANIZERS, INC.

For Commodore 64™ & VIC-20™



WELCOME, to the family of **INNOVATIVE ORGANIZERS.** These **ATTRACTIVE** power organizer stands, made of **ABS PLASTICS, U.L. RATED,** provide an opportunity for a more **PROFESSIONAL** looking computer installation, both at **HOME** and in the **WORK PLACE.**

There is **AMPLE** room under the organizer to place all the computer accessories: **POWER SUPPLY, MODEM, GAME CARTRIDGES, MEMORY EXPANSION,** and **MUCH MORE.**

Now, let's examine how these organizers work. There are 3 CRT-Models.

CRT-1 holds a Monitor or T.V. Set and a VIC 20 or 64. All wires & cartridges fit neatly under the organizer.

CRT-2 same as CRT-1, but also has: **ONE-STEP MASTER SWITCH** (permits 4 pieces of hardware to be turned on and off from one switch, plus circuit protection, incoming power fused with 6 AMP Fuse).

CRT-3 same as CRT-2, but also has: **SURGE & SPIKE PROTECTION** built in.

P-D-1: **HOLDS 1525 PRINTER & 1541 DISK DRIVE.** All wires & cartridges fit neatly under the organizer.

Ask about our Power Lock Switch & Power Portable Computer Fan & 2 Outlet Surge & Spike Protection.

Also: At present New Models are being developed for Commodore P-500 and B-500 Series.

I.O. INNOVATIVE ORGANIZERS YOUR ALTERNATIVE

P.O. Box 108, Markleville, IN 46056

CRT-1	$ 35.95
CRT-2	$ 59.95
CRT-3	$ 79.95
P-D-1	$ 38.95

Plus Shipping

CALL TO ORDER
317/533-2460

Trouble — DIAL
317/646-8943

Charge it on Visa
or Master Card

Dealer Inquiries Invited

Keyboards

by Elizabeth Deal

The table following is a cross-reference of key numbers used in some programs. It should be particularly valuable for users of the Commodore 64 computers, because there are numerous programs for the PET that can be converted for the 64. Unfortunately some use key numbers. This chart should make the conversion easier.

Please note that on the Upgrade and BASIC4 PETs PEEK(151) tells us if a key is being pressed. If it is not, the value is 255. On the 64 an opposite process is true: PEEK(197) returns 64 if a key is not held down.

The list may seem funny or incomplete; 64 users will not find several symbols. They are not shown simply because there is no ambiguity such as exists with the CBMs, which may have two numeric keys.

The STOP key, CTRL, Commodore and SHIFT keys never show up in location 197. They are weeded out by the key-scanning routine and are, in fact, used to select a keyboard or signal a stop situation. They are listed here only to give a complete picture from the "wiring" point of view.

One final note. Try to use GET in your programs if you can. Use key numbers only if you must, as in converting from the PET. C

```
100 "-------------------------------------------------------------------------:
101 "      GRAPHIC          CBM40COL         CBM80COL         COMMODORE64
102 "KEY#  ORIG/UPGR        BAS4             BAS4
103 "-------------------------------------------------------------------------:
500 "  0                                                     DEL INST         :
501 "  1   =                                                 RETURN           :
502 "  2   .                                                 CR CL            :
503 "  3                   COLON *          STOP RUN         F7               :
504 "  4   STOP RUN        STOP RUN                          F1               :
505 "  5   <               9 )                               F3               :
506 "  6   SPACE           6 &                               F5               :
507 "  7   [               3 #                               CD CU            :
508 "  8   RVS OFFRVS                                        3                :
509 "  9   -               ↑                TAB              W                :
510 " 10   0               / ?                               A                :
511 " 11   RIGHT SHIFT                                       4                :
512 " 12   >               CLS HOME                          Z                :
513 " 13   SPACE           M                RETURN           S                :
514 " 14   ]               SPACE                             E                :
515 " 15   @               X                                 LEFT SHIFT       :
516 " 16   LEFT SHIFT      RVS OFFRVS                        5                :
517 " 17   +               2                CD CU            R                :
518 " 18   2                                RVS OFFRVS       D                :
519 " 19                                     CLS HOME         6                :
520 " 20   ?               0                DEL INST         C                :
521 " 21   COMMA           COMMA <                           F                :
522 " 22   N               N                                 T                :
523 " 23   V               V                                 X                :
```

```
524 " 24   X              Z                        7                :
525 " 25   3              3                        Y                :
526 " 26   1                                        G                :
527 " 27   RETURN                                   8                :
528 " 28   SEMICOLON      .                        B                :
529 " 29   M              . >          CR CL       H                :
530 " 30   B              B                        U                :
531 " 31   C              C                        V                :
532 " 32   Z                           SPACE       9                :
533 " 33   *              4                        I                :
534 " 34   5              [                        J                :
535 " 35                  O                        0                :
536 " 36   COLON          CD CU                    M                :
537 " 37   K              U                        K                :
538 " 38   H              T                        O                :
539 " 39   F              E                        N                :
540 " 40   S              Q                        +                :
541 " 41   6              DEL INST                 P                :
542 " 42   4              P                        L                :
543 " 43   RETURN         I                        -                :
544 " 44   L              \            COMMA <     >                :
545 " 45   J              Y            - =         [                :
546 " 46   G              R            . >         @                :
547 " 47   D              W            / ?         <                :
548 " 48   A              TAB                      ENGL.POUND       :
549 " 49   /              6            1 !         *                :
550 " 50   8              @            2 "         ]                :
551 " 51                  L            3 #       CLS HOME           :
552 " 52   P              RETURN       4 $       RIGHT SHIFT        :
553 " 53   I              J            5 %         =                :
554 " 54   Y              G            6 &         ^                :
555 " 55   R              D            7 '         ?                :
556 " 56   W              A            8 (         1                :
557 " 57   9              5            9 )                          :
558 " 58   7              SEMICOLON +  COLON *     CTRL             :
559 " 59   ^ PI           K            SEMICOLON + 2                :
560 " 60   O              ]                        SPACE            :
561 " 61   U              H                        COMMODORE        :
562 " 62   T              F                        Q                :
563 " 63   E              S                        STOP RUN         :
564 " 64   Q              ESC          @                            :
565 " 65   DEL INST       9            A                            :
566 " 66   CD CU                       B                            :
567 " 67                  ^            C                            :
568 " 68   )              7 '          D                            :
569 " 69   \              0 TOP ROW    E                            :
570 " 70   '              7 '          F                            :
571 " 71   $              4 $          G                            :
```

```
572 " 72    QUOTE          1 !            H                    :
573 " 73    CR CL                         I                    :
574 " 74    CLS HOME                      J                    :
575 " 75    _              CR CL          K                    :
576 " 76    (              8              L                    :
577 " 77    &              - =            M                    :
578 " 78    %              8 (            N                    :
579 " 79    #              5 %            O                    :
580 " 80    !              2 QUOTE        P                    :
581 " 81                                  Q                    :
582 " 82                                  R                    :
583 " 83                                  S                    :
584 " 84                                  T                    :
585 " 85                                  U                    :
586 " 86                                  V                    :
587 " 87                                  W                    :
588 " 88                                  X                    :
589 " 89                                  Y                    :
590 " 90                                  Z                    :
591 "155                                  ESC                  :
592 "174                                  .                    :
593 "176                                  0                    :
594 "177                                  1                    :
595 "178                                  2                    :
596 "179                                  3                    :
597 "180                                  4                    :
598 "181                                  5                    :
599 "182                                  6                    :
600 "183                                  7                    :
601 "184                                  8                    :
602 "185                                  9                    :
603 "192                                  @                    :
604 "219                                  [                    :
605 "220                                  \                    :
606 "221                                  ]                    :
607 "222                                  ^                    :
608 "223    ELIZABETH DEAL                _                    :
```

A Keyboard Scan for the Commodore 64

by Elizabeth Deal

Have you ever thought about how a computer would behave if its operating system was written in BASIC? Well, I did and I didn't like it one little bit. It is slow; so slow, in fact, that you can watch it in slow motion. But that in itself can be an advantage, so I decided to like it for a while. The Commodore 64 permits us to observe things we could never do on the PET in BASIC, a keyboard scan being one example, since the interrupts can be turned off from BASIC.

There are many ways to get at the keyboard; the quickest is in machine code. A modified ROM routine can tell us how the keyboard is wired; hence we can get all the key numbers, not just those returned in location 197. The BASIC program at the end of this article imitates the process very slowly. You may press one or more keys and their numbers will sluggishly show up in reverse on an eight by eight display.

You should notice that you can reliably detect one and two keys being pressed. Press three and you may be in trouble, depending on whether the third key is in the same column as one of the previous two, since the row-column circuit is being closed.

The Commodore 64 keyboard is built in an eight by eight matrix (as opposed to PET's eight by ten). The scanning process is inverted from the PET's: here they put a zero on a row and look for a zero on a column. In any case the keys get numbered from zero to 63. A value of 64 means no key has been pushed.

The scanning routine places a pattern such as 11111110 to select the first row, rotates the pattern to 11111101 to select the second, and so on through all eight. It looks for a zero in a column so if a key in column four is pushed the pattern is 11101111. The pattern is shifted right until the zero appears in the carry bit. The key is then registered.

In both computers the scan takes the same amount of time independent of the key number, as all 80 (PET) or 64 (Commodore 64) combinations are looked at. The last situation is registered into location 197 (151 in PET), the keyboard buffer, and so on. The STOP key is the last thing it sees. Shift keys, and in case of the 64, the CTRL and Commodore-logo keys, are not logged, of course. They are used to select a keyboard decoding table from which to print the numbers, letters and graphics you see on the keytops. Except that in the 80-column machines the graphics are defeated on purpose for some reason (see Raeto West's book on PET/CBM).

The status of these special keys is logged in their own little registers. On the PET the shift keys are logged into 158. On the 64 the register is 653. Here, a shift key returns a one, the Commodore key a two, the control key a four, and decoding combinations are possible. A shift key is also logged in location 654 and 657; they are mostly used by the system.

My BASIC scanner for slow-motion display is a similar process except it registers the duplicates: it lets you see more than one key pressed. This could be valuable for two-player games if done quickly in machine code. It could also be used to detect three keys to play three voices on the SID, but only if you are careful in selecting the keys so no false triggering occurs. This is not easy.

Incidentally, the tables of letters are in ROM. If you know the key number, you can see how the computer looks up what to print on the screen by saying PRINT CHR$(PEEK(TABLE+KEY-NUMBER)), where TABLE is a number such as 60289 ($EB81 unshifted, $EBC2 shifted, etc).

Try not to use key numbers in your programs; they become next to impossible to convert to other systems. The main reason for this exercise is to get to know the computer better, and I think the little BASIC routine lets you do just that. Key #4, a function key, will not show up, since I use it to quit the program. Hold it down for a while to quit; don't tap it—it won't work. **C**

Reference: Jim Butterfield's memory maps

Note: (Listing on next page)

Keyboard Scan (continued from page 107)

```
1 REM"S=SAVE"0:BAS KB SCAN",8
10 REM-----------------------------
11 REM C64 KB-SCAN   ELIZABETH DEAL
20 REM-----------------------------
100 IN=56333:KR=56320:KC=KR+1:FF=255
110 FOR I=0 TO 7:P(I)=2^I:NEXT
115 PRINT"[CLEAR,DOWN9,SPACE3]C=[LEFT]"0;1;2;3;4;5;6;7
    :L=3
120 PRINT"[DOWN2]PUSH[SPACE]SEVERAL[SPACE]KEYS[SPACE]
    EXCEPT[SPACE]STOP"
130 PRINT"[DOWN]HOLD[SPACE,RVS]F1[RVOFF,SPACE]TO[SPACE]
    QUIT"
140 POKE IN,127
150 N=0:PRINT"[HOME]";
160 FOR R=0 TO 7:POKE KR,FF-P(R)
165 V=FF-PEEK(KC):PRINT"[RVS,SPACE2,LEFT2,RVOFF]"R"[LEFT]
    :";
170 FOR C=0 TO 7
175 Q=(V AND P(C))/P(C):Q$="[RVOFF]"
180 IF Q THEN Q=N:Q$="[RVS]":IF N=4 GOTO 220:REM F1
190 PRINT"[SPACE]"Q$RIGHT$("[SPACE2]"+MID$(STR$(Q),2),2)"
    [RVOFF]";
200 N=N+1:NEXT C:PRINT:NEXT R
210 GOTO 150
220 POKE KR,127:POKE IN,129
230 PRINT"[DOWN14]":END
```

Finding the Right Color Combinations for Your Commodore 64 and Your Monitor

by Gregory Yob

A while ago, I was working with a spreadsheet program on my Commodore 64 and my color TV. The program used white letters on a blue background in the working area where all of my dollars and cents were. This led to a major problem; I couldn't tell the numeral eight from the numeral zero and several other letter and numeral combinations were very hard to read!

I took a look at the chart of color combinations provided on page 152 of the *Commodore 64 Programmer's Reference Guide* and discovered that white over blue was considered an excellent color combination... but that wasn't on *my* monitor!! So, with my doubts in hand, I turned to the ultimate reference manual—my Commodore 64 and monitor. (If the manual doesn't seem right, the hardware is always right—if you experiment to find out what "right" really is.)

Program 1 resulted from my labors and experiments. It prints a line of hard-to-read characters in all 16 colors on the screen. One of these lines will be invisible, because it will match the background color. By pressing any key (space, which repeats, is fun) the background changes to the next color. Each line and the background are identified per the Commodore color names so you can quickly see which lines are readable for each background.

If you keep notes, a chart similar to Commodore's can be made. Figure 1 shows the chart for my monitor. These are the combinations I can read if I try hard enough. If I were more stringent (my spreadsheet surely is) the chart would have fewer dots on it. Use Program 1 and make your own chart. If you are writing software for commercial use, always make sure your colors are legible. I just LOVE trying to read purple over light blue!!

Explanation of Program 1

Lines 10 to 60 merely assert that I wrote this program and hope you will respect my creativity when you use it. **Lines 80 through 230** hold the names of each color and the CHR$ code for each one. (Note that the CHR$ codes are not completely described in the characters and keys tables in the Commodore 64 manuals.) These are arranged in the Commodore order, starting with color 0, black and ending with color 15, gray 3.

Lines 250 to 300 read the data into the color names array CN$() and the color ASCII codes array C(). The color characters are made in line 290 and go into the color strings array C$(). Then a hard-to-read string L$ is built in line 320. You can change L$ as you like, but note that the hardest to read characters are in L$ in as ugly a way as I could think of combining them.

Line 330 makes the screen clear and prints a title in the color white. I am using CHR$ codes for all the special characters so my printer can print the result. **Lines 340 to 370** go through the sixteen colors and make a line which starts with the color's name in white (white is usually legible) followed by the test string L$ in the selected color.

Line 380 tells you to press any key. Since the background is color number 6, line 390 notes this and jumps to 440, which POKEs the background color register to the current color (starting with blue, color 6). We then enter a loop in **lines 400 to 440**, which tells the background color and waits for your keypress. Line 410 prints a cursor-up for redoing the PRINTs in line 400 in the same place when the next color comes up. Note in line 430 how the background color, BK, counts through 0 to 15 and starts over. Have fun and learn a bit about your color monitor!

C

FIGURE 1

COMMODORE 64 LEGIBLE COLOR COMBINATIONS

BY GREGORY YOB
6/83

↓ BACKGROUND ↓ FOREGROUND →		BLACK 0	WHITE 1	RED 2	CYAN 3	PURPLE 4	GREEN 5	BLUE 6	YELLOW 7	ORANGE 8	BROWN 9	LIGHT RED 10	GRAY 1 11	GRAY 2 12	LIGHT GREEN 13	LIGHT BLUE 14	GRAY 3 15
BLACK	0		●		●					●				●			●
WHITE	1			●		●		●		●	●		●	●		●	●
RED	2		●							●		●					●
CYAN	3							●									
PURPLE	4	●	●														
GREEN	5	●	●												●		
BLUE	6		●		●											●	
YELLOW	7			●						●	●						
ORANGE	8		●	●							●						
BROWN	9		●						●								●
LIGHT RED	10		●	●													
GRAY 1	11		●											●			●
GRAY 2	12		●										●				●
LIGHT GREEN	13	●					●						●				
LIGHT BLUE	14	●	●				●										
GRAY 3	15		●	●				●					●	●			

NOTE: These combinations are for maximum legibility for word processing or spread sheets

Color Combination Program

```
80 DATA BLACK,144
90 DATA WHITE,5
100 DATA RED,28
110 DATA CYAN,159
120 DATA PURPLE,156
130 DATA GREEN,30
140 DATA BLUE,31
150 DATA YELLOW,158
160 DATA ORANGE,129
170 DATA BROWN,149
180 DATA LGT RED,150
190 DATA GREY 1,151
200 DATA GREY 2,152
210 DATA LGT GREEN,153
220 DATA LGT BLUE,154
230 DATA GREY 3,155
260 DIM CN$(15),C$(15),C(15)
270 FOR J=0 TO 15
280 READ CN$(J),C(J)
290 C$(J)=CHR$(C(J))
300 NEXT
320 L$="0000[SPACE]0123[SPACE]6699[SPACE]0698[SPACE]WMA@
    [SPACE]#$%&"
330 PRINT CHR$(147)CHR$(5)"MONITOR[SPACE]LEGIBILITY[SPACE]
    TEST"
340 PRINT:FOR J=0 TO 15
350 PRINT CHR$(5)CN$(J)TAB(10);
360 PRINT C$(J)L$
370 NEXT
380 PRINT:PRINT CHR$(5)"PRESS[SPACE]ANY[SPACE]KEY[SPACE]
    TO[SPACE]CHANGE[SPACE]BACKGROUND"
390 PRINT:BK=6:GOTO 440
400 PRINT CHR$(5)"BACKGROUND[SPACE]IS[SPACE]"CN$(BK)"
    [SPACE16]"
410 PRINT CHR$(145);
420 GET A$:IF A$=""THEN 420
430 BK=BK+1 AND 15
440 POKE 53281,BK:GOTO 400
```

TPUG: The World's Largest Commodore Computer Club

by Chris Bennett,
TPUG Business Manager

The Toronto PET User Group is probably the largest Commodore user group in the world. It's certainly the largest one registered with Commodore at the moment. It's also one of the oldest and most active groups, sponsoring seminars and computer fairs, publishing a regular magazine and collecting and distributing an enormous amount of public domain software. One of its original members, Jim Butterfield, has become known worldwide as an authority on using Commodore equipment.

TPUG's Business Manager, Chris Bennett, graciously agreed to take time from his hectic schedule to jot down a few notes about how our most successful group started—and grew—and grew some more.

Chris Bennett

Dealer area at 1983 TPUG convention.

The Toronto Pet User Group was five years old this summer. It started in 1978 with 35 people showing up for the first meeting in Lyman Duggan's party room. Jim Butterfield demo-ed some programs and amazed everyone with his sense of humor and knowledge of computers. The second meeting, also held in the party room, consisted of 50 people.

The meetings then moved to the Ontario Science Centre with membership up to 70 and still growing. A manufacturer of video projectors gave a demo of their commercial video projector that allowed the PET's screen to be displayed on a 15-foot by 15-foot screen. Frank Winter, who was then the Dean of Computing at Sheridan College, saw this and got

them to purchase the projector. The club moved out into Sheridan College's lecture theater in order to use this new device (we still have one meeting a month that uses the original projector).

Up to this time Lyman Duggan did all the organization for the club. When he was sent to Florida by his company, Lyman turned the club over to a board of directors. This basically consisted of everyone who was willing to put time and effort into organizing the club. I was put in charge of membership. Because of the need to keep track of our members, I wrote a disk-based mail list for the PET 2001 and 2040 disk drive. This program is now part of our library and many of the features found there were put in because of a specific need for keeping track of TPUG's members. Since then the club has grown by leaps and bounds to its present size of just under 10,000 members.

When I bought my PET in March of 1978, all that I received with it was a very small manual with a list of the BASIC commands and some general programming samples. There were no books or manuals about the computer because this was a new experience for everyone. The only way to get information about the PET was to get together in groups and share our knowledge. This pooling of information was the reason users groups have sprung up all over the world. Many of the utility programs I used and still use were contributed freely by user group members from many different parts of the country. Because these programs are freely passed from user group to user group, many of them are improved and enhanced as they make their rounds. For example, BASIC AID, which has many commands to help programmers write and debug BASIC programs, has improved many times over the years as various people have worked on it.

From one meeting a month, TPUG now runs four chapters each month. One each for the Commodore 64, VIC 20 and PET/CBM (SuperPET in the second half) and one general meeting for all machines. The format is generally the same. Each meeting is divided into two parts with a coffee break in between. The coffee break gives everyone a chance to mix and mingle with other club members. Some even say this is the best part of the evening.

Jim Butterfield teaching a TPUG class.

Both the first and second halves of the meeting are divided up into 15 to 30 minute segments for presentations by our members. By having four or five different subjects discussed and shown each night, everyone gets a chance to see something that interests them.

One thing we decided not to do at our regular meetings was to have club business discussed. This we reserve for special business meetings called once or twice a year as the need arises. We also have an annual meeting for the election of the board of directors. However, this is a special business meeting without any computers or demonstrations. By doing this, we get people who are interested in the running of the club and can spend all evening on any issues that may come up. Since few people are actually interested in how the club is run, the size of the group is much more manageable. When the board meets, they then decide who is going to hold which positions.

In addition to these regularly scheduled meetings, we also run special sessions. One of these is the machine language group taught by Jim Butterfield. This generally runs from October to April each year and teaches the new programmer how to write programs in machine language. Another special session we started this year is our summer session for new members who want to learn programming in BASIC on their VIC 20 or Commodore 64. This is the first time we have run an activity in the summer and the 200 spaces available were quickly filled up.

One of the big events of each year is our Annual

Conference. This ran for two days in May at George Brown College. We held a super disk copy session where up to 40 disks were copied free of charge. There was also a Dealer Exhibition where many new products were shown and sold. There were 38 seminars presented by various people throughout the weekend. On Saturday, Jim Butterfield taught a special one-day seminar on machine language. This session was filled within a week after it was announced. Next year, we expect the Annual Conference to be bigger and better.

About four months ago, TPUG bought an Electrohome video projector, which allows us to produce a 20-foot diagonal picture of the screen from any one of the Commodore computers we are using. Until we got this device, we were using Commodore Canada's video projector plus one at Sheridan College. Getting our own projector has given us the freedom to organize more meetings each month. This is still the greatest problem for most computer clubs. Once the size of the club increases, the ability to show the members what is going on gets increasingly more and more difficult.

After operating for a few months with our new board of directors and a new style of organization, it became apparent that we could help our members in many ways. The first, of course, is the monthly meetings. The second is by providing to the members copies of the programs from our public domain library. One thing I should make clear is that no commercial programs are kept in our library. We have over 4000 public domain programs with more coming in each week. If a commercial program is added to the library by mistake, it is removed as soon as the error is discovered.

At first, these programs were available only on tape. When the Commodore disk drives arrived, most of us quickly moved to that medium for the ease of copying programs. Now we provide copies of all our programs on both tape and disk. At meetings, each member hands in a diskette. We then copy that night's programs onto it and return them all at the end of the evening. This now requires that we have from six to eight 4040 disk drives at each meeting to copy the more than 350 diskettes handed in. Although it is a lot of work, most members like to get that evening's programs right away rather than waiting a month to pick them up. We also have our tapes commercially duplicated and available for members to buy at each meeting.

The third way to help members is by publishing a newsletter. We now have the *TORPET* printed and sent to all our members ten times a year. This not only provides the members with meeting times and places, but also supplies additional information such as the lists of program directories, articles of general interest and other useful information.

After a few months of operation, someone asked if we could have a special membership classification for out-of-town members who could rarely attend meetings. For this type of person, we created the "associate" membership. This person receives our newsletter and can order tapes and disks from our library through the mail. Our disks are $10 each ($12 for 8050) and our tapes are $6 each. Each tape or disk contains from 15 to 62 programs. This has been very popular for people in out-of-the-way places where there is no local club and often no local Commodore dealer.

One of the big changes that occurred this year was that we opened up an office with two full-time people, one of them being myself. Coupled with this, we have incorporated as a non-profit organization. We are now in the process of getting other clubs to affiliate with us. Members of these clubs join TPUG at a reduced rate by signing up 15 or more members at a time. Depending upon the number of members signed up, we then send to that club from one to three of our monthly diskettes for the VIC 20, Commodore 64 and PET/CBM. In this way we hope to circulate the many excellent public domain programs available for the Commodore machines.

In summary, I would like to say that the Toronto Pet Users Group was the best thing that happened to me as a Commodore user. I have learned more in the last five years than I could have learned in a lifetime without this organization. If any of you know of a users group in your area, JOIN IT NOW. If there is no local group, get some people together and START ONE. The more you talk and mix with other Commodore users, the more fun you will have with your own computer. **C**

User Group Listing

ALABAMA

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

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

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

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.

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

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

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

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

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

ARKANSAS

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

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

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

CALIFORNIA

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

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

Valley Computer Club
1913 Booth Road
Ceres, CA 95307

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

user groups

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

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

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

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

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

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

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

El Shift OH
P.O. Box 548
Cocoa, FL 32922
Mike Schnoke
Sat. mornings/every 4 to 6 weeks

Miami 20/64
12911 S.W. 49th St.
Miami, FL 33175
305-226-1185

Tampa Bay Commodore Computer Club
10208 N. 30th St.
Tampa, FL 33612
813-977-0877

GEORGIA

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

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

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

Golden Isles Commodore Users Club
Bldg. 68, FLETC
Glynco, GA 31524
Richard L. Young

HAWAII

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

20/64 Hawaii
P.O. Box 966
Kailua, HI 96734
Wes Goodpaster

Commodore Users Group of Honolulu
1626 Wilder #701
Honolulu, HI 96822
808-848-2088
Jay Calvin 808-944-9380

IDAHO

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

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

Commodore Users
548 E. Center

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

RAP 64/VIC Regional
Assoc. of Programmers
10721 S. Lamon
Oak Lawn, IL 60453
Bob Hughes

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

WIPUG
Rt. 5, Box 75
Quincy, IL 62301
217-656-3671
Edward Mills

INDIANA

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

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

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

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

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

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

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

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

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

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

Commodore Computer Club
3814 Terra Trace
Evansville, IN 47711
(812) 477-0739
John Patrick, President

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

Walnut Valley Commodore User Group
1003 S. 2nd St.
Arkansas City, KS 67005
Bob Morris

KENTUCKY

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

Louisville Users of Commodore KY. (LUCKY)
c/o Computer Showroom
1247 Hurstbourne
Louisville, KY 40222
2nd Monday

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
5112 Parklawn Terrace
Apt. #103
Rockville, MD 20853
(301) 231-7823
Tom Pounds

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

Westinghouse BWI Commodore User Group
Attn: L. Barron Mail Stop 5156
P.O. Box 1693
Baltimore, MD 21203

MASSACHUSETTS

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

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

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

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

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

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

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

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

VIC Interface Club
48 Van Cliff Ave.
Brockton, MA 02401
Bernie Robichaud

Cape Cod 64 Users Group
358 Forrest Rd.
S. Yarmouth, MA 02664
1-800-225-7136
Jim Close
(In MA. call) 1-800-352-7787

MICHIGAN

David Liem
14361 Warwick Street
Detroit, MI 48223

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

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

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

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

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

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

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

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

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

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

ComputerTowne
35171 Grand River
Farmington, MI 48024
(313) 471-4216

Ann Arbor Commodore Users Group
Ann Arbor, MI 48103
313-994-4751
Art Shaw
3rd Tues. 7:30-10:00

DAB Computer Club
P.O. Box 542
Watervliet, MI 49098
616-463-5457
Dennis Burlingham

West Michigan Commodores
c/o R. Taber
1952 Cleveland Ave., S.W.
Wyoming, MI 49509
616-458-9724
Gene Traas

MINNESOTA

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

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

MISSISSIPPI

Commodore Biloxi User Group (ComBUG)
Universal Computer Services
3002 Hwy. 90 East
Ocean Springs, MS 39564
601-875-1173
John Lassen

MISSOURI

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

Commodore User Group of St. Louis
Box 6653
St. Louis, MO 63125-0653
Dan Weidman, New Members
1541 Swallowtail Dr.
St. Louis, MO

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

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

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

Joplin Commodore Computers Users Group
422 S. Florida Ave.
Joplin, MO 64801
R. D. Connely

user groups

MONTANA

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

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

NEBRASKA

Greater Omaha Commodore 64
Users Group
2932 Leawood Dr.
Omaha, NE 68123
(402) 292-2753
Bob Quisenberry

NEVADA

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

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

Parsippany Computer Group
51 Ferncliff Rd.
Morris Plains, NJ 07950
(201) 267-5231
Bob Searing

NEW HAMPSHIRE

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

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

C-64 U.S.E.R.S. User Software
Exchange Pro
P.O. Box 4022
Rochester, NH 03867
Paul Kyle

NEW MEXICO

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

NEW YORK

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

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

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

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

Commodore Masters
25 Croton Ave.
Staten Island, NY 10301
Contact: Stephen Farkouh

VIC Users Club
76 Radford St.
Staten Island, NY 10314
Contact: Michael Frantz

West Chester County VIC
Users Group
P.O. Box 146
Pelham, NY 10552
Joe Brown

SPUG
4782 Boston Post Rd.
Pelham, NY 10803
Paul Skipski

VIC 20 User Club
151-28 22nd Ave.
Whitestone, NY 11357
Jean F. Coppola

VIC 20 User Club
339 Park Ave.
Babylon, NY 11702
(516) 669-9126
Gary Overman

VIC User Group
1250 Ocean Ave.
Brooklyn, NY 11230
(212) 859-3030
Dr. Levitt

L&M Computer Club
VIC 20 & 64
4 Clinton St.
Tully, NY 13159
(315) 696-8904
Dick Mickelson

Commodore Users Group
1 Corwin Pl.
Lake Katrine, NY 12449
J. Richard Wright

VIC 20/Commodore 64
Users Group
31 Maple Dr.
Lindenhurst, NY 11757
(516) 957-1512
Pete Lobol

VIC Information Exchange
Club
336 W. 23 St.
Deer Park, NY 11729
Tom Schlegel
SASE & phone please

New York Commodore
Users Group
380 Riverside Dr., 7Q
New York, NY 10025
(212) 566-6250
Ben Tunkelang

Hudson Valley Commodore Club
1 Manor Dr.
Woodstock, NY 12498
F.S. Goh
1st Wednesday of month

LIVICS (Long Island VIC Society)
20 Spyglass Lane
East Setauket, NY 11733
(516) 751-7844
Lawrence Stefani

VIC Users Group
c/o Stoney Brook Learning Center
1424 Stoney Brook Rd.
Stoney Brook, NY 11790
(516) 751-1719
Robert Wurtzel

Poughkeepsie VIC User Group
2 Brooklands Farm Rd.
Poughkeepsie, NY 12601
(914) 462-4518
Joe Steinman

VIC 20 User Group
Paper Service Division
Kodak Park
Rochester, NY 14617
David Upham, Sr.

Manhatten 64
426 West 48th
New York, NY 10036
(212) 307-6519
Charles Honce

Adirondack Commodore 64 Users Group
205 Woodlawn Ave.
Saratoga Springs, NY
(518) 584-8960
Paul Klompas

Rockland County Commodore
Users Group
P.O. Box 573
Nanuet, NY 10965
Ross Garber

New York 64 Users Group
222 Thompson St.
New York, NY 10012
212-673-7241
Bruce Cohen

Finger Lakes Commodore Users Group
c/o Rose City Computer Associates
229 West Union St.
Newark, NY 14513
315-331-1185

NORTH CAROLINA

Amateur Radio PET Users Group
P.O. Box 30694
Raleigh, NC 27622
Contact: Hank Roth

VIC Users Club
c/o David C. Fonenberry
Route 3, Box 351
Lincolnton, NC 28092

Microcomputer Users Club
Box 17142 Bethabara Sta.
Winston-Salem, NC 27116
Joel D. Brown

VIC Users Club
Rt. 11, Box 686
Hickory, NC 28601
Tim Gromlovits

Raleigh VIC 20/64 Users Group
410-D Delta Court
Cary, NC 27511
919-469-3862
Larry Diener

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

C.P.U. Connection
P.O. Box 42032
Brook Park, OH 44142
Danni Hudak

Commodore Users Group
18813 Harlan Dr.
Maple Heights, OH 44137
216-581-3099
Carl Skala

OKLAHOMA

Southwest Oklahoma
Computer Club
c/o Commodore Chapter
P.O. Box 6646
Lawton, OK 73504
1:30 at Lawton City Library

Tulsa Area Commodore Users Group
Tulsa Computer Society
P.O. Box 15238
Tulsa, OK 74112
Annette Hinshaw

Commodore Oklahoma Users Club
4000 NW 14th St.
Oklahoma City, OK 73107
(405) 943-1370
Stanley B. Dow

Commodore Users
Box 268
Oklahoma City, OK 73101
Monte Maker, President

Commodore Users of Norman
209 Brookwood
Noble, OK 73068
Matt Hager

OREGON

NW PET Users Group
John F. Jones
2134 N.E. 45th Avenue
Portland, OR 97213

Pacific Northwest Commodore
Users Group
P.O. Box 2310
Roseburg, OR 97470
503-672-7591
Richard Tsukiji

PENNSYLVANIA

PET User Group
Gene Beals
P.O. Box 371
Montgomeryville, PA 18936

Penn Conference Computer Club
c/o Penn Conference of SDA
720 Museum Road
Reading, PA 19611
Contact: Dan R. Knepp

PACS Commodore Users Group
LaSalle College
20th & Olney Ave.
Philadelphia, PA 19141
(215) 951-1258
Stephen Longo

Glen Schwartz
807 Avon
Philadelphia, PA 19116

Gene Planchak
4820 Anne Lane
Sharpsville, PA 15150
(412) 962-9682

PPG (Pittsburgh PET Group)
c/o Joel A. Casar, DMD
2015 Garrick Drive
Pittsburgh, PA 15235
(412) 371-2882

Westmoreland Commodore
Users Club
c/o DJ & Son Electronics
Colonial Plaza
Latrobe, PA 15650
Jim Mathers

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
c/o Gilbert Associates, Inc.
P.O. Box 1498
Reading, PA 19607
Extension 6472
Jo Lambert (215) 775-2600

Boeing Employees Personal
Computer Club
The Boeing Vertol Co.
P.O. Box 16858
Philadelphia, PA 19142
(215) 522-2257
Jim McLaughlin

South Central PA Commodore Club
2109 Cedar Run Dr.
Camp Hill, PA 17011
(717) 763-4219
David Persing

Main Line Commodore Users
Group (MLCUG)
c/o Main Line Computer Center
1046 General Allen Lane
West Chester, PA 19380
(215) 388-1581
Emil Volcheck

Commodore Users Group
781 Dick Ave.
Warminster, PA 18974
Matt Matulaitis

PUERTO RICO

CUG of Puerto Rico
RFD #1, Box 13
San Juan, PR 00914
Ken Burch

VIC 20 User Group
655 Hernandez St.
Miramar, PR 00907
Robert Morales, Jr.

RHODE ISLAND

Irving B. Silverman, CPA
160 Taunton Ave.
E. Providence, RI 02914
Contact: Michelle Chavanne

Newport VIC/64 Users
10 Maitland Ct.
Newport, RI 02840
(401) 849-2684
Dr. Matt McConeghy

The VIC 20 Users Club
Warwick, RI 02886
Tom Davey

Commodore Users Group
c/o Data-Co.
978 Tiogue Ave.
Coventry, RI 02816
401-828-7385
Victor Moffett

SOUTH CAROLINA

Beaufort Technical College
100 S. Ribaut Rd.
Beaufort, SC 29902
Dean of Instruction

Computer Users Society
of Greenville
Horizon Records-Home Computers
347 S. Pleasantburg Dr.
Greenville, SC 29607
(803) 235-7922
Bo Jeanes

Commodore Computer Club
of Columbia
318 Quincannon Dr.
Columbia, SC 29210
Buster White Sect./Treas.

Spartanburg Commodore Users Group
803 Lucerne Dr.
Spartanburg, SC 29302
803-582-5897
James Pasley

SOUTH DAKOTA

PET User Group
515 South Duff
Mitchell, SD 57301
(605) 996-8277
Contact: Jim Dallas

VIC/64 Users Club
608 West 5th
Pierre, SD 57501
(605) 224-4863
Larry Lundeen

TENNESSEE

River City Computer
Hobbyists
Memphis, TN
1st Mon. at Main Library

Nashville Commodore Users Group
P.O. Box 121282
Nashville, TN 37212
3rd Thurs. at Cumberland Mus

Commodore User Club
Metro Computer Center
1800 Dayton Blvd.
Chattanooga, TN 37405
Mondays 7:30 pm

Metro-Knoxville 64 Users Club
7405 Oxmoor Rd., Rt. #20
Knoxville, TN 37921
(615) 938-3773
Ed Pritchard

Memphis Commodore Users Group
2476 Ridvers Ave.
Memphis, TN 38127
901-358-5823
Harry Ewart

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

Gulf Coast Commodore Users Group
P.O. Box 128
Corpus Christi, TX 78403
(512) 887-4577
Lawrence Hernandez

Mid-Cities Commodore Club
413 Chisolm Trail
Hurst, TX 76053
Garry Wordelman

Mid-Cities Commodore Club
413 Chisolm Trail
Hurst, TX 76053
Bruce Nelson

Interface Computer Club
814 North Sabinas
San Antonio, TX 78207
M. E. Garza, President

UTAH

Utah PUG
Jack Fleck
2236 Washington Blvd.
Ogden, UT 84401

The Commodore Users
Club
742 Taylor Avenue
Ogden, UT 84404
Contact: Todd Woods Kap,
President
David J. Shreeve,
Vice President

The VIClic
799 Ponderosa Drive
Sandy, UT 84070
Contact: Steve Graham

VIC 20 Users
324 N. 300 W.
Smithfield, UT 84335
Dave DeCorso

user groups

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

Norfolk Users Group
1030 West 43rd St. B-4
Norfolk, VA 23508
489-8292
Larry Pearson

NASA VIC 20 User Group
713 York Warwick Dr.
Yorktown, VA 23692
Harris Hamilton

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

C.U.S.S.H.
3614 Sovereign Dr.
Racine, WI 53406
(414) 554-0156
Tim Tremmel
3rd Saturday of month

Madison Area Commodore Users Group
1552 Park St.
Middleton, WI 53562
608-831-4852
John Carvin
3rd Thurs. each month

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, Inc.
1912A Avenue Rd., Ste. 1
Toronto, Ontario, Canada
M5M 4A1
(416) 782-8900
or call 416-782-9252
Contact: Chris Bennett

PET Users Club
c/o Mr. Brown
Valley Heights Secondary School
Box 159
Langton, Ont. N0E 1G0

Vancouver PET Users Group
P.O. Box 91164
West Vancouver, British
Columbia
Canada V7V 3N6

CCCC (Canadian
Commodore Computer Club)
c/o Strictly Commodore
47 Coachwood Place
Calgary, Alberta, Canada
T3H 1E1
Contact: Roger Olanson

W.P.U.G.
9-300 Enniskillen Ave.
Winnipeg, Manitoba R2V 0H9
Larry Neufeld

VIC-TIMS
2-830 Helena St.
Trail, British Columbia
V1R 3X2
(604) 368-9970
Greg Goss

Arva Hackers
Medway High School
Arva, Ontario N0M 1C0
D. Lerch

Nova Scotia Commodore
Computer Users Group
66 Landrace Cres.
Dartmouth, N.S. B2W 2P9
Andrew Cornwall

Bonnyville VIC Cursors
Box 2100
Bonnyville, Alberta T0A 0L0
(403) 826-3992
Ed Wittchen

Commodore Users Club of Sudbury
938 Brookfield Ave.
Sudbury, Ontario
P3A 4K4

PET Educators Group
P.O. Box 454
Station A
Windsor, Ontario
N9A 6L7

COMVIC
P.O. Box 1688
St. Laurent
Montreal, Quebec
H4L 4Z2

Calgary Commodore Users Group
37 Castleridge Dr., N.E.
Calgary, Alberta
T3J 1P4
John Hazard

FINLAND

VIC-Club in Helsinki
c/o Matti Aarnio
Linnustajankj 2B7
SF-02940 ESP00 94
Finland

GERMANY

Kettenberg 24
D 5880 Lueden Scheid
West Germany
Rudi Ferrari

ITALY

Commodore 64 Club
Universita di Studi shan
V. Avigliana 13/1
10138 TORINO
ITALY

KOREA

Commodore Users Club
K.P.O. Box 1437
Seoul, Korea
Contact: S. K. Cha

MEXICO

Asociacion De Usarios
Commodore
c/o Alejandro Lopez
Arechiga
Holbein 174-6° Piso
Mexico 18, D.F.

Club de Usarios Commodore
Sigma del Norte
Mol del Valle, Local 44
Garza Garcia, N.L. 66220

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

that does not compute...

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

ROM II

When the first public domain software was released, it was written for the old ROM versions of the 64. When the new ROM was released, the software was rewritten to work on the new ROM version of the 64. If you happen to have some of the old software and it won't run on your new ROM version of the 64, here is a program which may fix the problem. Basically, this program changes the color memory of the screen to match the current cursor color. This is updated about once every two seconds. This won't fix every problem with every program, but it may let you run some programs that you couldn't before. C

```
10 REM 52736-52794 : BY : DANIEL BINGAMON
20 REM $CE00-$CE3A HEX ADDRESS
40 PRINT"[CLEAR,DOWN,SPACE]COMMODORE[SPACE]
   64[SPACE]ROM [SPACE]2[SPACE]SCREEN
   [SPACE]ADJUSTMENT"
50 FOR I=52736 TO 52793
55 READ A:POKE I,A:B=B+A
60 NEXT
70 DATA 169,206,160,13,120,141,21
80 DATA 3,140,20,3,88,96,234,234
90 DATA 234,238,255,207,173,255,207
100 DATA 201,0,208,29,169,0,133,87
110 DATA 169,216,133,88,169,0,160
120 DATA 0,162,0,173,134,2,145,87
130 DATA 200,208,248,230,88,232,224
140 DATA 4,208,241,76,49,234
150 IF B<>7924 THEN PRINT"THERE[SPACE]IS
    [SPACE]AN[SPACE]ERROR[SPACE]IN[SPACE]
    YOUR[SPACE]TYPING"
160 POKE 53248,0:REM INITIALIZE COUNTER
170 PRINT"[DOWN2,SPACE]SYS[SPACE]52736
    [SPACE]TO[SPACE]START"
180 PRINT"[DOWN,SPACE]RUN/STOP[SPACE]
    RESTORE[SPACE]TO [SPACE]RESTORE
    [SPACE]TO[SPACE]NORMAL"
190 END
```

User Bulletin Board

User Groups Forming:

CALIFORNIA

Commodore 64 user group forming.
Contact LOGIKS
Box 4095
San Rafael 94913

CONNECTICUT

Commodore 64 user group forming.
Contact Robert Kind
P. O. Box 1608
Groton 06340
(203) 887-0238
or Tad Church
(203) 442-5314

NEW YORK

Broome County 64 Users Group
Contact Richard Sher
31-S Jane Lacy Drive
Endicott 10760
(607) 754-7382

Radio Buffs: Amateur radio operators interested in the VIC 20 and the Commodore 64 now meet Saturdays at 3:00 pm (Eastern Time) on 7228 Kh. Anyone is welcome. Per Bruce Cameron, Temple Terrace, Florida.

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:
Viasala, Inc.
2 Tower Office Park
Woburn, MA 01801
617-933-4500

Product:
Home Automatic Weather Station—For use with VIC 20 and Commodore 64. Combines a professional quality weather sensor with a software package that teaches, forecasts and graphically displays weather. The sensor is the same one used by weather services in 60 countries. The package allows the user to monitor weather conditions inside or outside the home and interact with the software to help predict and cope with changing weather conditions.
Price: $199.95

Vaisala Home Automatic Weather Station


The Byte Bat

Company:
MicroTie Systems Corporation
P.O. Box 8112
Walnut Creek, CA 94546
800-227-3900

Product:
Byte Bat™—A foam rubber baseball bat, 17 inches long, that gives you a harmless but satisfying way to "strike back" at your computer when you get frustrated by its quirks. Package includes a complete user's manual, user button, poster and warning decal. Compatible with all Commodore computers.
Price: $9.95 retail, $12.50 postpaid

Company:
Micro Format
1271 West Dundee Road,
Suite 16A
Buffalo Grove, IL 60090
312-537-2426

Product:
Continuous forms for personal computers—Labels in two sizes: $3\frac{1}{2}'' \times {}^{15}/_{16}''$ and $5'' \times 2{}^{15}/_{16}''$. Index cards in two sizes: $5'' \times 3''$ and $6'' \times 4''$. Clean edge letterhead with continuous #10 envelopes. Rolodex-style cards.
Price: Send for catalog

Company:
Fantasy Computerware
P.O. Box 451
Sioux Falls, SD 57101
605-335-7684

Product:
Software for the Commodore 64—*Flight 64* flight simulator provides a flight panel on your computer screen with full-color graphic displays including radar, altimeter, artificial horizon, vertical speed indicator and other instruments. Topography changes with every flight. *Spellathon* is a spelling tutor for all ages that allows you to build and save your own word lists.
Price: Simulator $15.95; Spelling $19.95. Send for free catalog

Company:
Spinnaker Software
215 First Street
Cambridge, MA 02142
617-868-4700

Product:
Educational cartridge software for the Commodore 64—Titles

include *Alphabet Zoo, Kindercomp, Kids on Keys, Facemaker, Story Machine, Delta Drawing, Fraction Fever, Up for Grabs, Delta Music* and *Cosmic Life.*
Price: Contact company

Company:
Merritt Software, Inc.
P.O. Box 1504
Fayetteville, AR 72702
501-442-0914

Product:
MathWiz—Menu-driven math tutoring package for Commodore computers. Includes color graphics and other special effects. Designed for use by grades 5-8.
Price: $100.00

Company:
Davidson & Associates
6069 Groveoak Place #14
Rancho Palos Verdes, CA 90274
213-378-7826

Product:
Speed Reader II—On disk for the Commodore 64. Six stimulating activities designed by reading specialists to increase reading speed and build up comprehension. Developed and tested in the classroom for adults and students in high school or college. Additional data disks for junior high and upper elementary students are available.
Price: $69.95 two-disk set

Master Math from PMI, Inc.

Company:
PMI Incorporated
High Street, P.O. Box 87
Buckfield, ME 04220
207-336-2500

Product:
Master Math—On disk or tape for PET/CBM and Commodore 64. A comprehensive, self-paced program for teaching high school math, including algebra, geometry, trigonometry, statistics and basic accounting.
Price: $150.00 six-disk or cassette package; $30 single disk

Company:
Smoky Mountain Software
54 West Main Street
Brevard, NC 28712
704-883-2595

Product:
The Grade Manager—For VIC 20 and Commodore 64. Alphabetically sorts student lists, keeps track of assignments and grades and calculates averages for entire term. Prints out assignment summaries, student grades and averages and incomplete assignments. VIC version requires 8K memory expansion.
Price: $29.95 tape; $34.95 disk

Company:
Tamarack Software
P.O. Box 247
Darby, MT 59829
406-821-4596

Product:
GradeCalc—For the Commodore 64. Grade attendance and management package that files grades and assignments and generates several different kinds of reports including grade totals, averages, assignment summaries and a cumulative listing of missing assignments.
Price: $29.95 disk

Company:
Useful Software
Box 54-H
Scarsdale, NY 10583

Product:
Two software packages for VIC 20 and Commodore 64—*The College Pak* contains more than 25 programs for computer-aided instruction in college-level math, physics, chemistry, engineering, language, history and medicine. *The Investors Pak* contains over 25 business and investment programs in real estate, mortgages, bonds, loans, shelters and more.
Price: College $29.95 disk; Investor $39.95 disk

new products

Company:
Automated Design
P.O. Box 507
Valley Forge, PA 19481
215-935-2420

Product:
AUTOPLAN—Computer-aided drafting program for CBM 8032. Standard details, plan overlays, title blocks, logos and dimensions are stored on file. Drawing elements can be stretched, sheered or rotated about a point or axis and can be edited by addition or deletion of lines or dimensions. Provides fast new drawings because standard parts can be assembled into new configurations. Automatic scale changing makes it possible to combine Imperial scales with metric equivalents and convert from one to the other.
Price: Contact company

Company:
The Wizards
P.O. Box 7118
The Woodlands, TX 77387

Product:
How to Make Good Investments—Computer-aided instruction for the Commodore 64. The first part in a series of courses on investment and financial analysis. Developed by professionals from top business schools.
Price: $39.95 tape

Company:
Bytes and Bits
524 East Canterbury Lane
Phoenix, AZ 85022
602-942-1475

Product:
Two business programs for the Commodore 64—*Investment Portfolio Manager* is menu-driven, providing one summary page and nine detail pages. Each page accepts nine entries of up to $99,999 each. Designed to make tracking volatile assets easy. *Disk Directory Manager,* also available for the VIC 20, reads directly from disk directories and sorts over 1400 file names, file sizes, file types and disk ID's into an ordered list. Prints a hardcopy master directory. Written entirely in machine language.
Price: Investment $14.95 tape or disk; Directory Manager $19.95 disk

Company:
Management Accountability Group, Inc.
493 East Clayton Street
P.O. Box 346
Athens, GA 30603
404-353-8090

Product:
MAGIS Comprehensive—Total instructional package for microcomputer accounting for use with Commodore business computers. Contains text book, two practice sets, teacher's manual, test bank and two fully integrated accounting programs. The student is first introduced to new accounting principles via use of a manual accounting system. Then the same principles are applied to the use of a computerized accounting system. Intended for use in high schools, vocational schools and junior colleges.
Price: Contact company

Company:
Clockwork Computers
4612 Holly Ridge Road
Rockville, MD 20853
301-924-4157

Product:
Economical point-of-sale microcomputer system for retailers—Combines use of a CBM 8032 system (or a Commodore 64 with IEEE-488 adapter), an optional computer-controlled Indiana cash drawer and *CCI Retailer*™ or *CCI Retailer 64*™ software. Software provides accurate recording of cash and credit sales and payouts. Adjusts inventory as sales are made, highlights low volume items and recommends items you need to reorder. Prints a customer sales slip and copy for retailer's sales journal. In-

CCI Retailer System

cludes sales analysis, management reports and optional word processing and mailing list as well as telecommunications capabilities.
Price: (Software only) $529 Commodore 64; $546 CBM 8032

Company:
Nanos Systems Corporation
P.O. Box 24344
Speedway, IN 46224
317-244-4078

Product:
Reference Cards—For the Commodore 64, VIC 20 and 6502 microprocessor. Fold-up accordian-style cards provide quick access to programming information. Commodore 64 version includes information on using sprites and elements of sound; charts show control codes, color codes, basic functions, hex-to-decimal conversions and more. VIC 20 version also includes summaries of functions, hex-to-decimal, etc., and documents the VIC's entire graphics set by key, CHR$ and POKE/PEEK, as well. The 6502 version contains basic programming information plus reminders about how various commands work.
Price: VIC and 64 $5.95; 6502 $4.95

Company:
Jance Associates, Inc.
P.O. Box 234
East Texas, PA 18046

Product:
Jance Computer-Controlled Home Security System—Cartridge with disk or tape software for Commodore 64 or VIC 20. Kit for do-it-yourself installation provides perimeter protection, outside and inside alarms, reminder beeper for deactivation, magnetic switches for doors and windows, panic button, automatic re-arm and programmable operations.
Price: $195.00

Company:
H & H Enterprises
5056 North 41st Street
Milwaukee, WI 53209
414-461-9941

Product:
Disk Duplicator—Allows disk back-up on 2031, 1540 or 1541 disk drives. Versions available for VIC 20, Commodore 64 and CBM 8032, 4032 and 4016. Compatible with Micro Systems Development's CIE and VIE IEEE interface cartridges. Written entirely in machine language.
Price: $14.95

Company:
RAK Electronics
P.O. Box 1585
Orange Park, FL 32067-1585
904-264-6777

Product:
RTTY II—For Commodore 64 and VIC 20 with 8K expansion. Turns your computer into a radio teletype video display terminal. Features split-screen operation (compose messages while you receive), four 255-character user definable messages and four preset messages. Select 60, 66, 75 and 100 wpm baudot speeds. With morse code it provides callsign ID, RTTY ID, auto UNSHIFT. Hardware and software manuals and I/O edge connector included. Requires a RTTY terminal unit, also available from RAK.
Price: $19.95 tape; $22.95 disk. Send for free catalog

Company:
Red-Shift Software
P.O. Box 45488
Seattle, WA 98102

Product:
Spectrum-64—Software for studying or using the Fast Fourier Transform (FFT) on the Commodore 64. Contains a multi-mode input, transform, save and high-resolution display program as well as several utilities and sample data cases.
Price: $79.95 retail; $59.95 students/professors

Company:
Analytical Software Design
220 City Boulevard West,
Suite 111
Orange, CA 92668
714-978-3111

Product:
Sima-Universal Assembler Development Package—Machine language development tool, written in machine language, that includes an advanced source code text editor. Supports multiple processors, including 6502, 65C02, 6800, 6809, 8080, 8085 and Z80, along with features like conditional assembly, built-in software RS-232 communication port, and more.
Price: $75.00 C

The Official Computer Hater's Handbook

by D. J. Arneson
Published by Dell Books
Reviewed by Bernard Falkoff

The blurb on the front cover reads, "AT LAST! A COMPUTER BOOK FOR EVERYONE WHO DOESN'T KNOW AND DOESN'T CARE HOW THE DARN THINGS WORK." But the irony is, the more you know about computers the more you'll like *The Computer Hater's Handbook.*

"Are Computers Aliens From Outer Space?" "What If Famous People and Events in History Had Computers?" and "How to Get a Hacker to Bed" are just three chapter headings but they give you a pretty strong indication as to what this book is about. Quick wit, lots of puns, cartoons and photo funnies follow one another like characters in a buffer. Occasionally, there is even a ring of truth to what is said. For instance, here's a bit of "The Fallacy of Saving Time" chapter:

> When computerists tout their clever little machines, one of the first 'advantages' they'll bore you with is that the computer is the greatest timesaver ever invented.... The fallacy is that time cannot be saved. It can only be used.... Take a good look at a clock. Have you ever seen any extra time flake off a clock? Is there a little wad of it under your wrist watch? Of course not. Because it's all used up. There isn't a single second of more time in the world now since the invention of the computer. In fact, there's almost 50 years less of it.... The next time a computerist tells you his computer saves him time, ask him to show you where he keeps it.

The chapter on the computer industry tells you about "Old Line Computer Companies". By "old line" they don't mean Commodore; they're referring to the ancients that have been "serving you since March". And for those who are fortunate enough to be working for an original equipment manufacturer but may be looking for greener pastures, the section on "How to Start Your Own Computer Company" is a very accurate scenario of how more than one defector in the industry made good. (I won't mention any names, of course.)

No book for the computer hater would be complete without ways to bring computer cocktail party conversation to a screeching halt. Some of these little tidbits include lines like, "Is 911 the prefix, or is it the whole number?" or "I understand they're thinking of painting the hydrants again." "My shorts were inspected by number 10987," and of course a real c.k., "Would you like to see my root canal?" are other suggestions.

As for cuts to the computer personae itself, anyone who has a computer is on the way to becoming a "hacker", which, for those of you who don't know, is a computer enthusiast who lives, thinks and breathes computers but eats Chinese food. You can find out the differences between Silicon Valley and Silicone Valley. You can learn the danger signs of computer addiction, how to join Computerholics Anonymous and "how to tell if your son or daughter is using computers." There's even a section on computer "burn out" featuring Boris Karloff. Of course there's also "A Day in the Life of a Hacker", "A Hacker's Photo Album" and "The Anatomy of a Hacker". A computer person can get hacked to death if they're not careful.

All in all this book is fun to pass around and read through a few times. And although the glossary of *The Computer Hater's Handbook* defines ZERO as "The sum total of everything good that can be said about computers and computerists," the more you read the more you realize that this is really not a computer hater's handbook; it's a computer hacker's handbook. C

EduFun!

Commodore Educational Software Group Announces Contracts With:

Milliken Publishing Company:

Commodore is pleased to announce that the highly regarded Milliken *Edufun* series will be available for the Commodore 64 and VIC 20 computers. These programs, originally developed under National Science Foundation grants and keyed to the essential learning skills list developed by the National Association of Supervisors of Mathematics will be available to Commodore computer owners in the near future. These programs are an important step in the process of bringing Teacher Tested Educational Software to the Home Market, allowing parents and children to use the same software at home that they use in school.

Midwest Software:

This series of over 30 programs focuses on essential early learning skills for preschoolers through early elementary grades. A simple non-threatening format is used to teach such concepts as longest/shortest, number and letter sequence, visual perception and shape matching. These programs will offer the child a significant advantage in the early learning process.

Speed Reading:

Commodore will soon be announcing the most powerful and flexible speed reading and comprehension-building program yet seen on any microcomputer. The power of this software product will allow high school students to improve critical skills necessary for college as well as provide the additional skills needed to stay ahead of their field.

Type Right:

This very popular typing program will be familiar to all our users. The program teaches the skills needed for keyboard entry and then provides drills, practice and games to test your abilities. This is a fun way to learn an essential skill while relying on time-tested methods of teaching.

Chopper Math:

Fly your chopper to the landing pad while solving math problems in addition, subtraction, multiplication and division. Very good game action is used to enhance the learning strategies used in this program. This will be an exciting addition to any home or school software library.

Minnesota Educational Computer Consortium:

Over 100 high quality programs covering most subjects and grades.

All products are due on store shelves before Christmas.

commodore COMPUTER

new books

From Sybex Computer Books
2344 Sixth Street
Berkeley, CA 94710

Your First VIC 20 Program by Rodney Zaks, $9.95.

The VIC 20 Connection by James W. Coffron, $7.95.

The Commodore 64 BASIC Handbook by Douglas Hergert, $9.95.

The Easy Guide to Your Commodore 64 by Joseph Kascmer, $7.95.

Programming the 6502 by Rodney Zaks, $14.95.

Computer Power for Your Law Office by Daniel Remer, $19.95.

International Microcomputer Dictionary, $3.95.

From dilithium Press
P. O. Box 606
Beaverton, OR 97075

Computers for Everybody, 3rd Edition by Jerry Willis and Merl Miller, $7.95.

Computers for Everybody Buyers Guide by Jerry Willis and Merl Miller, $19.95.

How to Use the Commodore 64 by Jerry Willis and Deborrah Willis, $3.95.

How to Use the Commodore VIC 20 by Jerry Willis and Deborrah Willis, $3.95.

A PET for Kids by Sharon Boren, $7.95.

A PET in the Classroom: Activity Workbook, $5.95. Teacher's manual $14.95.

More Than 32 BASIC Programs for the Commodore 64 by Tom Rugg, Phil Feldman and Gene Moore, $19.95.

More Than 32 BASIC Programs for the VIC 20 by Tom Rugg, Phil Feldman and Clarence S. Wilson, $19.95.

32 BASIC Programs for the PET Computer, $19.95

From ELCOMP Publishing, Inc.
53 Redrock Lane
Pomona, CA 91766

Tricks for VICs by Sam D. Roberts, $9.95.

More on the 64 by H.C. Wagner, $9.95.

The Great Book of Games (for the Commodore 64), $9.95.

From Celestial Software
3010 Warrington Avenue
Lakeland, FL 33803

How to Effectively Market Your Computer Software by Celestial Software, $19.95.

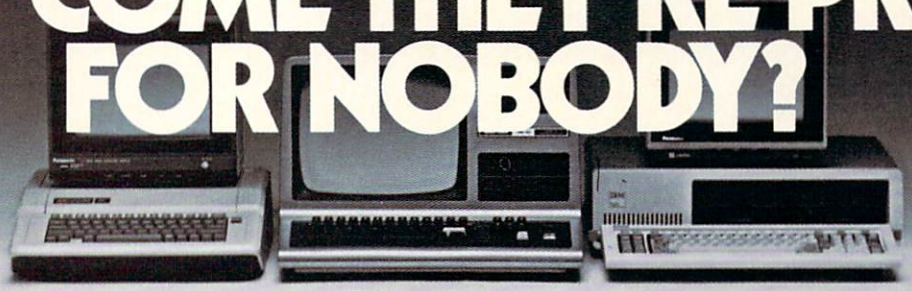
From Hayden Book Company
50 Essex Street
Rochelle Park, NJ 07662

Microcomputers Can Be Kid Stuff by Anna Mae Walsh, $8.95.

C

advertisers' index

IF PERSONAL COMPUTERS ARE FOR EVERYBODY, HOW COME THEY'RE PRICED FOR NOBODY?



$1395* APPLE® IIe 64K

$999* TRS-80® III 16K

$1355* IBM® PC 64K

A personal computer is supposed to be a computer for persons. Not just wealthy persons. Or whiz-kid persons. Or privileged persons.

But person persons.

In other words, all the persons whom Apple, IBM, and Radio Shack seem to have forgotten about (including, most likely, you).

But that's okay. Because now you can get a high-powered home computer without taking out a second mortgage on your home.

It's the Commodore 64. We're not talking about a low-priced computer that can barely retain a phone number. We're talking about a memory of 64K. Which means it can perform tasks most other home computers can't. Including some of those that cost a lot more. (Take another look at the three computers above.)

By itself, the Commodore 64 is all the computer you'll ever need. Yet, if you do want to expand its capabilities some day, you can do so by adding a full complement of Commodore peripherals. Such as disk drives. Modems. And printers.

You can also play terrific games on the Commodore 64. Many of which will be far more challenging than those you could ever play on a game machine alone.

And as great as all this sounds, what's even greater-sounding is the price. It's hundreds of dollars less than that of our nearest competitor.

So while other companies are trying to take advantage of the computer revolution, it seems to us they're really taking advantage of something else:

Their customers.

*Manufacturers' suggested list prices. Monitor included with TRS-80 III only. Commodore Business Machines–P.O. Box 500R, Conshohocken, PA 19428; Canada–3370 Pharmacy Avenue, Agincourt, Ont., Can. M1W 2K4.

commodore COMPUTERS

Apple is a registered trademark of Apple Computer, Inc.
TRS-80 is a registered trademark of Tandy Corp. IBM is a registered trademark of International Business Machines Corp.

THE COMMODORE 64™. UNDER $300.*

You can't buy a better computer at twice the price.

ISBN 0-88731-003-6