# COMMODORE WORLD

Issue 11
Volume 2, Number 6
$4.95 U.S.

THE NEWS MAGAZINE FOR COMMODORE 64 & 128 USERS

## Commodore Drive History
*The 15xx Evolution*

## A Simple Guide to Disk Drives

## Serial Bus Kernal Routines
*Access Devices in ML*

## Reviews
*Fontigus*
*Turbo Assembler*

## Plus...
*Navigating on the Net*
*Consequences of Virtual Reality*

SECOND CLASS

# SOFTWARE SUPPORT INTERNATIONAL

**We Carry America's Largest Selection of C-64/C-128 Software!**

## ENTERTAINMENT

| | | | |
|---|---|---|---|
| 10 Game Grab Bag | $9.95 | Pacman | $9.97 |
| Arkanoid 2 | $7.97 | Plundered Hearts | $12.97 |
| Beyond Dark Castle | $7.97 | Pool of Radiance | 14.97 |
| Beyond Zork 128 | $12.97 | Portal | $14.97 |
| Defender of the Crown | $9.97 | Questron 2 | $9.97 |
| Double Dragon 2 | $9.97 | Rampage | $7.97 |
| Heavy Metal | $12.97 | Realms of Darkness | $12.97 |
| Heros of the Lance | $14.97 | Roadwars | $7.97 |
| Keys to Maramon | $9.97 | Sidearms | $7.97 |
| Mean Streets | $9.97 | Steel Thunder | $9.97 |
| Monopoly | $12.97 | Strip Poker | $14.97 |
| Ms Pacman | $9.97 | Trump Castle Casino | $12.97 |
| Ogre | $9.97 | Wheel Fortune 1-2-3 | $9.97 |

## SPORTS & FLIGHT

| | | | |
|---|---|---|---|
| 4th & Inches | $7.97 | Tony LaRussa Baseball | $7.97 |
| Champshp Baseball | $7.97 | Tournament Tennis | $7.97 |
| Champn Basketball | $7.97 | WWF Wrestling | $7.97 |
| Dave Winfield Batter Up | $9.97 | Acrojet | $9.97 |
| Fast Break | $7.97 | Apache Strike | $7.97 |
| Hardball | $7.97 | Blue Angels | $7.97 |
| Jack Nicklaus Golf | $9.97 | F-14 Tomcat | $9.97 |
| Jordan vs Bird | $7.97 | F-19 Stealth Fighter | $12.97 |
| Leaderboard Golf | $9.97 | Flight Sim Games | $4.97 |
| Pro Football Facts | $9.97 | High Roller | $7.97 |
| Pro Soccer | $7.97 | Jet Combat Sim | $7.97 |
| Pure Stat Baseball | $9.97 | Skyfox | $7.97 |
| Sporting News Baseball | $9.97 | Super Huey I | $7.97 |
| Star Rank Boxing | $7.97 | Top Gunner | $9.97 |

## ACCESSORIES

| | | | |
|---|---|---|---|
| C-128 RGB Cable | $17.95 | Floppy Disk Notcher | $6.95 |
| Computer Hand 2 | $6.95 | Dust Covers - specify | $7.95 |
| Convert A Com | $24.95 | Ergostick Joystick | $16.95 |
| Disk Bank 10/3.5" | $2.95 | CBM 1200 Baud | $9.95 |
| Disk Bank 10/5.25" | $2.95 | Aprotek 2400 Baud | $49.95 |
| Disk Bank 100/3.5" | $12.95 | 1351 Smart Mouse | $44.95 |
| Disk Bank 100/5.25" | $12.95 | MW 350 Interface | $44.95 |
| Disk Bank 70/5.25" | $7.95 | 64 Power Supplies | $34.95 |
| Disk Mailers | $0.39 | 128 Power Supplies | $49.95 |
| Drive Cleaners | $6.95 | Printer Ribbons | CALL |
| Serial Cable 6ft | $8.95 | Userport Expander | $24.95 |
| Serial Cable 10 ft | $9.95 | Video Ram Upgrade | $49.95 |
| Drive Power Cable | $7.95 | 3.5" 10 cnt.Floppy | $7.95 |
| User Port Cable | $15.95 | 5.25" 20 cnt.Floppy | $4.95 |
| Com Modem Adapter | $15.95 | | |

## PRODUCTIVITY

| | | | |
|---|---|---|---|
| 1750 Super Clone | $99.95 | Geos 128 v2 | $44.97 |
| Basic Compiler 64 | $12.97 | Graphic Label Wizard | $14.97 |
| Big Blue Reader | $29.97 | Graphics Basic | $9.97 |
| B/W Prog Tools | $14.97 | Home Designer 128 | $24.97 |
| B/W Power C | $9.97 | Manager, The | $12.97 |
| B/W Turbo Cartridge | $17.97 | Maverick v5 | $24.95 |
| C128 Graphics Bundle | $29.97 | Model Diet | $9.97 |
| Christmas Model Kit | $9.97 | Newsroom | $14.97 |
| CSM Protection Man 1 | $14.97 | On Line Help | $9.97 |
| CSM Protection Man 2 | $19.97 | Outrageous Pages | $19.97 |
| Data Manager 2 | $14.97 | Postcards | $14.97 |
| Designer's Pencil | $9.97 | Printmaster Plus | $19.97 |
| Drive Align 1541/71 | $12.97 | Superbase 64 | $19.97 |
| Easy Working Tri Pak | $9.97 | Superscript 64/128 | $14.97 |
| Geos 64 v2 | $39.97 | Swiftcalc w/Sideways | $14.97 |

## EDUCATIONAL

| | | | |
|---|---|---|---|
| Early Learning Friends | $9.97 | Spellicopter | $9.97 |
| European Nations & Loc | $9.97 | Stickybear Math | $9.97 |
| Keys to Typing | $9.97 | Stickybear Numbers | $9.97 |
| Memory Manor Cart | $9.97 | Typing Tutor 4 | $9.97 |
| Magic Spells | $9.97 | Word Attack | $9.97 |
| Snoopy Sky Scramble | $9.97 | Word Spinner | $9.97 |

## REFURBISHED HARDWARE

| | | | |
|---|---|---|---|
| C-64 Keyboard w/ PS | $79.95 | 1541 Clone Drive | $39.95 |
| C-128 Keyboard w/PS | $139.95 | 40 Col Monitor | $99.95 |
| C-128D Computer | $174.95 | 80 Col Monitor | $139.95 |
| 1541 Disk Drive | $64.95 | Printers/Call for Make | $39.95 |
| 1571 Disk Drive | $129.95 | Misc. | CALL |

Items Listed Above Do Not Include Shipping. U.S 48 States - Add $5.50 per order. Alaska, Hawaii & Canada - add $5.50 for the first piece and $1.00 per each additional piece per shipment. Second Day Air shipping is available. Call for shipping charges. Call Or Write For Your Free c-64/128 Catalog Listing HUNDREDS Of Commodore Products And Special Offers For Your Computer. Our Order Takers Are On Duty 6:00 a.m. - 5:00 p.m. M - F and 7:00 a.m. - 3:00 p.m. Sat. - Pacific Time.

SOFTWARE SUPPORT

Software Support Int.
2700 N.E. Andresen Rd.
Suite D-4
Vancouver, Wa 98661
(360) 695-1393

**CALL TOLL FREE TODAY!** **1-800-356-1179**

*Major Credit Cards Accepted.*

# C O N T E N T S

ISSUE 11 • VOLUME 2 • NUMBER 6 • NOV/DEC 1995

COMMODORE WORLD
THE NEWS MAGAZINE FOR COMMODORE 64 & 128 USERS

GENERAL MANAGER
*Charles R. Christianson*

EDITOR
*Doug Cotton*

ASSISTANT EDITOR
*Jenifer Esile*

ADVERTISING SALES
*Charles A. Christianson*
*(413) 525-0023*

PHOTOGRAPHY
*Wayne Wrubel*

GRAPHIC ARTS
*Doug Cotton*
*Jenifer Esile*

ELECTRONIC PRE-PRESS & PRINTING
*Mansir/Holden*

*Cover Design by Jenifer Esile*

**See Our CHK-LIST Utility On Page 54**

# CMD/CW MARKET SURVEY

We've skipped our usual *From The Editor* column in this issue to bring you something we feel is far more important—a survey. Yes, I know. Most of us hate filling out surveys, but there's no doubt that they help companies like ours to better serve you, our customers.

Of course, this also means you're helping us stay in business, and that has value. So as a bonus for your assistance, we'll issue a $5.00 credit towards a future purchase from CMD to everyone who returns a completed, legible response to this survey.

Please send you completed survey to:

CMD/CW Market Survey
Creative Micro Designs, Inc.
P.O. Box 646
East Longmeadow, MA 01028-0646

## Demographic Information

1. Your *CW* subscription number: ____________
2. Your age: ________
3. Number of years of education: ________
4. Annual Income (optional): ________
5. Number of Children: ________
6. Children's Ages: ____________________

## General Questions

7. For how many years have you owned a computer? ________
8. What types of computers have you owned?
   A. C-64 B. C-128 C. MS-DOS
   D. Macintosh E. Amiga
   F. Other (specify) ________________
9. Which computers do you still use?
   A. C-64 B. C-128 C. MS-DOS
   D. Macintosh E. Amiga
   F. Other (specify) ________________
10. How many hours per week do you use each of the above listed computers? ________
11. What are the two most frequent uses for each of the computers listed above? (i.e. games, productivity, telecom., desktop publishing., etc). ____________________
12. How long have you owned your Commodore computer(s)? ________
13. How satisfied are you with your Commodore's ability to perform the tasks you want to use your computer for? (Use a number from 1 to 10 with 10 being most satisfied.) ________
14. What features or characteristics do you most like about your Commodore? ____________
15. What features or characteristics do you least like about your Commodore? ____________
16. How many more years do you think the Commodore will meet your needs? ________
17. Most computer users outside of this market consider the Commodore obsolete. Why do you feel differently? ____________
18. (a) If you were to buy another computer at this time, what computer platform would you consider most seriously?
   A. MS-DOS B. Macintosh C. Amiga
   D. Other (specify) ________________
   (b) Why? ____________
19. Assuming that you were making a decision whether or not to change platforms today, what additional features or new peripherals would keep you from changing? ____________
20. Is there a particular hardware add-on or software program that you would like to see for the C-64/128? (specify) ____________
21. Do you feel that existing manufacturers are producing computers that best meet the needs of the average home user? ________
22. What features do you feel are most important for an entry level computer and can you name an existing computer that offers them at a reasonable price? ____________
23. (a) Would you be interested in a new CMD computer if it had compatibility with existing Commodore software or hardware? ________
   (b) What features would you deem to be most important and what specific compatibility level would you consider satisfactory? ____________
   (c) How much would you be willing to spend on such a computer, without a monitor? ________
   (d) Would you still be interested if it required you to purchase an SVGA monitor (approx. $200 to $300)? ________

## Telecommunications

24. (a) Do you currently use your Commodore for telecommunications? ________
   (b) If so, what services do you use and/or subscribe to?
   A. Internet B. GEnie C. CompuServe
   D. Delphi E. BBS's
   F. Other (specify) ________________
   (c) What terminal program do you use? ____________
   (d) What brand and speed modem do you have? ____________
25. What additional telecommunications capabilities would you like to see available to Commodore users? ____________

## Commodore World

26. Overall, are you satisfied with the quality and content of this magazine? ________
27. Do you feel as though you gain knowledge from each issue? ________
28. Is there (a) too much, (b) not enough, or (c) enough coverage of CMD products? ________
29. What would you like to see more of? ____________
30. What would you like to see less of? ____________
31. How would you rate the artistic quality of Commodore World? (Use a number from 1 to 10, with 10 being best.) ________
32. How would you rate the editing quality of Commodore World? (Use a number from 1 to 10, with 10 being best.) ________
33. Please also forward any additional comments that you feel will help us produce a better product or publication.
____________________

# LOADSTAR SOFTWARE VALUES!

## The Compleat* Series

**The Compleat PRINT SHOP I:** (The Print Shop by Broderbund or Printmaster required). Over 1300 artistic and never before published PRINT SHOP images. The smart, fast software package included allows you to quickly scan through the many PRINT SHOP images sequentially, by name, or by group number. Press a key and save the graphic you want in 2-block, 3-block and even PRINTMASTER graphic files! All that plus a printed guide to your new sea of graphic files. And if you like the first volume, you'll probably want to get your hands on **The Compleat PRINT SHOP II**, which also contains over 1300 PRINT SHOP images, previously published on LOADSTAR issues over the past ten years. Included is the same smart, fast software package included in The Compleat PRINT SHOP I. Each volume is $20.00.

**The Compleat Programmer:** Become the Commodore programmer you've always wanted to be! Megabytes of knowledge crammed and stuffed onto eight 5.25-inch disks or two 1581 disks! Plus we include all the tools, extensions, languages, assemblers, tutorials and utilities you'll need to create the same type of software you see on LOADSTAR! This massive collection is way over two megabytes of instructional text and valuable tools. $20.00. For $5 more, get C= Hacking MAG, not available separately, to complete your programming set.

**The Compleat Maurice:** A compilation of 26 solitaire card games written by Maurice Jones, the acknowledged master of card game simulations for the C-64/128. There's even a brand new, never before published game called Boomerang. Two 5.25 inch disks or one 3.5 inch disk. $20.00 postage paid!

**The Compleat Dave:** Two 1581 disks or three 1541 disks crammed with SID music. Over 250 classic melodies from yesteryear, arranged and transcribed by the Master of Music, Dave Marquis, and now they're available in one gigantic 8-hour collection. If you enjoy SID music, you owe it to yourself to get THE COMPLEAT DAVE. Two 1581 disks or three 1541 disks $20.00 postage paid!

**The Compleat Walt:** Ten years of Walt Harned's slideshows and multimedia events gathered into one huge collection: Seven 5.25-inch disks or three 3.5 inch disks! Over 250 pictures, including some that have never been published. The greatest one-man collection of art on any computer platform! $20.00 postage paid!

**The Compleat Roger:** 25 educational quiz programs, each carefully crafted by Roger Norton, an educator who uses C-64s. These programs come crammed on two 5.25-inch 1541 disks or one 1581 disk. $20.00 postage paid!

*** No, we didn't misspell "complete." Compleat is the ten dollar spelling of complete.**

## Other Products

**Geopower Tools** - 19 Geos utilities: Calendar Printer, Fast Format, Geo Fetch (grab any portion of a screen as a Photo Scrap), Phoenix (resurrect a trashcanned file), Programmer's Calculator are just a few of the handy tools. Side Two is filled with Clip Art (in Photo Album format) and fonts. $9.95 (C-64/128) Item #080525

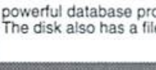
**Songsmith** - LOADSTAR's own music-making program. With this deluxe music editor/player you can easily transcribe music from sheet music or make up your own tunes. Songsmith comes with a slick 30-page manual and a jukebox player with eight tunes. $9.95

**Game Star #1** Eight games from LOADSTAR #70-#100). The Tenement, Stack 'Em, The Sherwood Open, Gems, Stealth Bomber, Eagle Eyes, Moonraker & Circuitry. $9.95

**Just For Fun** - Eight original games. There are arcade games, educational games, puzzle games and just games that are just plain fun on this disk. $9.95

**Fun Four** - Four original games. A huge maze game, trivia game, solitaire and a space shoot 'em up -- all runnable from a menu. $9.95

**Sport** -In the early days of science, "sport" meant "mutant." This is a full novel on C-64 disk by author, Jeff Jones, about a murderous neo-human. Over 500 terrifying pages. Automatic presentation software included. Printing capability. Bookmarks. Warning! This is a *real* novel with strong content, frank language, violence and adult situations. If the hard-hitting action of *real* sci-fi/horror novels turns your stomach, please don't buy this book. Must be 18. 1 CMD HD disk , 2 3.5-inch disks or 3 5.25-inch disks. $5.95 $1.00 Shipping.



**Brainpower/Brainstorm** - 80-column word processor and idea processor for the C-128. These two programs are together on one disk. It comes with a detailed 32-page manual. $9.95 (C-128 only) Item #069421

**Best Of Loadstar Compilations:** LOADSTAR is over 11 years young! We have 5 anthology disks that take you through the evolution of LOADSTAR, from humble beginnings to the well-oiled machine we've become. If you want to hop on a time machine and go back, start with **The Best Of LOADSTAR #5** and move your way back to **The Best Of LOADSTAR #1.** Each of the 136 back issues are available for purchase with discounts on large orders for you collectors out there. Best Of LOADSTAR disks are available on 5.25-inch disks only. C-64 disk. $9.95.

**Master Base** - Database of users' groups for the 80-column C-128. This is a fast, powerful database program for handling addresses and mailing labels (includes barcode printing). The disk also has a file of over 600 users' groups addresses. $4.95 (C-128 only)

**LOADSTAR is a monthly "magazine on disk" for the Commodore 64/128. Subscribers receive two 1541 disks (or one 1581 disk) in their mailbox every month filled with news, articles and programs. These non-PD, high-quality programs are written by the best home-based programmers in the field and edited by the crack LOADSTAR team of Fender Tucker and Jeff Jones. Subscription prices are at an all-time low of $69.95 for a 12-month subscription, or $19.95 for a three-month subscription. You may also elect to subscribe "by the month," where we charge your credit card $7.95 for each issue after it's shipped.**

## Loadstar Products Order form 1-800-594-3370

| Quantity | Description | Price | Item# | Total |
|---|---|---|---|---|
| | 3-month LOADSTAR subscription | $19.95 | na | |
| | 1-year subscription to LOADSTAR | $69.95 | na | |
| | Month to month @$7.95 | $7.95/mo | na | |
| | Compleat Print Shop I on 5.25-inch | $20.00 | 0009D5 | |
| | Compleat Print Shop I on 3.5-inch | $20.00 | 0001D3 | |
| | Compleat Print Shop II on 5.25-inch | $20.00 | 0010D5 | |
| | Compleat Print Shop II on 3.5-inch | $20.00 | 0002D3 | |
| | Compleat Programmer on 5.25-inch | $20.00 | 0005D5 | |
| | Compleat Programmer on 3.5-inch | $20.00 | 0005D3 | |
| | Hacking mag 3.5-inch only | $5.00 | 0006D3 | |
| | Compleat Maurice 5.25-inch | $20.00 | 0007D5 | |
| | Compleat Maurice 3.5-inch | $20.00 | 0007D3 | |
| | Compleat Dave 5.25-inch | $20.00 | 070525 | |
| | Compleat Dave 3.5-inch | $20.00 | 070523 | |
| | Compleat Walt 5.25-inch | $20.00 | 070425 | |
| | Compleat Walt 3.5-inch | $20.00 | 070423 | |
| | Compleat Roger 5.25-inch | $20.00 | 0004D5 | |
| | Compleat Roger 3.5-inch | $20.00 | 0004D3 | |
| | Geopower Tools | $9.95 | 080525 | |
| | Game Star #1 | $9.95 | 080825 | |
| | Just For Fun | $9.95 | 073525 | |
| | Fun Four | $9.95 | 073525 | |
| | Sport 5.25-inch | $5.95 | 070325 | |
| | Sport 3.5-inch | $5.95 | 070323 | |
| | Sport CMD High Density | $5.95 | 070327 | |
| TOTAL FROM THIS COLUMN | | | | |

| Qty | Description | Price | Item# | Total |
|---|---|---|---|---|
| TOTAL FROM LEFT COLUMN | | | | |
| | Songsmith | $9.95 | 069525 | |
| | Best Of LOADSTAR #5 | $9.95 | 049525 | |
| | Best Of LOADSTAR #4 | $9.95 | 049425 | |
| | Best Of LOADSTAR #3 | $9.95 | 049325 | |
| | Best Of LOADSTAR #2 | $9.95 | 049225 | |
| | Best Of LOADSTAR #1 | $9.95 | 049125 | |
| | Masterbase | $4.95 | 081025 | |
| SUBTOTAL | | | | |
| Louisiana Residents ad 4% sales tax | | | | |
| Shipping $5.50 UPS Second day air. Pay no shipping if LOADSTAR subscription or any product from the Compleat series is ordered. | | | | |
| Total Enclosed: | | | | |

**Payment method:**
❑ Check/money order made payable to "Softdisk Publishing" in US Funds
❑ MasterCard ❑ Visa ❑ American Express ❑ Discover
Card # ____________________
Expiration Date _____ / _____ / _____
Authorized signature: ____________________
Subscribers: I prefer: ❑ 5.25-inch diskettes ❑ 3.5-inch diskettes

**Mail or Fax to:**
**Softdisk Publishing**
**P.O. Box 30008, Shreveport, LA 71130-0008**
**Questions: ○ 1-318-221-8718 ○ Orders 1-800-594-3370**

# BACKTALK

## LETTERS AND QUESTIONS FROM OUR READERS

Dear CW,

Does anyone out there know a way to back up Cosmi Swift Sheet 128? Or does anyone know of a pretty good 80 column spreadsheet that's available somewhere for the 128? I have Maverick with all of the modules, and it does copy the program, but the copy will not boot up.

Thanks
T.J. Moyles

*To be honest, no-one here knew the answer, as none of us have that particular program. But we've obviously got a few thousand readers, one of whom may know. So if anyone has the answer to this question, drop us a line, and we'll reprint it for everyone to see.*

Dear Commodore World Magazine,

I read in Issue 9 of CW that geoFax is now available. Is there a bed or page scanner available for the Commodore 128 so that I can scan images and then fax them? Can I use the HandyScanner and the Pagefox to scan images into GEOS? How can I get them? Will geoFax automatically slow down the baud rate if the remote fax machine is going at the slower rate of 9600 baud instead of 14.4 K baud? What happens if someone is on the telephone when receiving a fax, will the fax get lost or can that someone just hang up the phone? And, what happens if the computer is off when receiving a fax, will the fax get lost or can you just cut on the computer?

Sincerely,
Jeffrey L. McLean

*There aren't any page or flatbed scanners supported on the Commodore just yet, but geoFAX itself can scan images using many of the modern fax machines available today. This is done by getting a fax machine that supports direct connection to a fax-modem (without hooking either into your telephone jack).*

*The Handyscanner 64 can indeed be used to scan images for placement into GEOS applications. It currently comes supplied with a utility program to convert Handyscanner files into geoPaint documents. Handyscanner and Pagefox are both available in the US from CMD.*

*Fax machines and fax-modems automatically connect with each other at whatever speed the calling machine is set for. So if someone calls you with a fax machine set at 2400 baud, your 14.4K bps fax-modem will step down to 2400 baud to match the incoming signal.*

*If someone tries to send you a fax while your phone line is in use, they'll get a busy signal. Likewise, if your computer is off, the modem won't respond correctly to the caller. So they'll just have to try again later.*

Dear Editor,

My first issue of *Commodore World*, August/ September 1995 has the type-in programs for Basic Instincts, which I wanted.

But when I typed in the CHK-LIST and ran it I was rewarded with Working....(4 lines) and "?ILLEGAL QUANTITY ERROR IN 32".

I very carefully checked every line, including the data statements, where I found a few boo-boos, several times and not only checked, but re-typed the lines 10-60 three times, but I only got the same message, above. What am I doing wrong?

Respectfully,
Roland Lowery

*The problem almost has to be in the data statements. Run the program in 64 mode (if you're using a 128), and when the error occurs, type the following:*

```
PRINTM,D
```

*You'll see two numbers on the screen. The first is an address where data was to be placed, and the second is the data itself. The first number should roughly coincide with the line numbers in the program (in most cases it will fall somewhere between the values used for two of the line numbers). Now look at the line numbers in that area of the program to see if you can find a data element that matches the second number on your screen. Once you find it, compare it to the lising in the magazine, and make the necessary corrections to your program.*

Dear Doug,

As an eight-year GEOS user and Commodore's resident PostScript guru, I have marveled at the expertise exercised in the pages of *Commodore World*. However, there are times when great articles have left out information that leaves me quite puzzled.

Sherry Freedline's article, "Spreadsheets by the Numbers," sang the praises of geoCalc, only to end that section by saying, "If only geoCalc could create graphs as well." I always thought that geoChart was supposed to create graphs. Sherry does not mention if it is fair to expect GEOS users to buy this application to supplement geoCalc. Nor does she say if it functions well when used for that purpose. She simply fails to mention it at all!

An article by Steve Vander Ark gives us "Some Tips on Using GEOPUBLISH." He states, "You get a taste of the potential that still exists in the Commodore computer when you see a document slide out of a laser printer..." I can verify his euphoria here because I have been using a laser printer with geoPublish for nearly five years. Therefore I was surprised to later read that "resizing [bimapped graphic images] is pretty much out of the question." To the contrary, GOES users who have access to a PostScript compatible laser printer at home or at KINKO'S, for example, should try to find the biggest bitmap possible for nearly all their geoPublish graphics.

PostScript offers GEOS users the unparalleled ability to increase the resolution of Commodore graphics, normally 80 x 72 dots per inch. By pasting a 4" x 4" size geoPaint image in a 2" x 2" space within a geoPublish document, you can double the active resolution of the image to 160 x 144 dots per inch. It beats the jaggies!

Steve warns us that using "oversized images" can eat up a lot of disk space, and this is true. Still, the laser printer has gotten a "bad rap" for printing graphics that look "blocky." I suggest that anyone who seriously thinks this may be a problem should subscribe to the laser printed Double-Click, a user newsletter of the East Lansing Commodore Club produced by Maurice Randall.

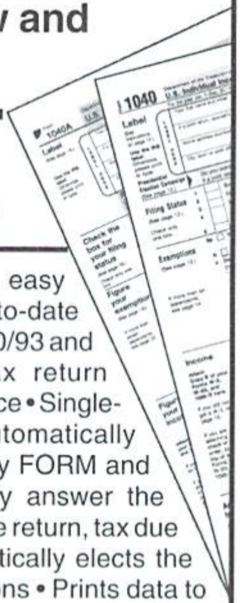
We need to keep in mind that, for those who want to use these larger images, geoPublish will allow us to downsize any image without penalty on a PostScript laser printer.

Sincerely,
K. Dale Sidebottom

*Thanks, Dale. Your comments are both welcome and appreciated. And you're certainly invited to submit detailed articles on your favorite subject to us.*

## PERFECT TAX™

**Get PERFECT TAX™ now and relax on April 15th...**



- **THIRTEENTH SUCCESSFUL YEAR! • MANY THOUSANDS ALREADY IN USE!**
- **Your Simple Answer to the TAX Reform Acts of 1986 through 1994**

- **Stores data files on disk for quick recall**
- **Yearly upgrades available to registered users**
- **Fully screen-prompted, menu-driven and easy-to-use**
- **Includes helpful tutorial example and forms library**
- **Built-in calculator accumulates input and enters totals**
- **Instant recalculation of your entire return when you change any item with just a single keystroke**
- **Prints directly onto IRS forms and Schedules or to blank computer paper for use with transparent overlays**
- **PERFECT TAX is Tax Deductible**

**Supports all of these FORMS and Schedules:**
FORMS: 1040, 1040A, 2441, 3903, 6251, 8615 and 8815
Schedules: A, B, C, D, E, SE, 1, 2, 3 and EIC-A
**TAXPERFECT 128 additionally supports:**
FORMS 2106, 2119, 2210, 3800, 4136, 4256, 4562, 4684, 4952 and 8396, plus Schedules EIC, F and R

• PERFECT TAX is the quick, correct, easy way to do your taxes • Absolutely up-to-date with the Tax Reform Act of 1986/87/88/90/93 and ***all new*** tax changes • Simplest tax return preparation program available - at any price • Single-keystroke form-to-form change • Automatically calculates and transfers data from every FORM and Schedule to the FORM 1040 • Simply answer the questions - PERFECT TAX calculates the return, tax due or amount of refund due you • Automatically elects the greater of Standard or Itemized deductions • Prints data to all FORMS or Schedules you need for your complete return - ready to sign and file • Highly acclaimed by the tax pros, CPA's and tax preparers, PERFECT TAX is easy to understand and a pleasure to work with • Separate versions available for Commodore 64 and Commodore 128 computers

| | |
|---|---|
| PERFECT TAX 64 | **$69.00** |
| PERFECT TAX 128 | **$79.00** |

*(plus shipping—see our main ad)*

Available from
Creative Micro Designs, Inc.
**1-800-638-3263**

**FOR THE BEST SELECTION OF**
## REFURBISHED COMMODORE EQUIPMENT
**CALL 1-800-638-3263**

**JiffyDOS** pre-installed on every computer & disk drive!

**30 DAY WARRANTY** on all refurbished equipment

| | |
|---|---|
| **COMPUTERS** | |
| C64 or C64c (refurbished) | $89.00 |
| C128 (refurbished) | $149.00 |
| 128-D (refurbished) | $239.00 |
| **MONITORS** | |
| 1702 (refurbished) | $99.00 |
| 1802 (refurbished) | $129.00 |
| 80-column monitors | CALL |
| **DRIVES** | |
| 1541 (refurbished) | $75.00 |
| 1541-II (refurbished) ***SPECIAL!*** | **$89.00** |
| 1571 (refurbished) ***SPECIAL!*** | **$99.00** |

Prices do not include shipping charges, and are subject to change without notice. All items subject to availability, call before ordering.

**SPECIAL!** 1541-II Drive **$89.00**

**SPECIAL!** 1571 Drive **$99.00**

**Creative Micro Designs, Inc.**

# COMMODORE TRIVIA

*by Jim Brain*

Welcome to another edition of Commodore Trivia. As many of you may know, these trivia questions and answers have been donated by me to the Commodore community at large. Unlike other articles in Commodore World, these trivia questions have been placed in the public domain. I ask only that the trivia questions remain intact and unchanged, and that my name and address appear somewhere so users can contact me. The trivia is also used for a contest I run on the Internet; contact me at the included address for more information. Because curiosity has the best of me, I always welcome a note or postcard detailing where the trivia goes. I always welcome new questions—provided they come with answers. Enjoy.

Jim Brain
Brain Innovations, Inc.
602 North Lemen
Fenton, MI 48430
brain@mail.msen.com

## COMMODORE TRIVIA #10 QUESTIONS

$090 The 6502 has a rich history. It is modeled after another 8-bit microprocessor. Name the processor.

$091 The 6502 has a older brother that was never produced. Name its number designation and why it was not produced.

$092 How many different opcodes are considered valid and "legal" on the MOS NMOS 6502 line?

$093 Every instruction takes at least __ cycles to complete. Fill in the missing number.

$094 Which instructions take more time than necessary as a result of the answer to $093?

$095 What did MOS Technologies manufacture before introducing the 650X line of microprocessors?

$096 Three companies manufactured the 6502 under a cross-licensing agreement. Name them.

$097 In NTSC-land, how fast does the 1MHz 6510 in the C64 actually run?

$098 What about in PAL-land?

$099 Data is latched into the 650X microprocessor on the (rising/falling) edge?

$09A Through the years, the 650X line has changed family numbers, yet the part has not been changed. (A family number is the upper 2 digits in this case) Name the other family numbers used by MOS to denote the 650X line.

$09B Consider the following code:

```
ldx #10
lda $ff,x
```

What location does the accumulator get loaded with?

$09C What about the following?

```
ldx #10
lda ($ff),x
```

$09D How many CPU clock signal lines does the 650X require to run?

$09E Where does the 650X line fetch its first byte from after reset?

$09F One of the original designers on the NMOS 6502 CPU now heads up Western Design Center in Arizona, and makes the 65C02 and 65C816 CPU chips. Name him. (Hint: it's not Chuck Peddle!)

## COMMODORE TRIVIA #9 ANSWERS

$080 The magazines were originally called *Commodore Microcomputers* and *Power/Play: Commodore Home Computing*. They never did seem to nail down the name of the latter as I see *Power/Play* and *Commodore: Power/Play* used as the original names as well. Anyway, *Commodore Microcomputers* started its life in 1979, whereas *Power/Play* started in 1981. Both magazines were published until around 1987, when they were merged to form *Commodore Magazine*. Then, around 1990, the magazine was sold to IDG Communications and was merged into *RUN*. *RUN* was continued for a while, but was finally pulled out of circulation. Creative Micro Designs purchased the rights to the magazine, and now *Commodore World* is being produced by CMD. I am not sure how strong (if any) a link there is between *RUN* and CW, but some of the same authors write for the new publication. Just for added info, here are the ISSN numbers:

| | |
|---|---|
| Commodore Microcomputers (Commodore Magazine) | 0744-8724 |
| Power/Play:Commodore Home Computing | 0739-8018 |
| RUN (Commodore/RUN) | 0741-4285 |

"The Transactor" is also a correct answer, and info on it is below.

$081 The infamous "Tarnsactor". One of the noted C64 hardware-hacking magazines, it was originally published by Commodore Canada, before being sold to an individual named Mr. Hilden. Its ISSN number is 0838-0163. As far as I can tell, this magazine died many deaths, but officilly ceased to exist in 1989-90. Its first issue is dated April 30, 1978.

$082 No! The newer 128 compatible chip (VIC-IIe) has 8 extra pins to perform timing functions specific for the 128. In addition, some of the registers have extra functions. However, a suitable card to make it compatible can be made.

$083 Phase Alternating Line is the answer I was looking for, which describes the video encoding used in Europe, but Programmable Array Logic is also correct, which describes the family of chips used as "glue" logic for the C64 I/O and processing chips.

$084 5: Play, Rewind, Fast-Forward, Record, and Stop/Eject. Later models separated the stop and eject functions into two buttons.

$085 When you change the volume of a voice. The voice need not be outputting anything.

$086 Take your pick:
Control Program/Monitor
Control Program for Microprocessors
Control Program for Microcomputers.
The last one is considered by many to be most correct.

$087 Normally, the user cannot enter a line number higher than 63999. If you want to be tricky, however, the numbers can be made to go up to 65535.

$088 The PI symbol. It is [SHFT-UPARROW] in uppercase mode, but becomes a checkerboard-like character when in lower-case mode. Unlike the graphics characters printed on the fronts of the keys, this one is positioned in the middle of the keycap, and should probably be accessible in both character sets.

$089 In lowercase mode, type a shift-@

$08A It is different from the 64/128. It is 50003. A zero (0) here indicates old ROMs, while a one (1) indicates new ROMs.

$08B Interrupt ReQuest. This interrupt is used for things that should usually be allowed to interrupt the processor. This interrupt can be masked off by the SEI instruction.

$08C Non-Maskable Interrupt. Unlike the IRQ, this interrupt cannot be masked by an instruction. However, some tricks can be used to mask it.

$08D 'N' stands for Negative. On instructions that change this flag, it is set to be equal to bit 7 of the result of the instruction.

$08E It stands for decimal mode. This mode causes certain instructions to treat a byte as 2 4 bit BCD-coded nybbles.

$08F pR is the way to abbreviate PRINT#. Note that ?# will fail.

## COMMODORE TRIVIA CORRECTIONS

It's a shame that trivia answers are not trivial (pun intention undecided) to fix. I try my best to both present complete and correct answers for the trivia and also correct errors that crop up as soon as I can. So, here is a compilation of corrections for the Commodore Trivia:

Q $000 Wouldn't you know it, I messed up on the first question. Although the answer stated is technically correct, here is a much better explanation:

Q $000 Commodore started out into computing with the PET series of computers. I am not sure if the first ones had the PET emblem, but nonetheless, what does P E T stand for?

A $000 Personal Electronic Transactor
It seems this name was an afterthought, so many other expansions can also qualify. It was basically named PET to cash in on the Pet Rock craze. Some examples of other expansions:

Personal Electronic Translator
Peddle's Electronic Transactor
Peddle's Ego Trip

This is where the Commodore magazine Transactor got its name.

Q $009 What is the difference between the printers in #$008?

A $009 MPS 802 (Serial), CBM 1526 (Serial), PET 4023 (IEEE-488).

(After presenting this answer, a number of people indicated that they had printers with dot patterns different from my answer. My only explanation is that, since the print heads on the printers were interchangeable, Commodore took advantage of that to keep stock low.)

Q $01F Commodore did NOT document the RREG command in C128 Manuals, so this question is technically correct, but almost everyone considers the C128D to be the same as the C128. The revised answer follows:

Q $01F On the Commodore 128, the user manual left three commands undocumented.One works, and the others give a not-implemented error. Name the commands and what each one does or does not do.

A $01F The answer depends on which manuals you have. In the C128 System Manual, the C128D System Manual, and the C128 Programmer's Reference Guide, the following commands are not documented and both return an unimplemented command error:

OFF It is, however, valid when used with the KEY command, as in KEY OFF.
QUIT

The third command is actually implemented, yet is not documented in the C128 System Guide: RREG: reads the internal registers after a SYS command. On page 326 of the C128D System Guide: "RREG [a[,[x][,[y][,status]]]]
This function returns the contents of the computer's internal registers after a SYS command. The contents of a, x, y, and processor status registers are assigned to the variable list.

```
EXAMPLE:
10 SYS DEC("FF59"),8 calls kernel routine "LKUPLA"
20 RREG A, X, Y, S
30 IF S AND I THEN PRINT "NOT FOUND": END
40 PRINT "FOUND:"; A; X; Y
```

The above example calls a standard kernel routine to check if a given logical channel is in use (in this case, logical file 8). If it isn't, "NOT FOUND" is printed. If it is in use, "FOUND" is printed, and the logical file number, device number, and secondary address of the channel is printed. So, this command is documented in the C128D System Guide. It is interesting to note that the command does not appear in the BASIC 7.0 encyclopedia in the C128 PRG, but does appear in the command list immediately following the encyclopedia. It seems that Commodore either forgot about the command when introducing the C128, and remembered it when publishing the C128D System Guide and the C128 PRG, or decided to leave it out and then later broke down and included it in later manuals.

Q $03E On every Commodore disk, the drive stores a copy of the BAM. What does the BAM stand for?

A $03E The correct answer is Block Availability Map. Somehow, I got BAM partially confused with the FAT (File Allocation Table) on MS-DOS.

Q $04C How many pins does a Commodore 1525 printhead have in it?

A $04C The 1525 (and 1515) platen actually has 18 bumps on it, which allow the single pin print head to construct the 7 bit tall graphics.

Q $060 When you turn on stock Commodore 16, how many bytes free does it report?

A $060 I went back and checked on the RAM in the Commodore 16 (when I finally unearthed it at the house), and found that it does indeed have 12277 bytes free, as the answer stated.

Q $06E Although the answer is correct, the wait 6502,x operation will only print out the secret message on Revision 2 ROMs. Evidently, Commodore found this message and removed it in Revision #3 ROMs.

To date, these are the only corrections I have been made aware of. However, don't think I am actually this close to perfection. Commodore World's installments of the trivia are 11 editions or so behind the latest edition (which can be found on-line), so CW readers never see some of my more ridiculous mistakes. I greatly appreciate the thousands of Internet users, FIDONet users, and magazine and newsletter readers that check over my answers for corrections.

# ON THE HORIZON

Commodore and Computer Industry News

### If Your Product Name is too Perfect

We've been recently informed by Financial Services Marketing Corp. that their Federal income tax preparation software, formerly known as TAXPERFECT, will now be marketed under the name *PERFECT TAX™*. The change comes at the end of nearly seven years of litigation over the name with industry giant WordPerfect (recently obtained by Novell).

Details of the litigation and subsequent name change will be published at a later date, as a final ruling on an appeal brought before the Fifth Circuit Court is still pending.

### Another Commodore Publication Passes Away

Sadly, we have to report that Lynn-Carthy Industries, Inc. has finally notified us that they no longer will attempt to resurrect their publication, *dieHard*. It's now been a year since LCII shipped their last issue of this publication. While many of us remained hopeful that *dieHard* would somehow manage to overcome their problems, official notification that it would not be financially possible for the publication to continue reached our offices last month.

In addition to the announcement, LCII approached *Commodore World* and other Commodore-related publications in hopes of striking a deal to fulfill remaining *dieHard* subscriptions. Due to the large losses this would incur, the opportunity was declined by each of the publishers. We have learned, however, that *LoadStar* has agreed to substitute their disk-based publication to subscribers of *dieHard's* Spinner disk, providing subscribers with a two-disk issue for every two disks remaining on their subscription.

### New Desterm & Browser Character Sets Announced

Gaelyne Moranec has made available several collections of new character sets for use with Matthew Desmonds's *Desterm* terminal program and Rod Gasson's *Browser* utility. Over 200 character sets in all are included, and are available via FTP on *ccnga.uwaterloo.ca* in the */pub/cbm/telecomm* directory. The filenames to be on the lookout for are: *fullset.sfx*, *ibmset.sfx*, *amiset.sfx*, *cbmset.sfx*, *cbmcgset.sfx*, *vt52.sfx*, *vt102.sfx* and *cbmchrs.sfx* (the latter is only for use with *Browser*).

The character sets in the *fullset.sfx* collection include complete ASCII character translations. The *cbmchr.sfx* collection of character sets for Browser are an adaption of sets used by *QWKRR128*. The original character sets were 5 blocks long and unsuitable for use with *Browser*, since *Browser* uses 9 block character sets. These sets also needed major adjustments so that Commodore graphics would be shown correctly when used with *Browser*.

### Threshold Keeps Them Coming

Announcement of the release of two more new games has arrived from *Threshold Productions*.

The first, "*Flummi's World*", is a very nice platform type game where you must help guide a rolling bowl from its starting point to an ending point somewhere on the screen. There are little beasties which get in your way and bonuses that let you jump higher. You'll build brick steps, walls, and many other things on your way to the end of this 30 level game. Retail price is listed at US$19.95.

The second release, the "*Gangster/Time Traveller*" 'shoot 'em up pack', provides two games with nicely done graphics and one intent: Kill the enemy! In "*Gangster*" you take on the role of a police officer who must try to stop the gangsters from breaking out of prison, escaping in automobiles, and from taking hostages in a building. In "*Time Traveller*", you become an army recruit who warps through time, stopping to shoot down bi-planes, helicopters and aliens. Retail price is US$14.95.

*Threshold Productions, 17730 15th NE Suite #229, Seattle WA 98155 (internet email: tpinfo@eskimo.com).*

### Color 64 Goes Freeware!

*Color 64 V 7.37*, the last version of the popular C-64 BBS software authored by Greg Pfountz, has been released by the author to the Commodore market as freeware.

This BBS software, with 12 years of work behind it, was declared freely redistributable on September 16, 1995. *Color 64 V8*, by Fred Ogle, remains a commercial product.

*Color 64* Freeware is being supported by two BBSes in the US. Timothy Allen's *Twilight Zone* in Mesa, AZ, can be reached at 602-827-2706, and Richard Cunningham's *Desert Oasis* in Phoenix, AZ, can be reached at 602-849-2892.

*Color64 V7.37* can be obtained via anonymous FTP at the site *indirect.com* in the *www/wanderer* directory. It may also be downloaded from the World Wide Web at *http.//www.indirect.com/www/wanderer/color.htm*.

For more information, contact *wanderer@indirect.com* or the support BBSes listed above.

### CommNet Looks To Expand

Michael Bendure, author of the *C-Net DS2 Networking* system and coordinator of *CommNet*, recently announced the 64/128 BBS network's plan to expand to support virtually every popular Commodore BBS system with networking capabilities.

At present, *C-Net 64 DS2 V2.0,2.5*, and *3.0*, *Image Vl.2a*, *C-Net128 v6.0* systems are supported by the 70-site network. In the interim, *CommNet* works by joining the 5 networks supported by these BBS systems into one through the use of specialized gateways, but there is work in progress to develop a set of network standards to be used on all supported systems.

Sysops of *C*Base 64*, *Color 64*, *Color 128*, *Omni 128* and other network-capable BBSes are invited to contact the *CommNet* organizers to provide input on the project, and to submit their network packet system for incorporation into the final structure.

Interested sysops should contact Michael Bendure at 614-788-8568 (voice), 614-522-6563 (CygnusX-IBBS) or *mbendure@infinet.com* (e-mail).

### Commodore Internet Connection

Daniel Dallmann has recently announced the availability of an Internet "demo" which he has written and released. The program is apparently capable of TCP/IP and SLIP, protocols used by systems connected directly connected to the Internet service providers. While not a complete application, the demo program displays that our Commodore 64's are indeed capable of direct internet connection.

The demo program is available for download via anonymous FTP at *ftp://131.188.190.131/pub/c64*, and requires a modified user port RS-232 interface. Details of the interface can be found in the Commodore FAQ, maintained by Jim Brain. To obtain a copy of the FAQ, send email to: *brain@mail.msen.com* with a subject of *MAILSERVE* (all uppercase),and the following three lines (all lowercase) as the body of the message:

```
send faq.p
help
quit
```

### GE Seeks Buyer For GEnie Online Service

Rumors circulated early in November that General Electric was looking for a buyer for its GEnie online service. According to Doug Wolford, a spokesman for GE, the rumors appeared to have been started by a journalist writing in the *Washington Post*, surmising that the logical step is to sell its online service. Four days after Wolford's comments appeared in a Newsbytes News Network story, a follow-up story reported that GEnie had informed its information providers, content providers, and development managers, as well as all of its employees, that the online service was indeed for sale. Horace Martin, VP, business development and sourcing and acting president, GEnie Online Services, confirmed that GE Information Services is working with the investment banking firm of Allen & Company to "identify potential buyers for the company's GEnie online service."

### AOL Passes 4 Million Members

Okay, so normally we wouldn't talk about an online network that only supports the Macintosh and MS-DOS platforms in our news, but since America OnLine, Inc. once operated the Commodore-only Q-Link, we thought we'd let everyone know how they're doing now that they've left for "greener pastures." Citing a recent Odyssey market study, AOL reports that it has passed the four million member mark, and is now as large as CompuServe and Prodigy combined. Ironically, the company also released its first quarter results around the same time, which showed a 250% increase in revenues, but at net loss of $10,262,000. With such a high loss after the substantial gains in revenue, we can only guess that they must have kept the same management that ran Q-Link into the ground during its last couple of years of operation.

### Commodore CEE GEOS Products

COMMODORE CEE has taken over distribution of several items for GEOS users from their distributors. A few items are new or upgrades; some are items that have not been available for quite a while.

Among the products are *TOPDESK*, *DweezilDISK1* (*NewTools2*, *Marker*, *CreatPatt*, *PattDA*, *AutoPatta*, & over 30 GEOS fill patterns), *DweezilDISK2* (*ULTIPATT*, *geoGLOBE Collection*), *DweezilDISK3.5* (*DweezilLABEL28*, *MYgeoDIARY*, *geoWORDS*), *BIG STAMP*, *GEOS PROGRAMMER'S REFERENCE GUIDE* (by Alex Boyce—not the official version from Berkeley Softworks), *GEOPROGRAMMER HELP*, *geoJOURNAL MAGAZINE COMPENDIUM VOLUME 1*, and *geoJOURNAL MAGAZINE COMPENDIUM VOLUME 2*.

For additional information on these and other Commodore Cee products, contact: *Jack Vander White, COMMODORE CEE, P.O. Box 232115, Sacramento, CA 95823* (email: *jack.vanderwhite@cee-64.gigo.com*).

### Accelerator Update

There have been a couple of changes in the design of the new *Super64 CPU* accelerator series reported on in our last issue. The 65C02S processor has been replaced with the 65C816S, a 16-bit version of the 6502 which also contains an 8-bit emulation mode. This design change occured just as we went to press on Issue 10, and we were unable to get the new information into the news before going to press.

In addition, CMD has announced that the accelerators will now be equipped with 128K of fast static RAM, instead of the 64K previously advertised. This change will allow the operating system to be downloaded into the fast RAM during startup, thus avoiding any slowdowns when calling Kernal or BASIC routines. Even the fastest ROM chips available couldn't keep up with the fast processor without adding delays, and fast ROM chips proved to be just as expensive.

Several other important design changes have been implemented, but details have not been cleared for public release at this time. CMD has, however, agreed to provide a sneak preview of the prototype for our next issue. Stay tuned...

### ERROR CORRECTIONS

Issue 10, page 4: There was a typographical error in the email address provided for Al anger. The address should have read:

d04066c@dcfreenet.seflin.lib.fl.us

Issue 10, page 25: There was an error in the STATUS BYTE VALUES given near the bottom right corner of the Serial Bus Device Kernal Routine Reference Chart. The value shown for Device Not Present is $FF, but should have been $80.

# Just For Starters

*by Steve Vander Ark*

## A LITTLE ABOUT PRINTER SELECTION & MORE SIMPLE BASIC

When I bought my Commodore 64 over ten years ago, that was all I bought. No monitor, disk drive, or tape drive. I didn't even buy joysticks; I used ones from my Atari 2600. Over the next year or two I added to my hardware, starting with a disk drive. It was years before I finally unhooked my 64 from a portable TV and attached it to a monitor. Along the line, I bought a used printer. I wasn't sure I had any use for it, but I got it cheap and figured it might come in handy someday.

I don't remember who I bought that Okimate printer from, with its waxy printouts and the horrendous paper feed. I do remember what happened to the way I used my computer. Suddenly, I was a publisher. I created cards, letters, and posters using a program on a cartridge called Magic Desk and one on a disk called Print Shop. Everything came out looking so professional and clean! My computer was becoming more than a game machine; it was becoming a tool for my creativity. Not long after that, I bought my first copy of GEOS and between geoPaint and geoWrite, I could do anything I wanted. This kind of power was beyond anything I imagined when I bought that lone Commodore 64 a few years before.

Today you would never buy a computer without a disk drive. Besides the Commodore, I doubt there is a computer around without a drive built in. You won't be able to use a computer these days without a monitor. And more and more, you simply don't buy a computer without a printer. Computers are no longer just game machines; they're productivity tools or in the words of a radio commentator I heard, "information appliances." These types of uses require a printer. Because of this, the prices of printers are coming down below anything I could have dreamed of when I bought my Okimate. Back then, a laser printer, if you could find such a thing in a catalog, cost thousands. The dot matrix printer was just making inroads into the market, slowly replacing the daisy wheel from back in the Stone Age of typewriters. And there was no such thing as an ink jet printer. The printer you may be looking to buy now will be different from the ones I had to choose from, and not just in price. Here are some choices you'll have as you consider buying a printer for your 64 or 128.

The snazziest kind of printer you can get is the laser printer. It will create the neatest printouts of any printer and will do it a lot faster than any other type. However, most Commodore programs don't have printer drivers to handle a laser printer in its high resolution mode. Printer drivers are little programs that your software uses to know how to talk to your printer. For most Commodore programs, the drivers are built right in; you just have to go into some kind of set-up area of the program to specify which kind of printer you are using. That's where the trouble comes from; most Commodore programs were written quite a few years ago, before anyone could have imagined that regular folks like you and me might own a laser printer. Since there is no way to add drivers to one of these programs, you're out of luck. A few programs have separate printer drivers, however. GEOS is a good example. As long as someone is around to write new drivers when new printers appear on the market, you're okay. There are a number of good laser printer drivers for GEOS. If you don't use GEOS, you can still use a laser printer if you choose carefully. Some laser printers will emulate an Epson FX-80; pretty much the standard for dot matrix printers. But if you're just going to make it act like a dot matrix, a laser printer is probably more than you need when you consider the relatively high cost per page.

In the last few years, a new type of printer has become popular: the ink jet printer. It creates printouts which rival those of a laser printer in quality. The prices for these printers are very reasonable, even for color. Since they're a new kind of printer, many Commodore programs don't support them directly, but they will work with the standard Epson FX-80 drivers. These printers are very quiet and handle paper in single sheets, although the cost per page is a bit steep. It is an excellent choice for the Commodore user.

The standard is still the dot matrix. If you buy one which is Epson compatible, it will work with almost any software using the Epson FX-80 printer driver. In fact, that Epson FX-80 is the printer all the rest try to act like. You can get a 9-pin or 24-pin model, the difference being that the 24-pin has better printouts in non-graphic modes. The dot matrix printers are usually inexpensive, include a large number of powerful features and handle paper quite nicely.

The fear that many users have, particularly those new to computing, is that they will make a wrong choice when it comes to hardware like this. When it comes to printers, if it isn't Epson compatible, it stands a good chance of not working with your software. Don't buy a printer because someone is selling it cheap; buy one because it will do what you need it to. If possible, buy a new printer instead of used. And don't forget to buy an interface, which is what you need to connect your Commodore to a standard printer. Interfaces are available from CMD, among other places.

### Back to BASIC

What do you want to do? Which choice do you want to make? Over the last few columns I've talked about using commands for branching or changing the program flow. We looked at GOTO, a command which makes the program jump to a new line, using IF and THEN to make choices, and about creating a good way for users to interact with our program. This time, we'll set up a routine which many programmers have used over the years that works nicely and is pretty much fool-proof. We'll also look at another powerful branching command.

Let's start with out input routine. I'll lay it out here with some notes.

```
1Ø REM INPUT ROUTINE
```

(That REM at the beginning of the line makes it a "REMark," which is ignored by the program)

```
2Ø PRINT "PLEASE CHOOSE YOUR FAVORITE PET
FROM THIS MENU"
3Ø PRINT "1 - CAT"
4Ø PRINT "2 - DOG"
5Ø PRINT "3 - FISH"
```

(These last few lines will print a nice menu on the screen, offering three numbered choices)

```
6Ø INPUT "ENTER 1, 2, OR 3 AND PRESS
RETURN";A$
```

(Whichever number the user presses will be assigned to the variable we are calling A$)

```
7Ø IF A$="1" THEN GOTO 2ØØ
8Ø IF A$="2" THEN GOTO 3ØØ
9Ø IF A$="3" THEN GOTO 4ØØ
1ØØ PRINT"PLEASE MAKE ANOTHER CHOICE"
11Ø GOTO 6Ø
```

(The reason for these last two lines is that someone might enter some number or letter other than 1, 2, or 3. If that happens, the program won't find a match with any of our IF...THEN statements in lines 70 through 90. Line 100 will suggest they make another choice and line

## Refurbished Units & Repair Service !

| | |
|---|---|
| C-64 Refurb. with P/S (No books) | $ 54.95 |
| C-64 Refurb. without P/S (No books) | 46.95 |
| 1541 Refurb. with cord, serial cable (No books/disks) | 64.95 |
| Monitor Cable, 5-pin DIN to RCA jacks | 4.95 |
| Serial Cable, 6-pin DIN, 6' | 8.95 |
| Commodore Power Supply for C-64 (used, tested) | 7.95 |
| Power Cord for 1541 | 3.95 |
| Power Supply for 1541-II | 19.95 |

**Printers/Monitors:** We have a limited supply of C-64 compatible printers and monitors. Write for current list & prices.
**Ribbons:** We stock ribbons for most Commodore and C-64 compatible printers. Write for current list & prices.

Service: We will repair or replace:

| | |
|---|---|
| C-64 (not including P/S) | $ 35.00 |
| 1541 | $ 45.00 |

Service price includes parts and labor. Send computer or disk drive with name, address, phone #, & describe problem. Repair or refurbished replacement will be shipped UPS.
(Please remove any non-Commodore upgrades)

Commodore Repairs
Since 1979
• • • • •
90 Day Warranty
on all repairs
and refurbished items.
• • • • •
Write for
unadvertised specials!

TO ORDER: Send check or money order for the total of your order plus $6.00 shipping (Personal checks take 2 weeks to clear). For COD orders, add $5.00. We currently do not accept credit cards.

**Unique Services** (413) 496-9275
68 Velma Avenue, Pittsfield, MA 01201
All items subject to availability. Prices subject to change.
MA residents add 5% sales tax (except for service)

## Commodore Chips and Parts

NEW PRICES >>> <<< NEW PRODUCTS

**Upgrade Chips**

| | |
|---|---|
| 6526, 6567,6569, 6581, 6522, 8502, 8562, 8500, 8563, 8564, 8721, 8722, 325302, 390059, 251968 | $8.95 |
| 901225, 901226, 901227 | $6.95 |
| 251715, 251913, 8580,WD1772 | $10.95 |
| 314972-03(C128 ROM Upgrade) | $24.95 |

**Motherboards**

| | |
|---|---|
| 1541 | $14.95 |
| 1541II | $19.00 |
| 1571 | $19.50 |
| C-64 | $34.95 |
| C-128 | $49.95 |
| C-128-D | $52.50 |
| 1700 REU(512K) Board Only | $17.95 |

**Floppy Disk Drives**

| | |
|---|---|
| 1541 | $64.50 |
| 1541II w/ P.S | $74.95 |
| 1571 | $89.50 |
| 1581 | $69.95 |

**Diagnostics & Manuals**

| | |
|---|---|
| Commodore Diagnostician: Guide to diagnose & fix C64/128/1541 | $6.95 |
| CBM C64/128 Dead Test Cartridge and Manual | $23.50 |
| C128 Diagnostic Cartridge and manual | $24.75 |
| Manuals: MPS 802 & 1230, 1084, 1902, 1581, 1526, 1571, 801, C64, C128 | Call |

**Miscellaneous**

| | |
|---|---|
| 1351 Commodore Mouse | $19.95 |
| Computer Saver (C-64 protection system) | $14.95 |
| Printer Port Adapter (Any CBM Printer to PC) | $24.95 |
| Monitor Cables | Call |
| Used Monitors (Low Prices) | Call |
| 1541 Alignment Kit | $15.95 |
| SX-64 (2 left in stock) | $125.00 |
| C65 Computer (65/1581 drive) | $169.00 |
| 1230 CBM Printer(Citizen120) | $39.95 |
| New CBM 3.5"DD disks (50ct) | $7.95 |
| MPS1000 Ribbons ( min. 3 boxes) | $3.00/box |
| C-64 Keyboard | $14.95 |
| C-128 Keyboard (limited Qty.) | $27.50 |
| C-128D Keyboard | $24.95 |
| SX-64 Keyboard (w/ door) | $59.95 |

**Power Supplies**

| | |
|---|---|
| C-64 non-repairable | $15.95 |
| C-64 repairable | $17.95 |
| C-64 Heavy Duty (5.2 amps) | $39.95 |
| C-128 Heavy Duty (5.2 Amps) | $39.95 |
| 1750 5.2 amps | $39.95 |
| 1541II external 110 volts | $9.95 |
| 1581 external 110 volts | $9.95 |
| 1084S Flyback (Phillips) | $35.50 |
| 1084-D1 Flyback | $35.95 |
| 1084 Flyback (Daewoo) | $35.95 |
| Str54051 for 1802C Monitor | $12.95 |

**Paxtron** CORPORATION
28 Grove Street, Spring Valley, NY 10977
914-578-6522 • ORDERS 800-595-5534 • FAX 914-624-3239
Hours: 9-5 pm EST • Add $5.00 UPS Charges • MC/Visa

*continued from previous page*

110 will jump them back to the point in the program where we originally asked for input.)

```
200 REM THIS IS WHERE THE PROGRAM GOES IF THE
USER CHOOSES 1 FOR CAT
210 PRINT:PRINT:PRINT:PRINT
```

(This line is really three commands, all strung together. You can do that with BASIC if you separate the commands with the colon. I try not to do this too much because it makes the program more difficult to follow, but for something like this it works well. All this line does is make the next print command happen a few lines lower on the screen.)

```
220 PRINT "MEOW"
230 GOTO 20
```

(This will send the program back to the beginning so the user can make another choice.)

```
300 REM THIS IS WHERE THE PROGRAM GOES IF THE
USER CHOOSES 2 FOR DOG
210 PRINT:PRINT:PRINT:PRINT
320 PRINT "ARF"
330 GOTO 20
400 REM THIS IS WHERE THE PROGRAM GOES IF THE
USER CHOOSES 3 FOR FISH
410 PRINT:PRINT:PRINT:PRINT
420 PRINT "GLUB GLUB"
430 GOTO 20
```

Notice how each set of commands was pretty much the same, except for the actual words we wanted printed on the screen. First, we sent a few PRINTs to the screen to drop down three lines, put a short message on the screen, then jumped back to the beginning. There is another command we can use to let us reuse some of our code more than once. We just have to send along a snippet of text each time to let it know what to print on the screen.

First we'll create our generic set of commands for responding to our user's input:

```
200 REM GENERIC RESPONSE
210 PRINT:PRINT:PRINT
220 PRINT R$
```

(We'll have to set R$ to be the correct response text somewhere back in the program. More on that in a moment)

```
230 RETURN
```

That RETURN is important. We're not just going to get to this routine with a GOTO this time. We're going to use the command GOSUB instead. GOSUB stands for "GO to SUBroutine," which is a proper computer term for that small section of BASIC commands we just wrote. A GOSUB jumps to a new line number just like a GOTO, but a GOSUB also has built into it a way to get back to wherever it jumped off. That's what the RETURN tells it to do. The nifty thing about RETURN is that it doesn't have its line number specified in advance. That means that you can use the same subroutine in several spots in your program and it will jump back to whichever spot it was called from that particular time.

Here's how we'll rewrite our program to use the GOSUB command:

```
10 REM INPUT ROUTINE USING GOSUB
20 PRINT "PLEASE CHOOSE YOUR FAVORITE PET
FROM THIS MENU"
30 PRINT "1 - CAT"
40 PRINT "2 - DOG"
50 PRINT "3 - FISH"
60 INPUT "ENTER 1, 2, OR 3";A$
70 IF A$="1" THEN R$="MEOW"
```

(The text variable gets set right here. When we get to the subroutine, the variable R$ will be all set.)

```
80 IF A$="2" THEN R$="ARF"
90 IF A$="3" THEN R$="GLUB GLUB"
100 PRINT"PLEASE MAKE ANOTHER CHOICE"
110 GOTO 60
120 GOSUB 200
```

(This line sends the program off to the subroutine starting with line 200. When the RETURN command comes up in line 230, the program will jump back to the line after this one, line 130.)

```
130 INPUT "QUIT? (Y/N)";S$
```

(We're going to give the user a chance to exit the program here)

```
140 IF S$="Y" THEN END
150 GOTO 10
```

(Note that we're only testing for a "Y," which would mean that the user wants to stop. No matter what else is entered, the program from here goes straight back to the beginning)

Our programs are getting more complicated. But if you've been paying attention as we learned new commands and techniques, you should be able to follow along quite easily. There is a consistent logic to programming which comes from the fact that computers "think" in extremely literal and logical terms. That doesn't mean that our programs have to look stiff and plain. Next time we'll look at some ways of sprucing things up on the screen.

# Foreign Exchange

*By Joseph Gaudl*

## WORKING WITH GODOT

In the last issue of *CW*, I introduced you to the best graphic program ever available for the Commodore 64/128: GoDot. If you just briefly scanned that article ("Waiting for Godot"), I would like to suggest that you get the issue and read the article through. In this article, we will take a quick walk through the basics of the program and explore the possibilities GoDot offers.

The program is easy enough to start. Just load the file "GODOT" as you usually do and wait a minute (or a few seconds if you're using a RAMLink!) as GoDot loads it's menus and modules. The main screen is the starting menu screen and is divided into six windows, which are anchored in their place. In each of the windows you will find labeled buttons, which are called gadgets. To activate a gadget, simply click on its button. The two top windows are the "Install" and "Command" gadgets. These are used to load or save the graphic type using the corresponding module, redisplay the last rendered graphic, and to exit the program. In using these loaders and savers, you can convert different types of graphic formats. The "Image Information" window contains the largest gadget in the main menu, the "Preview" gadget. This provides you with a small thumbnail image of the graphic you have loaded and are working with. Next to the "Preview" window you will find all the necessary information about the graphic; its name, its source memory format, the resolution of the four bit data and the type of the data (color or gray).

The "Screen Controls" window controls the rendering process of the current graphic. The

Exec Area determines the section of the graphic to be displayed (Full or Clip), and with Colors you can adjust the number of rendered colors (2 to 16). There is an additional button which controls the graphic resolution of the C64 (Hires/Multi).

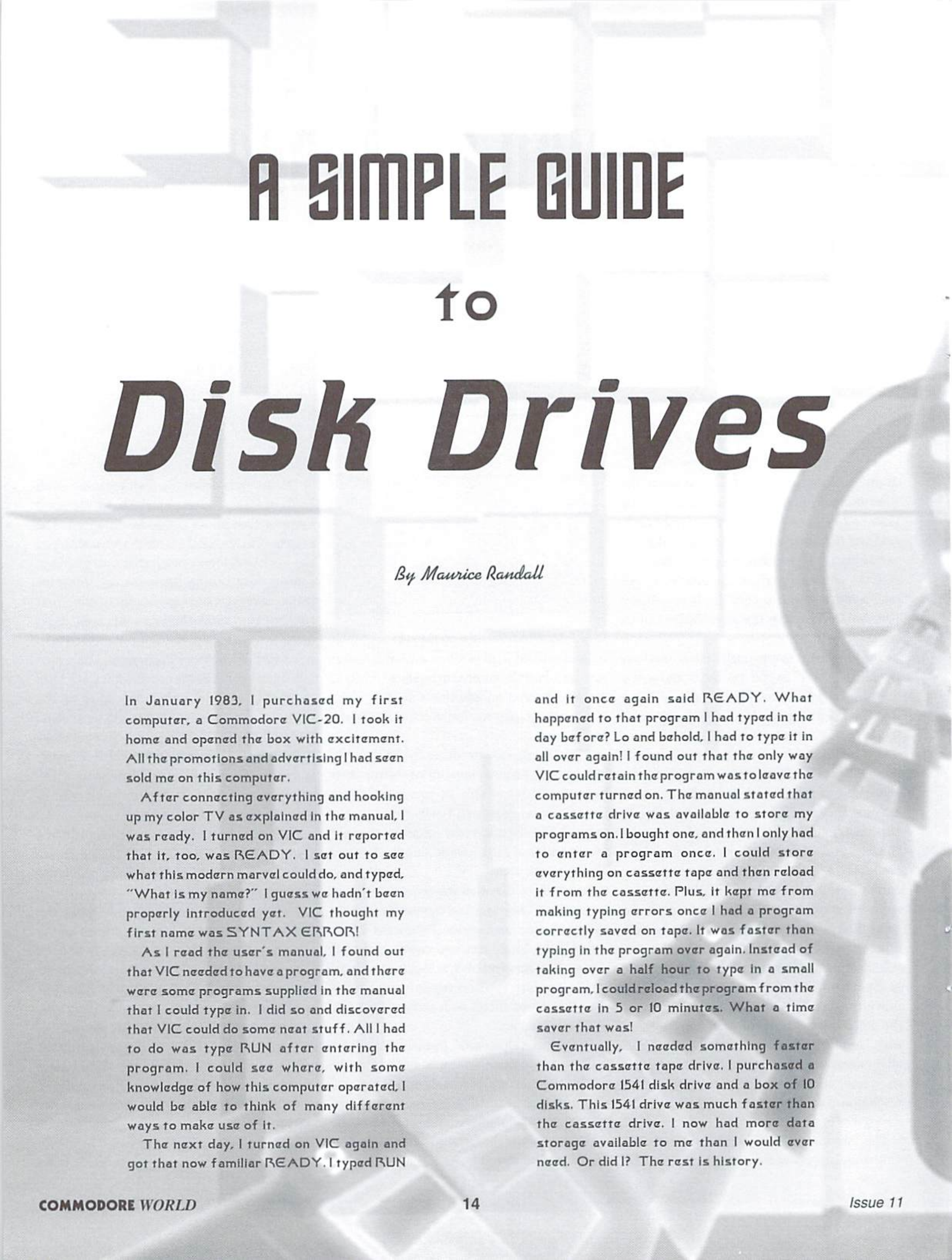
"Color Controls" introduces the Palette with its Requester. Requesters in GoDot open up other gadgets, which can only be accessed through their respective requester. The Palette serves the purpose of collecting a new color arrangement for current and future work. Display lets you view the color changes which your graphic has taken on. If you like what you see, Accept them. If you don't, Undo'em! You can use the Balance gadget to enter an additional requester which controls the brightness and contrast of your picture. This is a very sensitive tool and produces excellent results. You can even dither with it and alter the screen effects with ordered, pattern, noise and random.

The "Image Operators" install and activate GoDot's many modules. I was going to count and list them all until I saw how many there are. To take a look at them all for yourself, simply open up the file requester and get ready to be amazed! To active the installed module, click on the Execute button.

The program disk is filled with graphics in all kinds of formats, some of which are the finest I've ever seen on a Commodore monitor. The graphic of a clown and of a hawk are classics! You can work with these or with your own graphics, convert them, overlay them with each other for really cool effects and experiment to your hearts content! The German manual contains 99 pages of super instructions, but someone will have to translate this handbook in ordere for US users to take advantage of all the possibilities GoDot has to offer. One of the authors has taken on the preliminary translation, but it will need some revamping before being released. Graphic freaks could order the program directly from Germany and shouldn't experience problems getting it to run, but waiting for the English handbook has definite advantages in this case.

Almost every month new modules and drivers are written for GoDot. The most recent drivers are for color bubble jet printers. The results are astonishing to say the least.

CMD loaned a RAMLink to Arndt Dettke, the author of GoDot so that he could work on the necessary device drivers and basically test the program with state of the art hardware. When Arndt called our European office with his test results, we could hear his heart pounding over the phone! He never expected his program to deliver such speed. This was the way GoDot was meant to fly! A US distributor should start negotiations with the German authors and pick up GoDot ASAP! It's a shame that only European users have access to this fine program.



# A SIMPLE GUIDE to *Disk Drives*

*By Maurice Randall*

In January 1983, I purchased my first computer, a Commodore VIC-20. I took it home and opened the box with excitement. All the promotions and advertising I had seen sold me on this computer.

After connecting everything and hooking up my color TV as explained in the manual, I was ready. I turned on VIC and it reported that it, too, was READY. I set out to see what this modern marvel could do, and typed, "What is my name?" I guess we hadn't been properly introduced yet. VIC thought my first name was SYNTAX ERROR!

As I read the user's manual, I found out that VIC needed to have a program, and there were some programs supplied in the manual that I could type in. I did so and discovered that VIC could do some neat stuff. All I had to do was type RUN after entering the program. I could see where, with some knowledge of how this computer operated, I would be able to think of many different ways to make use of it.

The next day, I turned on VIC again and got that now familiar READY. I typed RUN and it once again said READY. What happened to that program I had typed in the day before? Lo and behold, I had to type it in all over again! I found out that the only way VIC could retain the program was to leave the computer turned on. The manual stated that a cassette drive was available to store my programs on. I bought one, and then I only had to enter a program once. I could store everything on cassette tape and then reload it from the cassette. Plus, it kept me from making typing errors once I had a program correctly saved on tape. It was faster than typing in the program over again. Instead of taking over a half hour to type in a small program, I could reload the program from the cassette in 5 or 10 minutes. What a time saver that was!

Eventually, I needed something faster than the cassette tape drive. I purchased a Commodore 1541 disk drive and a box of 10 disks. This 1541 drive was much faster than the cassette drive. I now had more data storage available to me than I would ever need. Or did I? The rest is history.

## The Most Famous Disk Drive Of All Time

This 1541 disk drive from Commodore has been loved by millions, and no doubt has been hated by just as many. No matter what, the 1541 format was and still is the standard used by software companies to distribute their work. Every Commodore user needs to have at least one of these drives. Most devoted users have at least two.

Even though my first 1541 seemed new to me, it was not new to the industry. It had been around for some time, and even the first 1541's were a product of time. They were descendants of previous disk drives that Commodore produced and used with the PET/CBM series of computers. The 1541, though bulky looking today, was rather compact in size compared to it's predecessors.

The 1541 continued a tradition of Commodore in that it has it's own operating system. Inside the

1541 is a self-contained computer that can be programmed to do many special things. With it's own Disk Operating System (known as DOS), the 1541 can receive a command through the serial port from the 64 or 128 (or the VIC-20). If the command is valid, the 1541 will report back to the computer with the results of the command, otherwise it will report an error. There are many commands the 1541's DOS recognizes and through these commands or a combination of them, the 1541 can be instructed to do things that would normally require a considerable amount of overhead on the computer's part.

Upon plugging in a 1541 and turning it on, it's ready to go without any complicated installation on the users part. Other computer platforms have a new term for this, they call it 'Plug N Play'. Commodore marketing never gave this feature a name, it's just the way we've always done it!

Since the 1541 contains its own operating system as well as a small amount of RAM, a program can be written to reside and run within the drive. This is normally how a copy-protected disk is accessed. Various schemes have been employed in order to fool the software pirates, and most of those schemes take advantage of the drive's programmable features. But by the same token, the pirates also use the drive's capabilities in order to defeat the copy-protection schemes.

Perhaps the most useful programmability of the 1541 has been in the area of fast loaders; routines that are loaded into both the drive and the computer in order to speed up the communication on the serial port. GEOS operates this way. If this were not possible, then GEOS would never have gained the popularity that it has, due to it's extensive amount of disk accessing.

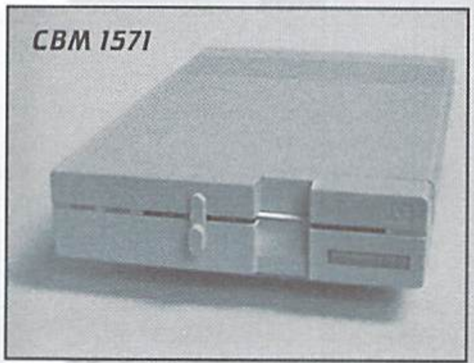
In the early days, single-sided disks were easy to find. This is the type of disk that was used in the 1541. But most of the IBM drives used a double-sided disk. The IBM drives had two heads, thus allowing both sides of a double-sided disk to be accessible. It wasn't long before 1541 users realized they could use double-sided disks. All they had to do was flip the disk upside down and cut a notch for the write-protect detector. One disk could now take the place of two disks. Double-sided disks became cheaper, and single-sided disks soon became extinct. IBM users were buying double-sided 5.25 disks in large quantities, and so were Commodore users.

## The Most Under-Rated Drive

The 1571 drive came along when Commodore released the new 128. Not only did Commodore design and build a remarkable computer, they had an equally remarkable disk drive to go along with it. Now those double-sided disks could be formatted and used without having to flip them over, just like the big boys.

Now we have a single-sided drive for the 64 and a double-sided drive for the 128. But Commodore also gave the 128 a unique feature. It can perfectly emulate the 64. Not to be outdone, the designers of the 1571 decided to make the drive emulate the 1541. The owner of the new C-128/1571 combo could have the best of both worlds.

To top it off, since the 128 had the ability to boot up in CP/M mode, Commodore added some



additional capabilities to the 1571. It could read from and write to most disk formats from other CP/M machines. It could even use an MS-DOS formatted disk that was intended for those double-sided IBM drives.

Remember how fast the 1541 seemed in comparison to the cassette drive? The 1571 seemed like another step up in performance when used on the 128. The C-128/1571 combo was designed to operate at a much faster serial port speed than the C-64/1541 counterpart. Commodore saw the need for the faster disk access and it was indeed welcome. Unfortunately, when the 128 was operated in 64 mode, the 1571 had to slow down to the speed of the 1541. Supposedly, compatibility was the reason.

The 1571 never really caught on big. Most software companies still released their software in the 1541 format because the 1571 was able to read the single-sided format with no problem. It wouldn't matter which drive was being used, only one disk format was needed. As advanced as the 1571 was, the Commodore user could get by without ever having to own one. It's a shame; there was no other drive like it on any platform.

### Disks Are Getting Smaller

Just about the time you would think you needed a bigger drive to hold bigger disks, the disks and the drives get smaller! Along came the 1581. It was smaller and so were the disks. But those handy little 3.5 inch plastic-cased disks actually hold more data than even the larger 5.25 inch double-sided disks.

Commodore was in the Amiga business and the Amiga's standard drive was a 3.5 inch unit that was built into the machine. Since Commodore already had some 3.5 inch mechanisms lying around, someone got the idea of modifying the DOS from the 1571 to work with it. The result was a nice little drive that could hold more than double the 1571.

The IBM community was beginning to see the value in 3.5 inch drives also. The 5.25 inch drives were beginning to fade in popularity on that platform. The 1581 was also given the ability to read and write to IBM formatted disks. It's native format, however, was specific to the Commodore. It could connect to the serial port just like the other drives and communicated just as fast as the 1571. To the user, it seemed a little faster because the actual disk access within the drive was quicker.

CMD FD-2000

Oddly, the 1581 was never packaged along with a computer. When Commodore released the 128D, it had a built-in 1571 drive. The 1581 had to be purchased separately. Many serious Commodore users bought one, but it's suprising how many users have never owned one. That has to be blamed on the software companies. Original software was still released in the 1541 format.

The 1581 uses a disk that is considered to be a double-density disk. Probably in the early days of the 3.5 inch disk, there was not any mention of its density. But the IBM users now needed more data storage, so along came the high-density disks. That's where Commodore stopped...

### And CMD Continued

Commodore must have made a bunch of 1581's, because long after production stopped, they were still shipping units. Even after Commodore closed its doors, new units could still be found. Someone in engineering no doubt thought of producing a high-density version of the 1581. But there were just too many 1581's sitting in the warehouse. I think the pencil pushers won. The 1591 never came to be (I think that's what it would have been called, it was the only 15x1 number left!) It would have been a good drive to have.

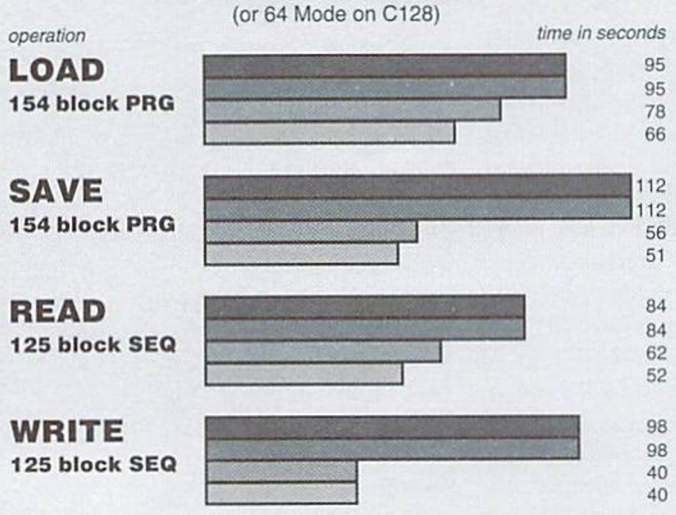
Thanks to Creative Micro Designs, we have the 1591 after all. But it's called something else, and it is also much more than the 1591 ever would have been. It's the FD-2000. This drive can plug into the serial port just like any of the Commodore drives and will do just about anything a Commodore user needs a drive to do.

### Disk Drive Speed Comparison

Because disk drives are mechanical devices, differences in the mechanisms used can make a big difference in their performance—especially when comparing 5.25 inch mechanisms to 3.5 inch mechanisms. Other factors in speed include efficiency of the DOS (Disk Operating System) and serial bus communication methods. The Commodore 128 uses a fast serial transfer protocol (in 128 mode) that has a profound effect on all the devices tested—except the 1541, which has no built-in fast serial routines.

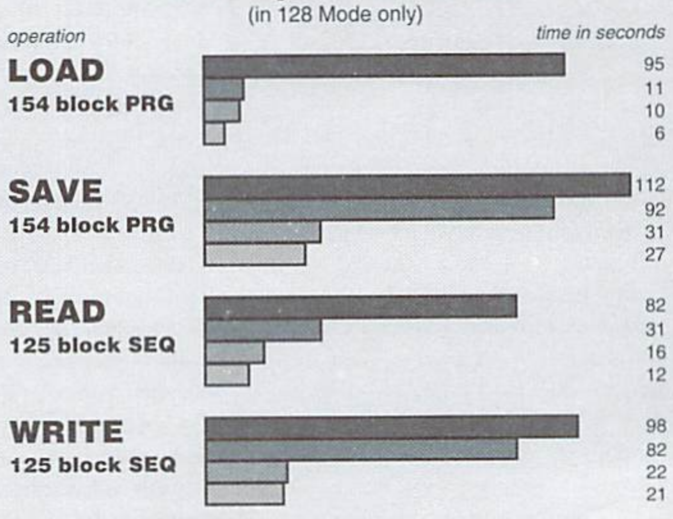
#### Drive Speed On C64

(or 64 Mode on C128)



#### Drive Speed On C128

(in 128 Mode only)



The beauty of it is that it can read and write to a 1581 disk.

The FD Series drive is actually offered in two different models. The other one is known as the FD-4000. The main difference is in the drive mechanism that is used. The FD-4000 can not only use the double-density disks like the 1581 uses along with the high-density disks as the FD-2000 uses, but it can also use an enhanced-density, or ED disk. The ED disk can hold twice as much data as the HD disk and the HD disk holds twice that of the DD disk.

The FD Series drives are really at the top of the heap in the computer industry. They are the most remarkable and versatile drives ever made for any computer platform. A disk can be formatted and partitioned in much the same way. An individual partition on an FD formatted disk can simulate any of the popular Commodore disk drive formats. You can put 1541, 1571, or 1581 partitions on the disks, or you can use CMD's proprietary native mode partitions. Of course, you can have multiple partitions on each disk, allowing you to have a mixture of different partition types. The only limitation is on the amount of disk space available.

One example of using the FD drive would be to take an inexpensive high-density disk and put 9 1541 partitions on it. You could then take the whole disk copier that CMD supplies, known as "MCOPY", and copy nine of your non copy-protected 1541 disks to it. The disk could either be used for backup purposes, or you could use it for your actual work disk and put your original 1541 disks away for safe keeping. If you have several disks that you use on a regular basis, you would now only need one disk.

## The Pattern Ends

You may have noticed that the IBM industry has been setting the stage and controlling the types of disks we use. We've gone from single-sided to double-sided disks. Then we started using 3.5 inch disks. When CMD introduced the FD drives, we could use the high-density 3.5 inch disks. The IBM users keep wanting more, and somehow we are able to follow along and keep up with them. You would think they would also want the ED disks. After all, they hold twice as much data as the HD disks. On an IBM machine, an HD disk will only hold 1.44 megs. Some of the files used on an IBM can be quite large, and the data files that can be created can be huge. So, it would seem that the ED disk format would be popular. This just wasn't the case. Software manufacturers steered towards the CD-ROMs for their original software. If they supply software on disk, it's in high-density format. If they need more than one disk, they do so. I don't know if the users aren't interested in the ED format, if it's the computer manufacturers reluctance to equip new machines with the mechanisms, or if software companies are setting the standard. Whatever it is, floppy drives are out and CD-ROMS and huge hard drives are in. Most IBM users have one floppy disk drive; many Commodore users have three, four, or even more.

What this has done is made the ED format scarce. It never took off; the IBM industry sets the standard. The companies that manufacture the ED mechanisms have all but ceased production of the units. That means that the FD-4000 is in very limited supply and the ED disks are still much higher priced than the HD disks. The FD-2000 does not suffer from any shortage problem. It will be around for a long time to come.

I think we've seen the end of the disk drive pattern between computer platforms. To a Commodore user, a high-density drive can hold a tremendous amount of data. Since our programs and data files tend to be smaller than a typical file on another computer platform, we are perfectly happy with a high-density disk. But we also have the luxury of the ED disk in an FD-4000, at least for those who have already acquired one or are planning to. Just think, a single floppy disk that can hold four 1581 partitions. It sure is nice.

What's really amazing is that many people are still very content with their 1541 drives. You would think the 1541 would be too old and primitive today. The 1541 grew out of it's predecessors, and likewise, every drive since then has been built on the same ideas, only better. It would be hard to top the FD Series drives, and any serious Commodore user should think about owning one. But you still need one of those old beloved 1541's!

## Disk Drive Specifications & Capacities

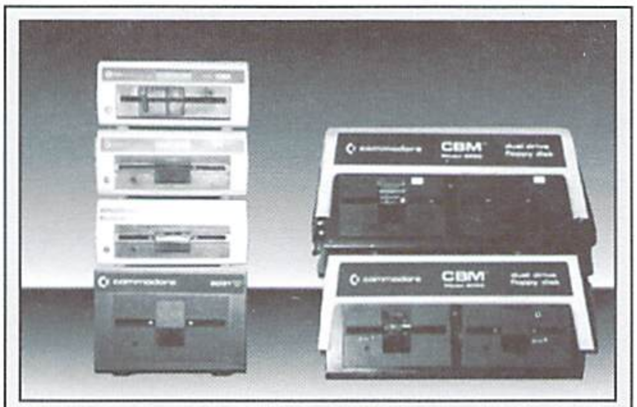
| | 1541 | 1571 | 1581 | FD-2000 | FD-4000 |
|---|---|---|---|---|---|
| Controller Type | GCR | GCR/MFM | MFM | MFM | MFM |
| Physical Media Size (inches) | 5.25 | 5.25 | 3.5 | 3.5 | 3.5 |
| Media Type | SSDD | DSDD | DSDD | DSDD/HD | DSDD/HD/ED |
| Rotation Speed (RPM) | 300 | 300 | 300 | 300 | 300 |
| Heads (disk sides) | 1 | 2 | 2 | 2 | 2 |
| Cylinders | 35 | 35 | 80 | 80 | 80 |
| Logical Blocks/Cylinder | 17-21 | 34-42 | 40 | 80 | 160 |
| Logical Block Size (bytes) | 256 | 256 | 256 | 256 | 256 |
| Physical Sectors/Cylinder | 17-21 | 34-42 | 20 | 20 | 20/40 |
| Physical Sector Size (bytes) | 256 | 256 | 512 | 512/1024 | 512/1024 |
| Formatted Capacity (bytes) | 174,848 | 349,696 | 819,200 | 1,638,400 | 3,276,800 |
| Maximum Blocks | 683 | 1,366 | 3,200 | 6,400 | 12,800 |
| Maximum Free Blocks | 664 | 1,328 | 3,160 | 6,336 | 12,736 |
| Overhead Blocks | 19 | 38 | 40 | 64 | 64 |
| Maximum Files/Directory | 144 | 144 | 296 | 5,658 | 11,347 |
| Maximum DOS Directories | 1 | 1 | 27 | 2,830 | 5,674 |
| DOS ID | 2A | 2A | 3D | 1H | 1H |
| DOS Buffers | 5 | 5 | 8 | 32 | 32 |
| Device Number Switching | N/A | 8-11 | 8-11 | 8-15 | 8-15 |
| Bus Protocols | IEC | IEC/CFS | IEC/CFS | IEC/CFS/JD | IEC/CFS/JD |
| Microprocessor | 6502 | 6502A | 6502A | 65C02 | 65C02 |
| Clock Speed | 1 MHz | 2 MHz | 2 MHz | 2 MHz | 4 MHz |
| RAM | 2K | 2K | 8K | 32K | 32K |
| ROM | 16K | 32K | 32K | 32K | 32K |
| MS-DOS Format Support | N/A | 360K | 720K | 720K/1.44M | 720K/1.44M 2.88M |
| Options | N/A | N/A | N/A | Real-Time Clock | Real-Time Clock |

# Commodore Drives are HISTORY

*by Jim Brain*
*pictures supplied by George Page*

Commodore has had a long history, producing a wide variety of peripherals for their popular computer systems. In this article, we'll focus on the disk drives making up Commodore's 15xx series, mainly intended for use with the VIC-20, C-64, and C-128 computers. I hope you'll enjoy finding out about the various models and how they came about.

At this time, I'd like to precede this article with a brief disclaimer concerning the information presented here. I have tried to the best of my ability to present the most accurate information pertaining to the history of Commodore and its products. Most of this information has been confirmed, but some of the details either have not been, or cannot be confirmed at this time. Where such information is used in this article, I've taken special care in noting it as such. If you have any information that can either confirm or disprove the statements presented here, please forward these to me by whatever means possible.


*Left stack, top to bottom: SFD-1001, 2031LP, 2031, 2031 (full-height). Right stack, top to bottom: 8250, 8050.*

**In The Beginning...**

Let us step back in time. Commodore produced disk drives almost since the time they introduced their first computer system, the Personal Electronic Transactor or PET. Since those first computers were aimed at business environments, Commodore produced disk drives to satisfy the need to store and retrieve large amounts of data in an easily accessible and economical fashion. Many of those will be covered at a later date, in a separate article on PET/CBM Drives.

In the early 1980's, Jack Tramiel of Commodore decided that the world was ready for a "computer for the masses" and introduced the Commodore VIC-20, one of the so called "home computers". Jack estimated that typical home computer buyers would purchase a cassette tape drive unit for mass storage. One might say that Jack estimated wrong. Soon after the VIC was introduced, customers started asking about the disk storage device that Commodore was "planning" at the time.

Caught somewhat off guard, Commodore raced to deliver on the promise of an inexpensive disk drive unit for the Commodore VIC-20. The result was the Commodore VIC-1540 Floppy Disk Drive. The 1540 was the father of the very successful Commodore 15xx disk drive line. However, the 1540 drive was itself a descendant of another Commodore drive, which is where we begin our tour.

The Commodore 15xx drive line can trace its roots back to the 2040 dual floppy disk drive, developed for the PET/CBM line of computer systems. Although we won't go into much of the detail of the 2040 in this article, it was a dual drive unit that was enclosed in a heavy sheet metal case, and contained "full height" drive mechanisms. The 2040 is one of the earliest drives that Commodore produced, and it was mainly used with the early PET/CBM line for computers.

**The PET/CBM 2041—A Myth?**

Research suggests that the 15xx series of drives was not created directly from the 2040 disk drive, however, but that an intermediate drive existed between the two. This drive is called the Commodore 2041 disk drive. Although no confirmed reports of this drive's existence have surfaced, Commodore sources note that this drive was actually the unit that became the 1540. We do know that the Commodore 2041 disk drive was pre-announced and was slated to be available in mid-1979. Evidently development was temporarily scrapped, only to be rejuvenated when the demand for a disk drive for the Commodore VIC-20 swelled. The development of the 2041 disk drive was then evidently resumed, albeit with one important change: the unit would be called the VIC-1540 floppy disk drive.

**The VIC-1540**

The 1540 unit utilizes 5.25" single-sided, double-density magnetic floppy disks for information storage and can store 168,656 bytes of user data per disk. For recording data onto the floppy disk, the Commodore-proprietary Group Code Recording (GCR) system was used. Although earlier Commodore drives were dual drives and used full height meachisms enclosed in metal cases, the 1540 uses a half height drive mechanism developed by ALPS Electric Company, LTD., and is housed in a durable plastic case measuring 97mm high by 200mm wide by 374mm deep. The case is off-white in color, and the affixed decal contains simplistic white lettering on a brown background. In addition, the VIC-1540 drive differs in its interface to the computer system. Earlier drive units used the standard IEEE-488 parallel bus to communicate with the computer system, but the 1540 broke with that tradition and implemented a new serial bus (also called IEC bus) interface. Why the drive uses this bus is an interesting story in itself. A noted Commodore authority, Jim Butterfield, related the details of this change to me:

"As you know, the first Commodore computers used the IEEE bus to connect to peripherals such as disk and printer. I understand that these [cables for the IEEE-488 bus] were available only from one source: Belden cables. A couple of years into Commodore's computer career, Belden went out of stock on such cables (military contract? who knows?). In any case, Commodore was in quite a fix: they made computers and disk drives, but couldn't hook 'em together! So [Jack] Tramiel issued the order: 'On our next computer, get off that bus. Make it a cable anyone can manufacture.' And so, starting with the VIC- 20 the serial bus was born. It was intended to be just as fast as the IEEE- 488 it replaced."

The story might have ended there, except anyone who has ever used a 15xx series drive knows they can be as slow as molasses when loading data. How could this be? Jim Butterfield explains:

"Technically, the idea was sound: the 6522 VIA chip in the VIC-20 and the 1540 has a 'shift register' circuit that, if tickled with the right signals (data and clock) will cheerfully collect 8 bits of data without any help from the CPU. At that time, it would signal that it had a byte to be collected, and the processor would do so, using an automatic handshake built into the 6522 to trigger the next incoming byte. Things worked in a similar way outgoing from the computer, too. However, we early PET/CBM freaks knew from playing music using the 6522 chip that there was something wrong with the 6522's shift register: it interfered with other functions. The rule was 'turn off the music before you do anything else'."

"The Commodore engineers, who only made the chip, didn't know this until they got into final checkout of the VIC-20. By this time, the VIC-20 board was in manufacture. A new chip could be designed in a few months (yes, the silicon guys had application notes about the problem, long since), but it was TOO LATE!"

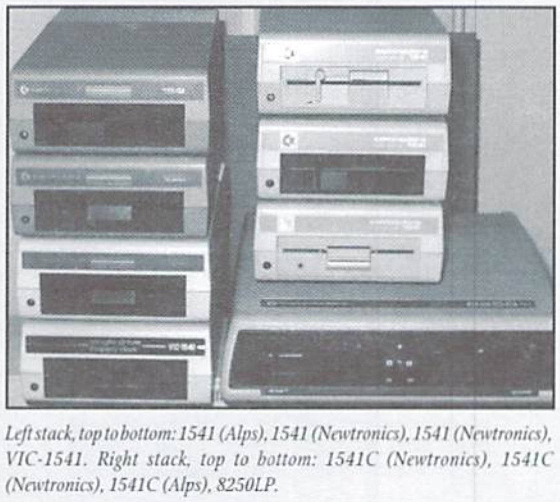
"A major software rewrite had to take place that changed the VIC-20 (and the 1540 disk drive) into a 'bit-catcher' rather than a 'character-catcher'. It called for eight times as much work on the part of the CPU; and unlike the shift register plan, there was no timing/handshake slack time. The whole thing slowed down by a factor of approximately 5 to 6 [times]."

So, Commodore started out the drive series with a significant speed handicap, which we will see them perpetuate through many models in this series.

To create the disk operating system for the VIC-1540 drive, Commodore used its PET/CBM drive DOS code as a base and striped out the support for a second drive unit. In addition, the IEEE-488 communications code was removed and replaced with support for the new Commodore serial bus. With these changes, Commodore created a new DOS revision that would become the base for most of the 15xx line: version 2.6.

## The VIC/CBM 1541

The most popular (and most common) drive in the Commodore 15xx series is undoubtedly the 1541, which was produced in a number of permutations over the years. I'll attempt to briefly explain the differences between the various versions.


*Left stack, top to bottom: 1541 (Alps), 1541 (Newtronics), 1541 (Newtronics), VIC-1541. Right stack, top to bottom: 1541C (Newtronics), 1541C (Newtronics), 1541C (Alps), 8250LP.*

The 1541 disk drive unit was introduced with the Commodore 64 computer system. The first version of this drive, the original 1541, was physically equivalent to a 1540, and differs only in the DOS code used inside the unit. Again, the version is 2.6, but changes are apparent. Let us again turn to Jim Butterfield for explaination:

"When the Commodore 64 came out, the problem VIA 6522 chip in the VIC-20 had been replaced by the CIA 6526. This did not have the shift register problem which had caused trouble on the VIC-20, and at that time it would have been possible to restore plan 1, a fast serial bus. Note that this would have called for a redesign of the 1540 disk drive, which also used a VIA. As best I can estimate—and an article in the IEEE Spectrum magazine supports this—the matter was discussed within Commodore, and it was decided that VIC-20 compatibility was more important than disk speed. Perhaps the prospect of a 1541 redesign was an important part of the decision, since current inventories needed to be taken into account. But to keep the Commodore 64 as a 'bit-banger', a new problem arose."

"The higher-resolution screen of the 64 (as compared to the VIC-20) could not be supported without stopping the CPU every once in a while. To be exact: every 8 screen raster lines (each line of text), the CPU had to be put into a WAIT condition for 42 microseconds, so as to allow the next line of screen text and color nybbles to be swept into the chip.(More time would be needed if sprites were being used). But the bits were coming in on the serial bus faster than that: a bit would come in about every 20 microseconds! So the poor CPU, frozen for longer than that, would miss some serial bits completely! Commodore's solution was to slow down the serial bus even more. That's why the VIC-20 has a faster serial bus than the 64, even though the 64 was capable, technically, of running many times faster."

So, as Mr. Butterfield explained, the 1541 DOS code was patched to send bits to the computer at a slower speed. Since VIC-20 users could take advantage of the full maximum data transfer speed, two special user commands (UI+ and UI-) were added to the DOS to allow software control of the transfer speed. In typical Commodore tradition, UI- speeds up the transfer speed 25%, while UI+ slows it down. The drive powered up in UI- mode.

So, the 1541 was basically a kludge to the 1540 to allow the drive to work with the new machine. The first units were off-white in color, most likely an effort to deplete existing 1540 case stock, while later units sport a brown case that matches the Commodore 64 computer. The decals evolved from the basic white-on-brown decal similar to the one of the 1540, to the multicolored decal present on the later units. Internally, the drive electronics went through numerous revisions, and the mechanism underwent a change as well.

The early 1541 drives used the same ALPS drive mechanism as found in the 1540. This mechanism is easily spotted because the closer mechanism of the drive is of the "push-down" type, where a user inserts a disk, and the pushes down on the drive door. To release the disk, the user pushes in on the closed door. This drive had numerous overheating and mechanical problems, so Commodore eventually formed a joint venture with a company called Mitsumi Electric Company, LTD. The joint venture was named Newtronics Company, LTD., and its sole purpose was to produce a cheaper and more reliable drive mechanism. These newer units are distinguished by the presence of a "turn-down" lever closer. The user inserts a disk and turns the lever to engage the unit. Returning the lever to the open position frees the disk for removal. These drives were indeed better, although both versions suffered overheating to some extent.

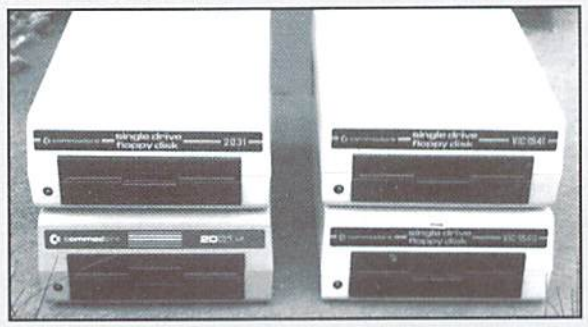
The Commodore SX-64 portable system also made use of a (built-in) 1541 disk drive unit. Although this unit has no unique model number, it could be purchased separately to upgrade the SX-64 into a double drive DX-64 portable. This upgrade was made possible by removing the plastic compartment above the lower drive and installing the second drive unit in its place.

The next version of the 1541 to be released was the 1541C disk drive, which was introduced with the Commodore 64C. This model shares the same physical dimensions of the 1541 drives, but is housed in a case color coordinated with the C-64C unit. Internally, this was the first 15xx series drive to feature a track one sensor, which could be used to alleviate the dreaded "head knock" problem. This problem, present on all of the previous Commodore drives, was caused by the drive not knowing where track one was on the disk. The DOS compensated by stepping back the head the maximum number of tracks present on the disk, thus ensuring that it would reach track one. If the head was anywhere but at the end of the disk, head banging would occur as the head reached the track one stop and the software tried to continue stepping the head. Interestingly, although early 1541C units contained a DOS patch (again, no version number change) to utilize the sensor, undisclosed compatability problems forced Commodore to make this sensor and the DOS patch optional on most versions of the 1541C. Later units contained no sensor at all. For units with the drive sensor, users could cut a jumper on the circuit board to enable it.

At this time it is important to also mention the 1542 floppy disk drive. Basically a 1541 in a charcoal gray case (color matched to the Commodore 264 series of computers), this drive was pre-announced, but may not have been actually produced.

The last drive in the 1541 series is the 1541-II disk drive. This unit was the result of a possible lawsuit. At the time, Commodore found out that a company called Chinon was developing Commodore drive mechanisms for the clone market and was underselling Commodore. CBM asked Chinon either to give them a good deal, or Commodore would sue them. Needless to say, Chinon cut CBM a deal on mechanisms, and the 1541-II was born.

With this drive, Commodore changed from using an internal power supply to supplying power via an external power supply "brick". With the power supply now located outside the drive case, the drive shrunk in size to 77mm high by 184mm wide by 256mm deep. As with the 1541, there are two versions of the 1541-II drive. One has a deep "turn-down" lever that hits a landing at the top of the front bezel (Chinon mechanism), while the other has a shallow lever that has no such external landing to rest against (Newtronics mechanism).


*Left stack, top to bottom: 2031, 2031LP. Right stack, top to bottom: VIC-1541, VIC-1540.*

## The CBM 1551

The 1551 was an "enhanced" 1541 for the Commodore 264 computers series. Although much of this unit is identical to the 1541, this drive did differ in some respects. Colored charcoal gray to match the computers, the 1551 interfaced to the Commodore Plus/4 and C16 via a special parallel interface cable. This arrangement allowed the unit to transfer data 4 to 5 times faster than the 1541, but was only usable on the 264 series. The version of the DOS in this unit is 2.6 TDISK, although the DOS is changed substantially to accommodate the parallel interface and the numerous enhancements in the drive DOS. Research suggests that the 1551 drives came with both ALPS and Newtronics drive mechanisms. Unlike earlier 15xx drives, of which up to 4 could be attached to a computer, only two 1551 units could be hooked up to a 264 series computer.

*Top to bottom: 1541-II (Newtronics), 1541-II (Chinon), the rather rare 1570.*

During the drive unit's development, prerelease documentation referred to this drive as the SFS-481 Fast Disk Drive (SFS probably stands for Super Fast System, or something similar). This drive gathered many nicknames during its development, including TED Disk, (TED stands for Text Editing Device and this name pays homage to the original 1540 being called the VIC-1540), the Kennedy Drive, or Kennedy Technology Disk (KDisk). The latter two suggest a play on the name TED Disk, and refer to Ted Kennedy, the US Congressman. (What that says about the drive I have no idea.) Ironically, this drive is a rarity in the United States, as almost all of the production units were manufactured for the European community.

## The CBM 1571

The next drive in Commodore's drive evolution is the original 1571 drive, which ushered in a new case style. The slim refined case matched the styling of the Commodore 128 computer, The computer which this drive was introduced with. This drive is, in some respects, the most complicated drive Commodore ever manufactured.

The 1571 unit can function in one of two modes. One mode, called 1541 mode, causes the unit to behave just like a 1541 drive, complete with DOS 2.6 and the idiosyncrasies of the 1541 drive. The second mode is called 1571, or native mode, where the drive allows data storage on both sides of the floppy disk. This increases the storage capacity from 170 kilobytes to 340 kilobytes.

The 1571 also has hardware and firmware that allow it to read and write any number of single- or double-sided Modified Frequency Modulation (MFM) encoded disk formats, included so that the drive could be used with the Commodore 128's CP/M mode to read CP/M disks made for other computer systems. The MFM recording system bears special mention since it is the first time since the production of the Commodore 8" 8280 floppy disk drives (which could read IBM 8" floppy disks) that Commodore supported an industry standard disk format.

While in native mode, the 1571 utilizes version 3.0 of the DOS code, which is built off the 1551 DOS code. This may seem acceptable, since Commodore finally changed the version, but Commodore had already issued version 3.0 of DOS with the 8280 8" PET/CBM floppy drive and the D9060 and D9090 PET/CBM hard drives. The drive ROMs were not similar in any way.

The various 1571 drives manufactured by Commodore made use of drive mechanisms by both Newtronics and ALPS. Unlike the 1541 model, however, the differences between the mechanisms isn't immediately obvious, as both use a "turn-down" latch engagement system.

With the 1571 drive, Commodore finally corrected the data transfer speed problems present in the 15xx series by introducing "burst" loading. Although touted as a new development, this enhancement merely utilized the original plan of performing hardware bit transfers that were developed for the 1541 disk drive.

Commodore had also created a single-sided version of the 1571, dubbed as the CBM 1570 disk drive. This unit, featuring a 1541C style case and single-sided drive mechanism, contained the same DOS version number, and could operate in burst mode just like the 1571.

With respect to the 1570 and 1571, I've been unable to determine which was released first, and how much time there was between the release of the two drives. There is some speculation that the 1570 may have been released to accompany the early C128's in Europe (where no FCC approval was needed) while Commodore was resolving some final problems with the ROM code needed for the full 1571. While the only 1570 drives I've encountered contain ALPS drive mechanisms, it is believed that 1570 drives with Newtronics mechanisms were also produced.

As Commodore prepared to release the 128D, a cost reduced version of the 1571 was created for inclusion as an internal drive for that system. Although the mechanism differs from the older external 1571, the drives are functionally equivalent. There are two names by which the 128D's internal 1571 are identified; some refer to it as the 1571D, while others refer to it as the 1571CR.

Oddly enough, Commodore designed a new external 1571 which used the same mechanism, and probably had similar electronics, and this project was referred to by those working on it as the 1571CR. It appears that this drive was to be the 1571-II drive, which is basically a 1571 drive with a smaller case and drive mechanism and a 1541-II external power supply. While some reports claim this unit exists in larger quantities in Europe, sources from within the late Commodore Engineering department claim that the design was never produced.

Commodore also worked on creating a 1572 disk drive, which was a dual drive version of the 1571 disk drive unit. The project was never completed, however, and thus the 1572 was never produced. A concern over power supply failure problems and DOS code bugs kept this drive from ever reaching the market.

## The CBM 1581

The last in the line of drives produced by Commodore for 8-bit users was the 1581 3.5" disk drive. Just as the 1571 drive differed substantially from previous drives, the 1581 was also a major departure from all the previous Commodore (non-Amiga) disk drives. For the first time, Commodore built a 3.5" drive unit, using a standard IBM drive mechanism, and supported the IBM standard MFM recording format for the disk media. To shield users from major differences, Commodore manipulated the DOS code in the drive extensively to create the illusion of a single-sided, 256-byte sector drive; in reality, the disk drive mechanism employed was double-sided, and had 512-byte sectors.

*Top to bottom: CMD FD-4000, two 1581's, CMD HD-20, 1571 (Alps), 1571 (Newtronics).*

This drive shared the same "burst" loading capabilities of the 1571 drive units, and could store 802,640 bytes of user data. The physical dimension of the unit are 63mm high by 140mm wide by 230 mm deep. This unit was only manufactured with an external power supply identical to the ones used on the 1541-II (and planned for 1571-II).

Three more 3.5" disk drive units were in the works at Commodore at one time or another. One of these was to be installed internally in a computer system often referred to as the C65 or C-64DX. The Commodore C65 was never placed into full production, however, and I'm unaware of whether the drive was ever formally named. Some call it the 1581D drive, making the 1571D analogy, while others refer to this unit as the C65 drive, as this drive is substantially different from the 1581 unit.

This internal drive used a cost reduced version of the 1581 drive mechanism. However, unlike the 1571D (which shared the same DOS code as the 1571) this unit contained a DOS that was adapted from the Commodore 8250 LP IEEE drive, version 2.7. In addition to the DOS base code switch, this drive avoided the speed limitations in the 15xx line by increasing the transfer rate between disk and computer to a maximum of 50K bits per second, compared to 4800 bits per second for the 1541 drive. The most interesting aspect of this drive is that it has NO microprocessor of its own. All previous drives contained at least one CPU, and the drives prior to the 1540 contained two. In comparison, this drive shared a CPU with the C65 computer system.

An external drive was also planned for the C65, though again, I'm not aware of a model number. This optional external 3.5" disk drive was to use a special disk drive interface plugged into a dedicated port on the rear of the C65, and was to transfer data as fast as a standard IBM floppy drive—slightly slower than the internal drive.

So, what was the third drive? This one is a recent discovery, and was also part of Commodore's ongoing engineering efforts in finding new ways to market their 8-bit line. CMD recently obtained and then sold a one-of-a-kind European-style 128D computer, the plastic portable 128 that used the standard PAL version of the 128 motherboard combined with a standard 1571 drive. But this unit had a differerence—the front panel opening had been made for a 3.5" drive instead. Inside, the new owner found a drive similar to the 1581, but the DOS ROM clearly identified it as a 1563!

## I'll Be Back...

As you can see, Commodore disk drives have a rich history. It's been difficult to document the exact progression of drives over the years, but we've a lot more information. Watch for details in future issues!

# Notes on 15xx Series Drives

All the 15xx disk drives—with the exception of the C65 external floppy drive—came from the factory wired as serial bus device number 8, even the internal drives and the 1551. With the exception of the 1551 and the C65 drives, all 15xx units utilized a 6502 CPU, identical to the one found in the VIC-20. The 1551 used a 6510T microprocessor, a derivative of the 6510 CPU in the 64. As stated earlier, the C65 internal drives used no CPU, and the CPU in the external C65 drive is unknown.

## 1540

All 1540 units are housed in off-white plastic cases with the brown/white decal, and contain an ALPS mechanism. Internally, all contain the first printed circuit board revision, nicknamed the 'long board' because it covers almost all of the drive's length.

Early versions of the 1540 DOS code contain a section of code that is designed to automatically load a file from disk upon bootup. On power-up, the drive would jump to a subroutine at $E780 after performing the reset routine. The code there would check for the high state of CLOCK and DATA. If found, the code would wait until both went low and then store '*' into the filename buffer, set the filename length to 1, and then jump to the & command, which loads a USR file and executes it. The Commodore computer never used this feature, and some machines would boot with these lines randomly high, so Commodore removed this feature in later ROM revisions. This code was also present in the early 1541 version.

## 1541

One of the best selling disk drives in Commodore's history, the 1541 underwent many changes over its lifetime. It was originally produced with the same long printed circuit board, off-white case and decal (save 1540 model ID changed to 1541), and the same ALPS mechanism in the 1540. Subsequent models sported a 64 color matched brown case, a new multicolored decal, and circuit board and DOS ROM code revisions. The ALPS mechanism was used from 1982 to the end of 1984, when the NewTronics mechanism was substituted. Models with the NewTronics mechanism are sometimes erroneously termed the 1542 by repair technicians.

The following circuit board revisions are known to exist:

| | |
|---|---|
| PCB# 1540001 | The 'long board', as used in the 1540. (would only work with the ALPS mechanism) |
| PCB# 1540008-01 | Minor revisions to the 1540001 board. |
| PCB# 1540048 | The 'short board', so named because it was shorter in length. |
| PCB# 1540050 | Minor revisions to the 1540048 board. |
| PCB# 1540050-01 | ALPS mechanism |
| PCB# 1540050-03 | NewTronics mechanism |
| PCB# 250442-01 | A revision of the short board that is termed the 1541 A board (Jumper selects mechanism) |
| PCB# 250446-01 | Minor revisions to the #250442 board, termed the 1541 A-2 board |
| PCB# 250446-03 | Cost reduced 250442-03 board. Termed the 1541A C/R |

The Commodore DOS code exists on these boards in two 8 kilobyte ROMS. One ROM, model number 325302-01, remained unchanged throughout all the 26 DOS code revisions. The other 8 kilobyte ROM can be of one of the following revisions: 901229-01, 02, 03, or 05. The 01 revision is most similar to the 1540 DOS v2.6, and contains the autoboot code. Each revision contains minor bug fixes. With all of these mechanism, case, DOS, and circuit board choices, many different combinations of the basic 1541 drive exist. Interestingly, even though the 1541 was introduced with the Commodore 64, the first models of the 1541 were labeled 'VIC-1541'. Later revisions dropped the 'VIC' prefix.

## 1541SX

This drive, only seen in the SX-64, uses many 1541 components. Documentation suggests only APLS mechanisms were utilized. Although this drive uses a standard 1541 circuit board, the ROM in this unit is different in the size of the gap between sectors written on the floppy disk, called the 'header gap'. Like the 1540, the 1541 writes 8 non-GCR bytes to disk before writing a sector to disk. The 1541SX on the other hand, writes 9 GCR bytes to disk, like the 2040/4040 drives. This anomaly usually causes no problems, but some incompatibility reports exist. Research in this area blames drive alignment problems as the cause for these incompatibilities.

## 1542

Not much is known about this drive. It is assumed that, as a repackaged 1541, it used the same components. Pre-announcement photos show the unit with a ALPS mechanism.

## 1551

To our knowledge, this drive was only produced outside of the United States. We know that it was not a big seller, and did use some 1541 components. Evidence suggests it utilized the 1541B circuit board. The DOS part number is unknown, but was included on a 16 kilobyte ROM. The Plus/4 user manual depicts a 1551 in an illustration that uses the ALPS mechanism. However, the 1551 manual depicts a drive with a NewTronics mechanism. The confirmation unit has an ALPS mechanism. Since the ALPS to NewTronics switch occurred at the end of 1984, drives with the ALPS mechanism are very rare finds.

## 1541C

Although originally slated as an upgrade from the 1541, this drive became just a repackaged 1541 drive. All 1541C units feature the beige case color, although the mechanism can be either ALPS or NetwTronics. Early drive mechanisms contain the track one sensor, although the circuit board in the unit might not enable the sensor (the user could enable the sensor by cutting a jumper), or may not support it at all. The inclusion of the track 1 sensor necessitated yet another revision of the 1541 circuit board:

| | |
|---|---|
| PCB# 250448-01 | Contains the track 1 sensor logic. Termed the 1541B board. |

Unlike earlier circuit boards which accommodated two 8 kilobyte ROMS containing the DOS code, this board contains a single socket to accommodate a 16 Kilobyte ROM. Thus, the two 8 Kilobyte DOS segments were written on a single DOS ROM for this board.

It is interesting to note that the 1541C circuit board was in production before the 1541C was produced. However, the DOS ROM evidently was not yet adapted for the circuit board, so for a few months prior to the introduction of the 1541C unit, standard 1541 units were shipped with this board, but with the old double ROMs. To adapt the two ROMS to the single socket on the circuit board, a small piggyback board was produced, model #252054.

When the 1541C unit was finally shipped, a new DOS ROM with speed and performance enhancements was installed, part number 251968-01. Then, after numerous complaints about compatibility problems and other concerns, Commodore created a new ROM revision, part number 251968-02. According to documentation, this change occurred on December 5, 1986.

With the introduction of this new ROM, Commodore finally conceded the fact that the infamous 'save with replace' bug existed and fixed it in this ROM revision. Interestingly, this bug in the code is present on all drives from the 1540 up to this revision of the 1541C, because the bug was actually a casualty of the imperfect removal of the dual drive DOS code done to create the DOS for the single drive 1540. Just because one owns a 1541C drive does not imply the existence of any of the features stated above. The 1541C case was designed to accommodate different side board and different mechanisms, so this model could have almost anything inside.

## 1541-II

As the 1541-II differed physically from the previous 1541 drives, the circuit board used in this drive differed as well, using PCB# 340503, a cost reduced board termed the 1541-II board. And, with the new circuit board, a new revision of the 16 Kilobyte DOS ROM used in the 1541C was created, part number 251968-03.

## 1570

Even though the version of DOS code in the 1570 is 3.0, there are known incompatibilities between it and the software written for the 1571 drive. It is not clear why the 1570 was produced. Some reports suggest that the 1571 drive was not yet ready for production, and the Commodore 128 introduction could not be delayed. So the 1570 was developed as an interim model. Other sources claim that the unit was designed as a lower cost alternative to the 1571, which was a somewhat expensive drive upon introduction.

## 1571

Although a successful drive in its own right, this drive suffered some reliability problems in early revisions. Although not confirmed, these units may have the ALPS mechanisms, as ALPS 1541 mechanisms had reliability problems as well.

The 1571 was a kludge, according to C65 DOS ROM writer Dennis Jarvis. According to Jarvis, Commodore Marketing touted the Commodore 128's CP/M capabilities but was referring to the use of the Commodore 64 CP/M cartridge with the C128. However, Engineering has known for some time that the CP/M cartridge only worked with the earliest versions of the Commodore 64, and would never work with the more complex C128. Work quickly began on internal CP/M support for the 128, and then the question of a drive for the unit was brought up. Managers noticed a hacked up 5.25" drive on an engineer's desk and inquired. It turns out the engineer had modified the unit to read and write MFM encoded disks. The prototype was quickly worked into what became the 1571 drive.

However the 1571 originated, it used some old technology. Numerous sources report that the 'burst' transfers were merely a new name given to the original transfer protocol slated for the 1540/VIC-20, somewhat updated for the 1571 and the C128. The ROM code for this drive was taken from the unsuccessful 1551 drive and the 1541 drive enhanced to utilize both sides of the floppy disk. Fred Bowen, formerly of Commodore and instrumental in the development of the OS and BASIC for the C128 and C65, states that the original specification of the 1571 was: 1541 + 1551 = 1571.

The bug-ridden 310654-03 ROM in the 1571 was replaced by the 310654-05 ROM on 1986-12-05 (same time as new R251968-02 ROM for 1541C). The new ROM among other things, fixed the save-with-replace bug as noted in the 1541C notes, allowed loading of 'locked' files in burst mode, and would determine whether a single or double-sided disk had been inserted much quicker than the previous version.

## 1572

Fred Bowen reports that only two such drives exist, and the unit he has burned out its internal power supply during its preview at the Winter 1986 Consumer Electronics Show. In addition to the power supply problems, Jarvis reports that the unit was simply too 'fragile' to be useful, as Commodore had tried to reintroduce the dual drive code into the 1571 DOS, code which had been taken out to produce the 1540 years earlier. The added drive 1 code was buggy at best, and frequent lockups occurred.

## 1581

Mr. Jarvis further states that this unit was another prototype drive sitting on an engineer's desk (same engineer? Who knows?) The DOS code was borrowed from the 1571, and enhanced with Commodore concept of 'subdirectories', which are unlike either IBM or CMD directory support. According to Jarvis, the ROM code in this drive was a huge 'mess', which is the main reason it was not used to develop the 1581-C65.

The 1581 drive is the first unit to contain a full track cache, meaning data was loaded into drive memory an entire track at a time. If another sector of the same track was subsequently requested, the drive would simply copy data from internal memory rather than access the disk.

This drive writes two sides of 80 tracks of 10 512-byte sectors onto a disk. However, because other Commodore drives wrote 256 byte sectors, Commodore wrote code into the DOS ROM that created a logical single sided, 80 track, 40 sector, 256 byte/sector drive.

## 1581-C65

Since the Commodore 65 was never fully developed, the characteristics of this drive weren't fully appreciated by the Commodore general public. The DRIVE DOS code was developed by Dennis Jarvis under contract to Commodore. Perhaps the most impressive feat of the internal drive is its ability to share processor time with the C65 CPU. Jarvis notes that he and Fred Bowen fought over this aspect of the drive for months. Jarvis wanted full use of the CPU during disk activities, while Bowen favored a time slicing approach. Of the handful of Commodore 65 prototypes, few share an identical DOS code revision, as the code was changed almost daily.

Dennis Jarvis notes that the 8250 LP DOS code was used to develop this unit's DOS because it was the most stable of the DOS revisions produced by Commodore.

## Credits:

Many thanks to George Page, who owns many of these drives, for information and clarification. Also, special thanks to Gil Parrish for helping nail down release dates and chronology. For the ROM version information, I am indebted to Gil and Raymond Day, who specializes in 1541 information.

# EXPLORING THE SERIAL BUS KERNAL ROUTINES

## PART 2: USING LOW-LEVEL SERIAL BUS KERNAL ROUTINES

*by Doug Cotton*

In the last issue of *Commodore World*, we began covering the Kernal routines used for serial bus device access by providing steps and an example program showing how to use high-level routines to perform file access.

In this installment, we'll learn how file access is accomplished using the low-level routines. We'll use a program example that has a different purpose than the example in the last issue, but as you get to know these routines you should easily be able to create programs that perform most any disk or file operations.

### Opening a File

Using the low-level commands to open a file is a tedious task, and one of the reasons Commodore provided the Kernal OPEN routine. However, there may be occasions when you really feel it's necessary, and that's why you're bothering to read this article.

So you're going to brave doing the open all by yourself, but before you make that big step, you're going to have to make some decisions. Most importantly, you're going to have to decide if you'll have to access this file with any of the high-level Kernal routines, or from BASIC. If so, you'll have to set up information about the file in the operating system's file tables. These tables consist of three sets of ten bytes that signify the logical file numbers, device numbers, and secondary addresses of all open files. The tables are found in different locations in the 64 and 128 operating systems, as indicated in the chart located at the bottom of this page.

When an opened file is recorded in these tables, it is done by placing the appropriate information at the same index point in each table. For example, if a file with a logical file number of 2 is recorded in the third byte of the Logical File Number Table, then the device number is found at the third byte of the Device Number Table, and the secondary address is found in the third byte of the Secondary Address Table.

This brings up the question, "Where should a new entry go in the tables?" New entries are created at the next unused file entry, starting at the first byte of each table. So how do you know which "slots" are used, and which are open? The operating system has a reserved variable location for this, found at location $98 in both the 64 and 128 operating systems. Reading this location tells you how many files are open, and at the same time provides a usable index into the tables.

Now, let's say we want to open a file on a device, and we plan to use the tables. Our example program does this by making a call to a subroutine we've created called 'tent', which you can refer to while I describe the steps involved.

The first step required is to make sure there's room in the tables for another entry (the tables can only hold ten entries, so the Commodore OS limits us to having ten files open at any given time.) Checking the value at $98 to see if it's ten or greater will provide us with the info we need.

The next step is to make sure that the logical file number we're using doesn't already exist. This can be accomplished via a small loop that compares the file number we want to use with each of the existing entries. Naturally, we can skip this step if the tables are completely empty.

Once we're certain that the table has room, and that our logical file number hasn't been used, we can go ahead and and create our entries, using the number of files open as an index for writing into the tables. Notice that when we store the secondary address in table, we must first OR it with $60. This is the way table entries for secondary addresses are expected to be stored.

Okay, we're now ready to open the file itself. The actual open is done by our 'fopen' routine, which starts at the end of 'tent', and this also serves as the entry point for opening files without using the file table. First, we'll clear the status and tell the device to listen (using LISTN), then send it the secondary address (via SECND), adjusting it first by OR'ing it with $F0 to indicate that we want to open a file.

The next step is to send the fileneame. If we were opening a command channel, the name might not be necessary, so our code checks to see if the length of the name is zero. (The only time you might use a filename when opening the command channel would be to send a command using the open, such as you might do with BASIC, i.e., OPEN15,8,15,"I0:"). If there is a filename to be sent, then CIOUT is used to send each of the individual bytes.

To complete the open operation, we tell the device to stop listening via a call to UNLSN.

**C64/128 Operating System File Tables**

| Table Description | Commodore 64 | Commodore 128 |
|---|---|---|
| Logical File Number Table | $0259 - $0262 | $0362 - $036B |
| Device Number Table | $0263 - $026C | $036C - $0375 |
| Secondary Address Table | $026D - $0276 | $0376 - $037F |

## Serial Bus Device Kernal Routine Reference Chart

| Kernal Routine | Jump Table Address | RAM Vector | Calling Parameters .A | .X | .Y | Returned Parameters .A | .X | .Y | Status | Pre-requisites |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Low-Level Routines | | | | | |
| SECND | $FF93 (65427) | n/a | SA+$60[1] | ~ | ~ | - | - | - | ST | LISTN |
| TKSA | $FF96 (65430) | n/a | SA+$60[1] | ~ | ~ | - | - | - | ST | TALK |
| ACPTR | $FFA5 (65445) | n/a | ~ | ~ | ~ | DATA | + | + | ST | TALK,TKSA |
| CIOUT | $FFA8 (65448) | n/a | DATA | ~ | ~ | + | + | + | ST | LISTN,SECND |
| UNTLK | $FFAB (65451) | n/a | ~ | ~ | ~ | - | - | - | ST | (TALK,TKSA) |
| UNLSN | $FFAE (65454) | n/a | ~ | ~ | ~ | - | - | - | ST | (LISTN,SECND) |
| LISTN | $FFB1 (65457) | n/a | DEV | ~ | ~ | - | - | - | ST | CLEAR ST |
| TALK | $FFB4 (65460) | n/a | DEV | ~ | ~ | - | - | - | ST | CLEAR ST |
| | | | | | High-Level Routines | | | | | |
| READSS | $FFB7 (65463) | n/a | ~ | ~ | ~ | STATUS | + | + | ST[2] | None |
| SETLFS | $FFBA (65466) | n/a | LFN | DEV | SA | + | + | + | n/a | None |
| SETNAM | $FFBD (65469) | n/a | FNLEN | FNAL | FNAH | + | + | + | n/a | None |
| OPEN | $FFC0 (65472) | $031A (794) | ~ | ~ | ~ | ERROR | - | - | .C | SETLFS,SETNAM[3] |
| CLOSE | $FFC3 (65475) | $031C (796) | LFN | - | - | ERROR | - | - | .C | (OPEN (CLRCH)) |
| CHKIN | $FFC6 (65478) | $031E (798) | ~ | LFN | ~ | ERROR | - | - | .C | OPEN |
| CKOUT | $FFC9 (65481) | $0320 (800) | ~ | LFN | ~ | ERROR | - | - | .C | OPEN |
| CLRCH | $FFCC (65484) | $0322 (802) | ~ | ~ | ~ | - | - | + | n/a | (CHKIN,CKOUT) |
| BASIN | $FFCF (65487) | $0324 (804) | ~ | ~ | ~ | DATA | + | + | ST | (OPEN,CHKIN) |
| BSOUT | $FFD2 (65490) | $0326 (806) | DATA | ~ | ~ | ERROR | + | + | ST | (OPEN,CKOUT) |
| LOAD | $FFD5 (65493) | [$0330 (816)] | LV[4] | (SAL) | (SAH) | ERROR | (EAL) | (EAH) | .C | SETLFS,SETNAM[3] |
| SAVE | $FFD8 (65496) | [$0332 (818)] | SAP[5] | EAL | EAH | ERROR | - | - | .C | SETLFS,SETNAM[3] |
| GETIN | $FFE4 (65508) | $032A (810) | ~ | ~ | ~ | DATA | - | - | ST | (OPEN,CHKIN) |
| CLALL | $FFE7 (65511) | $032C (812) | ~ | ~ | ~ | - | - | + | n/a | (CHKIN,CKOUT) |
| | | | | | 128 Unique Routines | | | | | |
| SPIN_SPOUT | $FF47 (65351) | n/a | ~ | ~ | ~ | - | + | + | n/a | .C[6] |
| CLOSE_ALL | $FF68 (65384) | n/a | DEV | ~ | ~ | - | - | - | n/a | None |
| SETBNK | $FF68 (65384) | n/a | BA | FNBANK | ~ | + | + | + | n/a | None |

## Reference Chart Notes & Definitions

**SYMBOL DEFINITIONS**

| | |
|---|---|
| ~ | No parameter required |
| - | Register is not preserved during operation |
| + | Register is preserved during operation |

**NOTES**

1 Add $F0 to SA instead of $60 to open file, $E0 to close file

2 STATUS byte (ST) is not cleared unless current device is 2 (RS-232)

3 SETBNK also required for 128

4 0 for LOAD (requires address in .X and .Y if SA=0, returns ending address in .X and .Y); non-zero for VERIFY (address not required)

5 Pointer to zero page location holding starting address in low byte/high byte format

6 Clear .C (CLC) to set fast serial input, set .C (SEC) to select fast serial output

**VARIABLE DEFINITIONS**

| | | |
|---|---|---|
| .C | = | Processor Carry Flag |
| ST | = | STATUS byte ($90) |
| LFN | = | Logical File Number |
| SA | = | Secondary Address |
| DEV | = | Device Number |
| BA | = | Bank for LOAD/SAVE/VERIFY (128 only) |
| LV | = | LOAD/VERIFY Flag |
| SAP | = | Starting Address Pointer |
| FNBANK | = | Bank where filename for LOAD/SAVE/VERIFY is stored (128 only) |
| FNLEN | = | Length of filename in bytes (0 if no name is required for an operation) |
| FNAL | = | Filename Address Low |
| FNAH | = | Filename Address High |
| SAL | = | Starting Address Low |
| SAH | = | Starting Address High |
| EAL | = | Ending Address Low |
| EAH | = | Ending Address High |

**ACCUMULATOR ERROR CODES**

| | |
|---|---|
| $01 | Too Many Files |
| $02 | File Open |
| $03 | File Not Open |
| $04 | File Not Found |
| $05 | Device Not Present |
| $06 | Not Input File |
| $07 | Not Output File |
| $08 | Missing Filename |
| $09 | Illegal Device Number |
| $10 | Illegal LOAD (past $FEFF) on 128 |

**STATUS BYTE VALUES**

| | |
|---|---|
| $01 | Print Time-out |
| $02 | Input Time-out |
| $40 | EOF (End Of File) |
| $42 | Read past EOF |
| $80 | Device Not Present |

**FILE SECONDARY ADDRESSES**

| | |
|---|---|
| $00-01 | Reserved for LOAD/SAVE |
| $02-0E | Input/Output Files |
| $0F | Command Channel |

## Closing Files

Whenever you're done using a file, it needs to be closed. This is probably the easiest task of all, unless you're using the file tables. Our example program has routines for both situations. The 'fclose' routine searches the logical file number table for a match with the file number of the file you're trying to close. If found, it checks to see if it is the last entry in the table. If not, it copies the data from the last table entries over the data in the table entry you're deleting. The final step is to decrement the count of open files in $98.

The 'fclose' routine drops down into the 'clos' routine, which is also the entry point for closing files without using the file tables. This routine clears the status and tells the device to listen (via LISTN). Next, OR's the secondary address of the file to be closed with $E0, and sends this to the device via SECND. The final step is to call UNLSN, and the file is closed.

One warning: don't close command channels on a device that has other files open, as this will cause ALL FILES to be closed.

## Reading Data

Getting bytes from an open file is rather simple matter. First, clear the status and then call the TALK routine to tell the device that it can talk on the serial bus (it's nice to be in charge of who can talk!). Since we're accessing a file that's already open, we need to use the secondary address that the file was opened with, OR it with $60, and send it using TKSA.

Okay, that's the preparation, and our talker is ready to send data. In our example program, we have a routine in the main program section called 'read'. This opens a data file that we've previously created with the program, reads the data from the file, and displays that data on the screen. If you follow that section of code, you'll see that the data is read by calling ACPTR, which returns one byte of data in the accumulator (.a). Since our example program does a lot of error checking (looking for the end of file as well as other errors), we've ignored the fact that the data is in the accumulator, and focus instead on checking the status first. We can get away with this because we know a little trick. You see, both the 64 and 128 ACPTR routines use memory location $A4 to store the data byte temporarily. So after we make sure the data is valid, we get the byte from that location.

To finish this routine, we call UNTLK to tell the device to stop talking after we get to the end of the file. In our example, we also go ahead and close the file, since we're done accessing it.

## Writing Data

While we've already covered writing data to the drive in our discussion of opening files, there are some differences involved in writing data to an already open file. Our example program contains a routine within the main program segment called 'write'. This routine opens a write file using our 'tent' subroutine.

Once the data file is open, accessing it is quite similar to the way we access read files. First, we clear the status byte, then tell the device to listen useing LISTN. Since we're about to access a file that's already open, we take the secondary address it was opened with, OR it with $60, and send it with a call to SECND.

The file is now ready to accept data, which we send using CIOUT. Once we've sent all the data, we call UNLSN to stop the drive from listening. And in the case of our example, we're done with the file, so we close it.

## Command Channel Access

The command channel has two common uses in programs; it serves as a way to send commands to the drive, and it's also used by the drive to inform us of errors. Our example program has routines that make use of both of these features.

Early in the main segment of our example program, we send an "Initialize" command to the drive. We do this by setting some parameters, and then calling the 'csend' subroutine. Sending commands to a drive is no different than writing data to a write file; we can, however, skip opening the file, thus making the routine a bit shorter.

Likewise, when we read data from the command channel to check for DOS errors, it's just like reading from a data file. And again, we can skip the open. Our example program does this using the 'cerr' subroutine.

## Errors & Other Stuff

Throughout the example program, you'll see a number of calls to the 'serr' and 'cerr' routines, which deal with status and command channel errors, respectively. There are also several other subroutines used in combination with these to make the program capable of detecting and reporting errors. While all of these are integral to proper error detection, it's also important to follow how the program flow is affected when an error is detected. When errors occur after a LISTN, the program must UNLSN the device as part of aborting the routine. Likewise, an error detected after TALK must cause an UNTLK. And, of course, all open files have to be closed—if possible.

In addition to the error routines, there are a couple of display routines used. The most important of these is 'primm', which is a direct copy of the print immediate Kernal routine that resides in the 128. This allows printing of "in-line" messages. It can also be omitted in 128 programs by substituting calls to the Kernal's version.

There's no doubt that the routines provided here can be optimized. But hopefully they'll give you a clearer picture of how Kernal low-level serial bus access routines are used.

```
;
; low level drive access example
;
        .org $2000
        .obj "llexample.o"
;
; variables
;
lfn     =     $b8          ; logical file number
dev     =     $ba          ; device number
sa      =     $b9          ; secondary address
st      =     $90          ; status byte
fnadr   =     $bb          ; filename address pointer
fnlen   =     $b7          ; length of filename
latbl   =     $0259        ; lfn table ($0362 for 128)
dntbl   =     $0263        ; dev# table ($036c for 128)
satbl   =     $026d        ; sa table ($0376 for 128)
listn   =     $ffb1        ; kernal listen
secnd   =     $ff93        ; kernal sa for listen
unlsn   =     $ffae        ; kernal unlisten
ciout   =     $ffa8        ; kernal serial char out
talk    =     $ffb4        ; kernal talk
tksa    =     $ff96        ; kernal sa for talk
untlk   =     $ffab        ; kernal untalk
```

```
acptr  =     $ffa5        ; kernal serial char in
setlfs =     $ffba        ; kernal set lfn, dn, sa
chrout =     $ffd2        ; kernal char out
;
; start of program
;
start   jmp  main
;
; strings
;
icmd    .byt "i"
wnam    .byt "testfile,s,w"
rnam    .byt "testfile,s,r"
endstr  .byt 0
;
; main program
;
main    lda  #$01          ; lfn in .a
        ldx  #$08          ; dev in .x
        ldy  #$02          ; sa in .y
        jsr  setlfs        ; and set
;
        jsr  primm         ; print immediate
        .byt 147           ; clear screen
        .byt "initializing drive"
        .byt 13,0
;
        lda  #<icmd       ; low byte of address
        sta  fnadr        ; store it
        lda  #>icmd       ; high byte of address
        sta  fnadr+1      ; store it
        lda  #wnam-icmd   ; calculate length
        sta  fnlen        ; and store
        jsr  csend        ; call send command routine
        bcc  +            ; branch if no error
        jmp  serr         ; process status error
;
+       jsr  cerr         ; check cmd channel error
        bcc  +            ; status okay
        jmp  serr         ; process status error
;
+       lda  errflg       ; check error
        beq  write        ; no error, branch
        jsr  prnterr      ; print error
        rts               ; end program
;
; write data file
;
write   jsr  primm        ; print immediate
        .byt "opening write file"
        .byt 13,0         ; return, end of data
;
        lda  #<wnam       ; low byte of name address
        sta  fnadr        ; store it
        lda  #>wnam       ; high byte
        sta  fnadr+1      ; store it
        lda  #rnam-wnam   ; calculate length
        sta  fnlen        ; and store
        jsr  tent         ; call send command routine
;
        bcc  +            ; branch if no error
        jsr  serr         ; process status error
        jmp  endw         ; skip to end
;
+       jsr  cerr         ; check cmd channel error
        bcc  +            ; status okay
        jsr  serr         ; process status error
        jmp  endw         ; abort write
;
+       lda  errflg       ; get error number
        beq  +            ; branch if no error
        jsr  prnterr      ; else print error
        jmp  endw         ; abort write
;
+       lda  #$00         ; always clear status
        sta  st           ; before listn
        lda  dev          ; load .a with dev#
        jsr  listn        ; tell device to listen
        bit  st           ; check status
        bpl  +            ; status okay, branch
        jsr  nodevl       ; device not present
        jmp  endw         ; jump to end
;
+       lda  sa           ; load secondary address
        ora  #$60         ; adjust for write
        jsr  secnd        ; send sa for listen
        lda  st           ; load status
        bpl  +            ; status okay, continue
        jsr  nodevl       ; device not present
        jmp  endw         ; jump to end
;
str     .byt "sample data string"
        .byt $0d          ; a carriage return
slen    .byt $00          ; end of data
;
+       jsr  primm        ; print immediate
        .byt "writing data"
        .byt 13,0
;
        lda  #slen-str    ; calculate the length
        sta  slen         ; and store it
        ldx  #$00         ; clear .x

-       lda  str,x        ; get char from string
        jsr  ciout        ; output it
        lda  st           ; get status
        beq  +            ; no error, branch
;
        sta  tmpst        ; save status
```

```
        jsr  unlsn        ; unlisten device
        jsr  unk          ; unknown error
        jmp  xit          ; jump to end
;
+       inx               ; okay, increment .x
        cpx  slen         ; all chars sent?
        bne  -            ; no, get next char
;
+       jsr  unlsn        ; stop listening
        clc               ; clear carry
;
xit     bcc  +            ; branch if no error
        jmp  serr         ; process error
+       jsr  cerr         ; check drive error channel
        bcc  +            ; status okay
        jmp  serr         ; process error
+       lda  errflg       ; check for drive error
        beq  endw         ; exit write if no error
        jsr  prnterr      ; print error
;
endw    jsr  primm        ; print immediate
        .byt "closing write file"
        .byt 13,0
;
 jsr    fclose            ; close file
        bcc  +            ; branch if no error
        jmp  serr         ; process error
+       jsr  cerr         ; check for drive error
        bcc  +            ; branch if no error
        jmp  serr         ; process error
+       lda  errflg       ; check for drive error
        beq  read         ; skip to read if no error
        jsr  prnterr      ; print error
        rts               ; abort
;
; read/display data file
;
read    jsr  primm        ; print immediate
        .byt "opening read file"
        .byt 13,0
;
        lda  #<rnam       ; low byte of name
        sta  fnadr        ; store it
        lda  #>rnam       ; high byte of address
        sta  fnadr+1      ; store it
        lda  #endstr-rnam; calculate length
        sta  fnlen        ; and store
        jsr  tent         ; call send command routine
;
        bcc  +            ; branch if no error
        jsr  serr         ; process status error
        jmp  endr         ; abort read
;
+       jsr  cerr         ; check cmd channel error
        bcc  +            ; status okay
        jsr  serr         ; process status error
        jmp  endr         ; abort read

+       lda  errflg       ; get drive error number
        beq  +            ; branch if no error
        jsr  prnterr      ; else print it
        jmp  endr         ; abort read
;
+       jsr  primm        ; print immediate
        .byt "reading data"
        .byt 13,0
;
        lda  #$00         ; always clear status
        sta  st           ; before talk
        lda  dev          ; load .a with dev#
        jsr  talk         ; tell device to talk
        bit  st           ; check status
        bpl  +            ; status okay, branch
        jsr  nodevt       ; device not present
        jmp  endr         ; abort read
;
+       lda  sa           ; load secondary address
        ora  #$60         ; adjust for read
        jsr  tksa         ; send sa for talk
        lda  st           ; load status
        bpl  rdbyt        ; status okay, continue
        jsr  nodevt       ; device not present
        jmp  endr         ; jump to end
;
-       lda  $a4          ; get last byte received
```

```
        jsr  chrout         ; output to screen
rdbyt   jsr  acptr          ; get a byte
        lda  st             ; get status
        beq  -              ; status okay, branch
;
        cmp  #$40           ; end of file?
        beq  +              ; yes, then branch
;
        sta  tmpst          ; store status
        jsr  untlk          ; untalk device
        jsr  unk            ; unknown error
        jmp  endr           ; jump to end
;
+       lda  $a4            ; yes, get last byte
        jsr  chrout         ; and print it
        jsr  untlk          ; untalk device
;
endr    jsr  primm          ; print immediate
        .byt "closing read file"
        .byt 13,0
;
        jsr  fclose         ; close file
        bcc  +              ; branch if no error
        jmp  serr           ; process error
;
+       jsr  cerr           ; check drive for error
        bcc  +              ; check worked, branch
        jmp  serr           ; process error
;
+       lda  errflg         ; check drive error
        beq  +              ; none, branch
        jsr  prnterr        ; print error
;
+       rts                 ; exit program
;
; make table entry
;
tent    lda  $98            ; get number of open files
        beq  ++             ; table empty, branch
        cmp  #$0a           ; compare to 10
        bne  +              ; not 10, branch
        jmp  tmf            ; error, too many files
;
+       lda  lfn            ; desired lfn in .a
        ldx  $98            ; open files in .x
-       dex                 ; decrement .x
        bmi  +              ; no matches, branch
        cmp  latbl,x        ; compare lfn to table
        bne  -              ; next compare
        jmp  exists         ; file exists

+       ldx  $98            ; get open file index
        lda  lfn            ; get new lfn
        sta  latbl,x        ; store lfn in lfn table
        lda  dev            ; get new dev#
        sta  dntbl,x        ; store dev# in dn table
        lda  sa             ; get new sa
        ora  #$60           ; or sa with $60
        sta  satbl,x        ; store sa in sa table
        inc  $98            ; increment # of open files
;
; open file
;
fopen   lda  #$00           ; always clear status
        sta  st             ; before listn
        lda  dev            ; load .a with dev#
        jsr  listn          ; tell device to listen
        bit  st             ; check status
        bpl  +              ; okay, branch
        jmp  nodevl         ; device not present
;
+       lda  sa             ; load secondary address
        ora  #$f0           ; adjust for open
        jsr  secnd          ; send sa for listen
        lda  st             ; load status
        bpl  opn            ; okay, continue
        jmp  nodevl         ; device not present
;
opn     lda  fnlen          ; get length of name
        beq  ++             ; no name, skip ahead
        ldy  #$00           ; zero out .y
-       lda  (fnadr),y      ; get 1 char from name
        jsr  ciout          ; output char to device
        lda  st             ; get status
        beq  +              ; okay, branch
```

```
        jsr  unlsn          ; unlisten device
        lda  #$ff           ; indicate unknown error
        sec                 ; set carry
        rts                 ; and exit
+       iny                 ; increment index
        cpy  fnlen          ; all chars sent?
        bne  -              ; no, get next char
+       jsr  unlsn          ; yes, unlisten
        clc                 ; clear carry
        rts                 ; and exit
;
; close file (in table)
;
fclose  lda  lfn            ; get lfn
        ldx  $98            ; get number of files open
-       dex                 ; dec to use as pointer
        bmi  clos           ; none found, exit
        cmp  latbl,x        ; check for match
        bne  -              ; no match, next
        dec  $98            ; decrement file count
        cpx  $98            ; is entry last in table?
        beq  clos           ; yes, branch
        ldy  $98            ; index to last      entry
        lda  latbl,y        ; move last lfn entry
        sta  latbl,x        ; to empty slot
        lda  dntbl,y        ; move last dev# entry
        sta  dntbl,x        ; to empty slot
        lda  satbl,y        ; move last sa entry
        sta  satbl,x        ; to empty slot
;
; close file
;
clos    lda  #$00           ; always clear the status
        sta  st             ; before calling listn
        lda  dev            ; load .a with dev#
        jsr  listn          ; and call listn
        bit  st             ; check status
        bpl  +              ; okay, branch
        jmp  nodevl         ; device not present
;
+       lda  sa             ; sa for file
        ora  #$e0           ; adjust sa for close
        jsr  secnd          ; send sa for listener
        bit  st             ; check status
        bpl  +              ; okay, branch
        jmp  nodevl         ; device not present
+       jsr  unlsn          ; unlisten device
        clc                 ; clear carry
        rts                 ; and exit
;
; set status error parameters
;
unk     lda  tmpst          ; get status
        and  #$0f           ; clear upper nibble
        tax                 ; transfer to x
        lda  hex,x          ; get ascii value
        sta  unkst+1        ; store it
        lda  tmpst          ; get status
        clc                 ; clear carry
        lsr                 ; shift right
        lsr                 ; until upper
        lsr                 ; nibble becomes
        lsr                 ; lower nibble
        tax                 ; transfer to x
        lda hex,x           ; get ascii value
        sta  unkst          ; store it
;
        lda  #$ff           ; unknown error
        sec                 ; set carry
        rts                 ; exit
;
hex     .byt '0123456789abcdef'
;
tmf     lda  #$01           ; too many files
        sec                 ; set carry
        rts                 ; exit
;
exists  lda  #$03           ; file exists
        sec                 ; set carry
        rts                 ; exit
;
nodevt  jsr  untlk          ; tell device to shut-up
        jmp  +              ; skip unlisten
nodevl  jsr  unlsn          ; stop device from listening
+       lda  #$05           ; device not present error
```

```
        sec                 ; set carry
        rts                 ; exit
;
; display status error
;
serr    jsr primm          ;print immediate
        .byt "status error: "
        .byt 0
;
        cmp  #$01           ; is error too many files?
        bne  +              ; no, then branch
        jsr  primm          ; yes, print immediate
        .byt "too many files"
        .byt 13,0
;
+       cmp  #$03           ; is it file exists?
        bne  +              ; no, then branch
        jsr  primm          ; yes, print immediate
        .byt "file exists"
        .byt 13,0
;
+       cmp  #$05           ; is it device not present?
        bne  +              ; no, then branch
        jsr  primm          ; yes, print immediate
        .byt "device not present"
        .byt 13,0
;
+       cmp  #$ff           ; is it unknown type?
        bne  +              ; no, then branch
        jsr  primm          ; yes, print immediate
        .byt "unknown ($"
unkst   .byt $30,$30
        .byt ")"
        .byt 13,0
;
+       rts                 ; exit
;
tmpst   .byt $00            ; used for temporary status
;
; print disk command channel error
;
prnterr ldx  #$00           ; zero index
-       lda  errbuf,x       ; get from error buffer
        beq  +              ; branch if end of string
        jsr  chrout         ; output character
        inx                 ; increment index
        bne  -              ; get next character
+       rts                 ; exit
;
errbuf  .buf 256            ; buffer for error msg
endbuf  .byt $00            ; arbitrary end of buffer
errflg  .byt $00            ; error value
;
; read command channel
;
cerr    lda  #$00           ; always clear the status
        sta  st             ; before calling talk
        lda  dev            ; load .a with dev#
        jsr  talk           ; and call talk
        bit  st             ; check status
        bpl  +              ; okay, branch
        jmp  nodevt         ; device not present
;
+       lda  #$6f           ; sa for command channel
        jsr  tksa           ; send sa for talker
        bit  st             ; check status
        bpl  +              ; okay, branch
        jmp  nodevt         ; device not present
;
+       ldx  #0             ; zero index
-       jsr  acptr          ; get a byte
        lda  st             ; get status
        bne  +              ; status bad, branch
        lda  $a4            ; get last char in .a
        sta  errbuf,x       ; store in buffer
        inx                 ; increment index
        bne  -              ; get next char
;
+       cmp  #$40           ; end of file?
        beq  +              ; yes, then branch
;
        sta  tmpst          ; no, store st
        jsr  untlk          ; tell drive to shut-up
        jsr  unk            ; unknown status
        sec                 ; set carry to indicate
```

```
        rts                 ; error and exit
;
+       lda  $a4            ; get last byte
        sta  errbuf,x       ; and store it
;
        lda  #$00           ; zero byte
        inx                 ; increment index
        sta  errbuf,x       ; and store it
;
        lda  errbuf         ; get 1st byte of error
        and  #$0f           ; convert to value
        sta  errflg         ; store it
        asl                 ; multiply by 2 (x2)
        asl                 ; multiply by 2 (x4)
        adc  errflg         ; add it (x5)
        asl                 ; multiply by 2 (x10)
        sta  errflg         ; store it
        lda  errbuf+1       ; get 2nd byte
        and  #$0f           ; convert to value
        ora  errflg         ; or in tens
        sta  errflg         ; and store
;
        jsr  untlk          ; untalk device
        clc                 ; clear carry
        rts                 ; and exit
;
; send command to drive
;
csend   lda  #$00           ; always clear the status
        sta  st             ; before calling listn
        lda  dev            ; load .a with dev#
        jsr  listn          ; and call listn
        bit  st             ; check status
        bpl  +              ; okay, branch
        jmp  nodevl         ; device not present
;
+       lda  #$6f           ; sa for command channel
        jsr  secnd          ; send sa for listener
        bit  st             ; check status
        bpl  +              ; okay, branch
        jmp  nodevl         ; device not present
;
+       ldy  #$00           ; clear .y
-       lda  (fnadr),y      ; get char from command
        jsr  ciout          ; output it
        lda  st             ; get status
        bne  +              ; error, branch
        iny                 ; okay, increment .x
        cpy  fnlen          ; all chars sent?
        bne  -              ; no, get next char
;
+       jsr  unlsn          ; stop listening
        clc
        rts
;
; print immediate routine
;
primm   pha                 ; save .a on stack
        txa                 ; transfer .x to .a
        pha                 ; save .x on stack
        tya                 ; transfer .y to .a
        pha                 ; save .y on stack
        ldy  #$00           ; zero index
-       tsx                 ; get stack pointer in .x
        inc  $0104,x        ; increment return address
        bne  +              ; branch if result not zero
        inc  $0105,x        ; incr high byte of address
;
+       lda  $0104,x        ; get low address
        sta  $bb            ; save in zero page
        lda  $0105,x        ; get high address
        sta  $bc            ; save in zero page
        lda  ($bb),y        ; get character via pointer
        beq  +              ; branch if zero
        jsr  chrout         ; else output character
        bcc  -              ; get next character
;
+       pla                 ; get .y from stack
        tay                 ; transfer to .y
        pla                 ; get .x from stack
        tax                 ; transfer to .x
        pla                 ; get .a from stack
        rts                 ; exit
;
        .end                ; end assembly
```

## CMD • HARDWARE • ORDERS: 1-800-638-3263 • SOFTWARE • CMD

### HD Series SCSI Hard Disk Drives

| | |
|---|---|
| **HD-40, 42 MB (Limited Supply)** | **$249.00** |
| **HD-340, 340 MB (Special Edition)** | **$349.00** |
| **HD-500, 500+ MB (Special Edition)** | **$499.00** |
| **HD-1000, 1 GB (Special Edition)** | **$599.00** |
| **HD-2000, 2 GB (Special Edition)** | **$999.00** |

### FD Series 3.5" Floppy Disk Drives

| | |
|---|---|
| **FD-2000 (800K and 1.6 MB) (Special Offer)** | **$159.00** |
| FD-4000 (800K, 1.6 MB and 3.2 MB) | $249.95 |
| FD Real-Time-Clock Option/Kit | $20.00/$29.00 |
| Box of 10, High Density Disks (1.6MB) | $14.95 |
| Box of 10, Enhanced Density Disks (3.2 MB) | $29.00 |

### JiffyDOS

**(Specify computer serial number and drive model)**

| | |
|---|---|
| JiffyDOS C64/SX-64 System | $49.95 |
| JiffyDOS C-128/128-D System | $59.95 |
| Additional Drive ROM's | $24.95 |

### Miscellaneous Hardware

| | |
|---|---|
| 80-column Monitors (Refurb) | CALL |
| Aprotek 3-Way User Port Expander | $30.00 |
| Aprotek User Port Extension Cable | $19.00 |
| C-64/64-C Computers (Refurb, w/JD) | $89.00 |
| C-64, C-64C Power Supply (Repairable) | $39.00 |
| C-128 Power Supply (Repairable) | CALL |
| Cannon BJ-200ex Bubble Jet Printer | $289.00 |
| Cannon BJC-4100 Color Bubble Jet Printer | CALL |
| Commodore 1541 Disk Drives (NEW, w/JD) | $119.00 |
| Commodore 1541 Disk Drives (Refurb, w/JD) | $75.00 |
| Commodore 1541-II Disk Drive (Refurb, w/JD) | $89.00 |
| Commodore 1571 Disk Drive (Refurb, w/ JD) | $99.00 |
| Commodore 1802 40-column Monitor (Refurb) | $139.00 |
| CMD EX2+1 3-Port Cartridge Port Expander | $34.95 |
| CMD EX3 3-Port Cartridge Port Expander | $29.95 |
| CMD Gamepad/Joystick Controller | $24.95 |
| CMD GeoCable II Print Cable 6ft./15ft. | $29.00/$34.00 |
| CMD SmartMouse (1351 Compatible Mouse) | $49.95 |
| CMD SmartTrack (1351 Compatible Trackball) | $69.95 |
| Monitor Cables | CALL |
| Mouse Pad | $2.95 |
| MW-350 Printer Interface (ØK/8K Buffer) | $49.00/$60.00 |
| **Panasonic KXP-1150 9-pin Printer** | **$169.00** |
| **Panasonic KXP-2023 24-pin Printer** | **$229.00** |
| **Samsung SP-2417 24-pin Color Printer** | **$319.00** |

### Utilities

| | |
|---|---|
| Big Blue Reader V4.10 (SOGWAP) | $39.00 |
| CMD Utilities | $24.95 |
| JiffyMON-64 (ML Monitor) | $19.95 |
| The Compression Kit '94 (Mad Man) | $39.00 |

### GEOS

| | |
|---|---|
| Collette Utilities (Handy Geos Utilities) | $19.95 |
| Desk Pack Plus | $29.00 |
| Dweezils Greatest Hits (NewTools2,Stamp,Label128) | $30.00 |
| FONTPACK Plus | $25.00 |
| gateWay 64 or 128 (Specify Version) | $29.95 |
| geoBASIC | $20.00 |
| geoCalc 64/128 | $40.00/$45.00 |
| geoChart | $29.00 |
| **geoFAX** | **$39.95** |
| geoFile 64/128 | $40.00/$45.00 |
| geoMakeBoot (Makes Bootable copies) | $12.95 |
| geoProgrammer | $45.00 |
| geoPublish | $40.00 |
| GEOS 64 v2.0 | $44.00 |
| GEOS 128 v2.0 | $49.00 |
| geoSHELL V2.2 (CLI for GEOS) | $24.95 |
| International FONTPACK | $25.00 |
| Perfect Print LQ for GEOS (Laser-like output) | $49.95 |
| RUN GEOS Companion | $20.00 |
| RUN GEOS Power Pak I or II (Specify) | $20.00 |

### Books

| | |
|---|---|
| Anatomy of the 1541 | $12.00 |
| Basic Compiler Design for the C-64 | $12.00 |
| C-64 Science & Engineering | $12.00 |
| C128 Computer Aided Design | $12.00 |
| C128 BASIC Training Guide | $12.00 |
| Cassette Book for C-64 and Vic 20 | $12.00 |
| Commodore 64 Tricks and Tips | $12.00 |
| GEOS Programmers Reference Guide | $35.00 |
| Graphics Book for the C-64 | $12.00 |
| Hitchhikers Guide to GEOS | $35.00 |
| Ideas for Use on Your C-64 | $12.00 |
| Printer Book for the C64 | $12.00 |
| Mapping the C64 | $12.00 |
| Simple Internet | $16.95 |
| Superbase - The Book | $15.00 |

### Scanning & Video

| | |
|---|---|
| Handyscanner | $249.00 |
| Pagefox | $139.00 |

### RAMLink Power-Backed RAM Disk

| | |
|---|---|
| RAMLink Base Model (ØMB, No RAMCard) | $149.00 |
| **RAMLink w/1 MB RAMCard** | **$229.00** |
| **RAMLink w/4 MB RAMCard** | **$349.00** |
| RAMCard RTC Option/Kit | $10.00/$19.00 |
| RAMLink Battery Back-up (Optional) | $24.95 |
| Parallel Cable (RAMLink to HD) | $14.95 |

### MIDI & Sound

| | |
|---|---|
| Digimaster | $34.95 |
| SID Symphony Stereo Cartridge | $44.95 |
| Sonus 64 Sequencer, MIDI Interface, Cables | $99.00 |
| Sonus 128 Sequencer, MIDI Interface, Cables | $99.00 |

### Languages & Compilers

| | |
|---|---|
| BASIC 64 Compiler (Abacus) | $17.00 |
| BASIC 128 Compiler (Abacus) | $25.00 |
| Blitz! 64 Compiler (Skyles) | $30.00 |
| Blitz! 128 Compiler (Skyles) | $30.00 |
| Buddy 64/128 Assembler | $39.00 |
| Cobol 64 (Abacus) | $17.00 |
| Fortran 64 (Abacus) | $17.00 |
| Pascal 64 (Abacus) | $17.00 |
| Power C 64 (Spinnaker) | $14.00 |

### Productivity

| | |
|---|---|
| Cadpak 64 (Abacus) | $22.00 |
| Cadpak 128 (Abacus) | $25.00 |
| Chartpak 64 (Abacus) | $17.00 |
| Chartpak 128 (Abacus) | $25.00 |
| I Paint v1.5 (128, 80-col., 64K VDC) (Living Proof) | $39.00 |
| I Port v1.54 (128, 80-col., 64K VDC) (Living Proof) | $29.00 |
| Outrageous Pages (Batteries Included) | $19.00 |
| Personal Portfolio Manager (Abacus) | $16.00 |
| Pocket Writer 1 (64) Digital Sol.) | $35.00 |
| Pocket Writer 2 (64/128) (Digital Sol.) | $65.00 |
| Pocket Writer 3 (64 or 128) (Digital Sol.) | $70.00 |
| Pocket Planner 2 or Pocket Filer 2 (Digital Sol.) | $35.00 |
| PowerPlan 64 (Abacus) | $16.00 |
| RUN Productivity Pak I, II, or III (Specify) | $15.00 |
| RUN Super Starter Pak 1541 or 1581 | $20.00 |
| RUN Works | $20.00 |
| SEC Check Register 128 | $29.00 |
| SuperScript 64 (Precision) | $15.00 |
| SuperScript 128 (Precision) | $20.00 |
| Suberbase 64 Version 3.01 (Precision) | $35.00 |
| Superbase 128 Version 3.01 (Precision) | $35.00 |
| SwiftCalc 64 (Timeworks) | $16.00 |
| Perfect Tax 64 (Free '95 Upgrade) | $69.00 |
| Perfect Tax 128 (Free '95 Upgrade) | $79.00 |
| TWS 64 w/Speller (Busy Bee) | $29.00 |
| TWS 128 w/Speller (Busy Bee) | $39.00 |
| TWS Modules (HD/RL/Illustrator) | each $5.00 |

### Telecommunications

| | |
|---|---|
| Aprotek MiniModem C-24 (C= ready, 2400 baud) | $69.00 |
| Aprotek MiniModem C (C= ready, 1200 baud) | $50.00 |
| Aprotek Modem adapter (C= to Ext. PC Modem) | $20.00 |
| BOCA 2400 Baud Modem | $69.00 |
| BOCA 2400 w/SwiftLink and Cable | $99.00 |
| BOCA 14.4K bps FaxModem | $129.00 |
| BOCA 14.4K w/SwiftLink & Cable | $159.00 |
| BOCA V.34 28.8K bps FaxModem | $219.00 |
| BOCA V.34 w/SwiftLink & Cable | $249.00 |
| Dialogue 128 | $29.00 |
| SpeedTerm (Abacus) | $25.00 |
| SwiftLink RS-232 Cartridge (Up to 38.4K baud) | $39.95 |
| SwiftLink Modem Cable (DB9-DB25) | $9.95 |

### Games

| | |
|---|---|
| Atomino | $17.00 |
| Ballistix | $13.00 |
| Blood Money | $15.00 |
| Chomp! | $10.00 |
| Day in the Life of Prehistoric Man | $19.00 |
| Escape Route | $19.00 |
| Heavenbound | $19.95 |
| Island of the Dragon | $19.00 |
| Laser Squad | $13.00 |
| Lions of the Universe | $19.95 |
| Mainframe | $13.00 |
| Menace | $15.00 |
| Navy Seal | $10.00 |
| Rings of Medusa | $16.00 |
| RUN C128 Funpak | $10.00 |
| RUN C64 Gamepak | $10.00 |
| Skate or Die | $13.00 |
| The Amazing Spider-Man | $15.00 |
| The President Is Missing! | $10.00 |
| The Three Stooges | $10.00 |
| Tie Break Tennis | $16.00 |
| Total Eclipse | $10.00 |
| Ultima V | $17.00 |
| Wings of Circe | $19.00 |
| Wizardry 5: Heart of the Maelstrom | $24.00 |

### JiffyDOS

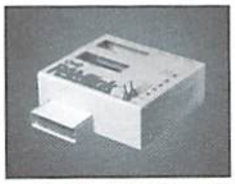
**Increase Speed Up to 1500% while retaining 100% compatibility**

- Speeds up Loading, Saving, Verifying, Formatting and Reading/Writing of Program, Sequential, User and Relative files unlike cartridges which only speed up Loading and Saving of PRG files
- Built-in DOS Wedge plus 17 additional features including file copier, text dump, printer toggle, and redefinable function keys make using your computer easier and more convenient
- ROM upgrade installs easily into most computers and disk drives. Supports C-64, 64C, SX-64, C-128, 128-D, 1541, 1541C, 1541-II,1571, 1581 and more.

### FD Series

**3.2 and 1.6 MB 3.5" Floppy Drives**

The FD-2000 and FD-4000 disk drives utilize today's latest 3.5 inch disk drive technology. FD-2000's support 800K (1581 style) and 1.6 MB (High Density) formats, while the FD-4000 also offers support for the 3.2 MB (Enhanced Density) disk format. Fast and reliable, they support 1541, 1571 and 1581 style partitions, Native Mode partitioning and can actually read and write 1581 disks. FD's feature built-in JiffyDOS, SWAP button and optional Real Time Clock. High capacity, speed and compatibility make the FD right for every application, including GEOS.



### RAMLink

**Power-Backed Expandable REU**

The fastest possible form of storage, RAMLink provides instant access to files and retains data while your computer is turned off. Easy to use and expandable up to 16 MB, RAMLink utilizes the same powerful operating system found in CMD Hard Drives. Unlike Commodore REU's which are compatible with less than 1% of commercial software, RAMLink supports more than 80% of the commercial titles. RAMLink also offers built-in JiffyDOS Kernal, SWAP feature, reset button, enable/disable switch, pass-thru port and RAM port for REU/GEORAM use. RAMLink offers maximum speed, expandability and compatibility with all types of software and hardware including GEOS.

### HD Series

**SCSI Hard Drive for the C-64/128**

HD Series Hard Drives are available in capacities up to 1 GB, are fully partitionable, and can emulate 1541, 1571, & 1581 disks while Native partitions utilize MS-DOS-style subdirectories. HD's connect easily to the serial bus or parallel via RAMLink. Includes built-in JiffyDOS, SWAP feature and RTC. HD's offer superior compatibility with most commercial software including BBS, Productivity and GEOS. And with new pricing, HD Series drives offer the lowest cost per megabyte of any C64/128 storage device.

### Shipping and Handling Charges

Use the chart below to match your order subtotal with your shipping zone and method

| | Continental United States | | | | | |
|---|---|---|---|---|---|---|
| Order Subtotal | UPS Ground | 2nd Day | Next Day | AK HI, PR | Canada | Foreign |
| $0.01 - $19.99 | $3.00 | $8.00 | $17.00 | $12.00 | $5.00 | Call |
| $20.00-$29.99 | $5.00 | $9.00 | $18.00 | $14.00 | $7.00 | Call |
| $30.00-$59.99 | $6.00 | $10.00 | $20.00 | $15.00 | $9.00 | Call |
| $60.00-$149.99 | $8.00 | $12.00 | $23.00 | $19.00 | $12.00 | Call |
| $150.00-$299.99 | $10.00 | $14.00 | $27.00 | $21.00 | $20.00 | Call |
| $300.00-$799.99 | $15.00 | $20.00 | $34.00 | $27.00 | $25.00 | Call |
| $800.00 + | $20.00 | $25.00 | $40.00 | $32.00 | $35.00 | Call |

UPS C.O.D. add $5.00 (U.S./PR only)

**Policies:** CMD accepts: Major Credit Cards, Money Orders, COD and Personal Checks. Personal Checks are held 3 weeks. Customer is responsible for shipping charges on refusals. Most items are stock, contact CMD for firm delivery. Returns for merchandise credit only within 30 days with prior authorization. Credits are less shipping, handling, taxes, customs, duties, and a 10% restocking charge. No refunds or credits on opened software. All prices and specifications are subject to change without notice.

**WE ACCEPT ALL MAJOR CREDIT CARDS**

MasterCard VISA DISCOVER NOVUS

**Creative Micro Designs, Inc.**

P.O. Box 646
East Longmeadow, MA 01028

**Info:** (413) 525-0023
**Fax:** (413) 525-0147

# Today's Most Advanced Mass Storage Devices Just Became More Affordable

## Our lowest price ever on the FD-2000 3.5" Floppy Disk Drive!

### FD-2000 Drive Specifications

- Amazing 1.6 MB capacity, plus reads & writes 1581 disks
- Up to 20% faster than a 1581 on stock computers, and up to 1500% faster on JiffyDOS-equipped systems
- Emulates 1541/71/81 disk formats for better compatibility, plus CMD Native partitions for greater flexibility
- Configurable as devices 8 through 15, plus front panel SWAP button for instant device 8 or 9 selection
- Optional Real-Time Clock (RTC) stamps time and date on all files and automatically sets GEOS clock
- Reads, Writes & Formats MS-DOS diskettes using Big Blue Reader utility program (available separately)

**ONLY $159⁰⁰**

Optional Real-Time Clock (RTC) add $20⁰⁰

This offer valid only in U.S. and Canada through December 31, 1995. May not be combined with any other offers. Prices do not include shipping and handling. See our main advertisement on the opposite page for shipping prices and policies.

## To Order Yours, Call 1-800-638-3263

## If you thought the HD-40 was affordable before, just look at this...

Recently, during one of our routine searches for hard disk drives, we were offered a significant number of 40 MB Conner and Quantum SCSI hard disk drives at a price so low that we just couldn't refuse. Additionally, were offered some superb pricing on the larger Hard Drives. So low, in fact, that we're offering the HD-340 at a price nearly $1,500.00 lower than the original HD-200 back in 1990. If you've ever considered adding a CMD HD Series hard drive to your system, **now is the time to do it!** These outstanding prices will only last as long as our supply of these specially priced drives. So don't hesitate—add the power and flexibility of a CMD hard drive to your system today!

**CMD HD-40 JUST $249⁰⁰**

| | | |
|---|---|---|
| HD-340 | 340 Megabytes | $349⁰⁰ |
| HD-500 | 540 Megabytes | $499⁰⁰ |
| HD-1000 | 1 Gigabyte | $599⁰⁰ |
| HD-2000 | 2 Gigabytes | $999⁰⁰ |

Prices do not include shipping and handling. See our main advertisement on the opposite page for shipping prices and policies.

CMD Creative Micro Designs, Inc.

SOFTWARE IN REVIEW

# FONTIGUS 128

*Fontigus 128 v2.0: "A Professional 80 Column Character Editor for the Commodore 128 and 128D". $29.95 from PHD Software Systems, PO Box 23, Moville, Iowa 51039-0023.*

Character sets give programs their personality and style, for the shape of letters and graphic characters can have an impact on how the program is perceived. Any program that has text uses character sets. Some use the set which is in the kernal ROM; others use sets specially designed for the program. Game programs use character sets for a different purpose: for the graphics you see on screen and in the background of a game.

Depending on the program used, editing character sets could be a tedious process, but Fontigus 128, an 80 column character editor, has features that make changing multiple character sets of any size quick and easy. One of my favorite features is the ability to have up to seven character sets in memory at the same time. Each set can be edited and saved individually. The user can cut and paste ranges or individual characters from one set to another. My second favorite feature is a "global" mode that lets you select multiple characters to change characteristics of them all at the same time. For instance, if I wanted all the lowercase letters in a character set to be shifted down a row, I can change them all at once instead of editing each individually. This isn't found in most other character editors.

The documentation for Fontigus, a 14 page spiral booklet, explains the difference between the upper case/graphics set, the lower case alphabetic set, and how each are used. Each function of the program and commands for each mode are described in detail. A command reference and index are also included.

After loading Fontigus, the main screen appears with both the upper and lower case portions of the CBM ROM character sets displayed, and a dividing line to show which set is which. Most of the commands are shown on the screen to remind the user of what's available.

The program will load any file as a character set regardless of its name. It can be interesting to load a non-character set into an editor just to see the results. Usually it's not very pretty or useful, but it shows how the editor manages unusual "character sets". With other editors, when I've loaded a character set that used different positions for the characters than what the editor was expecting, any helpful text on the screen turned into a mess, making it hard to read what keys to use for the different functions. Fontigus deals with this problem in a couple of ways. Since it uses the lower case alphanumeric set for its text, the user can either swap the upper and lower case sets while editing and then swap them back before saving, or the user can press the "0" key to temporarily restore the display using the CBM ROM character set instead of the one being edited.

The character the cursor is on is displayed in a grid on the main screen. Alongside this, the row byte values are shown, as well as the ASCII (Petascii) value of the character, its screen code value, and whether it's from the upper or lower case set. The row byte values for the character is a useful feature for programmers, but I think it would be more useful if it were possible to save them to disk as data statements for use in BASIC

programs. Underneath the character grid, the character is displayed in the 16 colors of the C128, in under lined video, reversed video, and the character in the same position of the alternate character set.

For editing, Fontigus shifts to a different screen with the editing commands shown. The character being edited is displayed in differing formats such as normal, inverted, flipped, or with its quadrants exchanged, so you have a visual reference of the effects of these options. Using a single command, characters can be mirrored left or right, flipped, inverted or rotated. They can also be shifted by line or row, both vertically or horizontally.

Fontigus can save the lower, upper, or both character sets together, and also prompts whether to save them in short or long form (with or without null zero padding). This is a good feature, as I've used other character editors which don't offer any choices about how the sets are saved.

There is a good directory viewer showing two columns of files at a time with a pause between multiple screens. It would be nice if the character sets could be loaded while viewing this directory. However, the user has to type the name of the set without this luxury. Fontigus supports DOS commands, but you can only send one command at a time and it prompts for the disk drive number each time. The author indicated he would improve this in the next version.

If you've been looking for an 80 column character editor, Fontigus 128 offers extremely powerful features. Projects I started with other editors, but put off due to difficulties, were easily completed using Fontigus. In the course of a couple days, I edited over 200 character sets without realizing it.

*- Gaelyne R. Moranec*

# TURBO ASSEMBLER

*Turbo Assembler; £9.95 plus £1.00 shipping. Turbo Cross Assembler; £19.95 plus £2.00 shipping. Available from Electric Boys Entertainment Software, 2A Woollaston Road, Harringay, London N41SE; Phone and Fax: 0181-348-4916.*

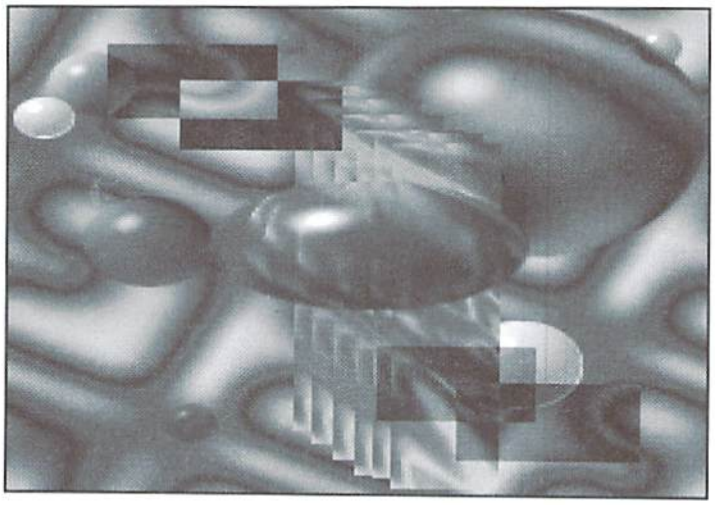
A friend said to me the other day "I didn't know you still used a Commodore 64 computer. What do you do with it besides use it as a great big calculator?" He chuckled at the last statement, but was quieted when I informed him that I can utilize the Internet, send and receive faxes, create publications, balance my checkbook, catalog my belongings, and play better games than the kids down the block with their Nintendo. My friend was stunned. He wanted to know how I could accomplish so much with so little.

You and I know it is the work of talented programmers. However, behind each talented programmer is a set of useful tools for the job. In the case of machine language programming, I know many talented programmers that use the following tool: The Turbo Assembler machine language assembler.

Turbo Assembler is an assembly language development system that runs on the Commodore 64 or 128. It will create executables suitable for any Commodore platform, and the language is similar to other assemblers, so someone can migrate from another assembler to this one without much trouble. The introduction quote sums up the purpose of this assembler: "After many years of honing and integrating, Turbo Assembler is finally available for use by people of all levels of programming ability."

The product comes on a 1541 formatted disk, and contains separate versions of the assembler for the C64 and the C128 (in C64 mode). The two versions are similar, but the C128 version will take advantage of the 2 MHz speed of the 128 CPU during assembly. In addition, the disk contains a C64 and C128 version of the assembler designed for cross-assembling.

Turbo is an integrated assembler. The program is loaded and started with a SYS 36864, which brings the user to the editor. In both versions of the assembler, the screen is 40 column and does not scroll side to side, so you always see the entire source code line on screen. The full complement of assembler opcodes are supported, as are pseudo opcodes like .BYTE, .WORD, and .TEXT. Labels and the #> and #< operators are supported, as is mathematical expressions as arguments for opcodes. The full complement of hexadecimal, binary and decimal values are permitted for assignments and arguments, and the familiar ";" character signifies comments. On the surface, the assembler seems just "normal".



After the program is started, it pops into the editor with a status bar at the bottom of the screen. Along with row and column position information, the line indicates whether the insert character mode is on or off and how much memory is left in the source code buffer. Command mode is entered by hitting the back arrow key, which enables access to the rich set of commands. Among the commands available are: assemble to disk or memory, load and save, block editing commands, simple arithmetic aids, memory dumps to source codes and memory fills. More esoteric commands include ones to change the color scheme, insert a row of dashes as a comment line, and redefine the function keys.

I tried compiling a small program with the assembler and noted that the integrated editor attempts to catch mistakes while you are entering a line of code. If I entered an illegal opcode or used the wrong addressing mode, the editor flagged the line as illegal by changing its color and noting the error on the status line. Assembling my program was but a keystroke away, and the program could be run immediately. After completion of the program, the assembler can be re entered by typing sys 36864.

In order to maximize performance and space, Turbo Assembler stores source code on disk in a proprietary format. However, for those people wishing to use existing source code file or wanting to save full listings of source, the editor has the option of reading or writing a PETSCII source code file. For the large software developer, one of the editors commands will save a copy of the labels and locations to a separate file, useful for multiple file source code development. But best of all, the assembly time is quick; further speeded up by the blanking of the screen during assembly.

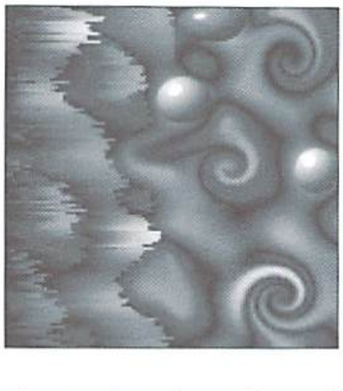
Although the copy of the assembler used for this review has the cross-assembly software, the required machine to machine cable was not available, so this feature was not tested. Nonetheless, the process is very straightforward. Load the assembler on one machine, while loading the small loader program on the other machine. Edit the source code in the editor, and assemble as usual. When the program is started, the results will show up on the remote machine. This alleviates the problems associated with having an error in the program under test destroying the development environment.

After using the assembler for many samples, I learned about many of its features. It has impressive parts, but unfortunately suffers from some major limitations. Every assembler has some limits, but these seem a bit constraining:

The assembler allows labels and function names to be used in source code, which is good, but each name must be in lowercase. Many developers use mixed case to improve readability, while some definitions, like TRUE and FALSE, are sometimes in all caps to distinguish them. The editor only allows 40 column lines. This seems adequate at first, and it helps allow a programmer to see the entire line on screen at once, but on-the-line comments can all too quickly reach the 40th column. The manual that accompanies the software is sparse at only 21 pages. While this won't affect intermediate to advanced programmers who have used an assembler before, the beginning user will find the going rough, and even experienced users will have to experiment with the more advanced editor commands. The assembler does not support any type of file includes or multiple file assembles. The programmer can manually assemble different files and concatenate them, but the assembler won't automate this process. This becomes a problem when you realize that the source code under development must fit between $0801to $8FEA.

The tone of the manual suggests that this assembler is ideally suited to demonstration or "demo" code development, which could be true. I see the program as providing a good assembler for testing small pieces of code or subroutines. The beginning user will like the error detecting editor, as well as the integration between the editor and the assembler, if they can cope with the user manual. After using the assembler for a few days, a few nit-picky items cropped up that should be noted. Some are just mere oversights, but any serious developer knows even the smallest problem with the tool can make the process seem longer and more frustrating.

Variables and labels can only be 15 characters in length. The editor is nice enough to note this as it occurs, but some programmers may find this a limitation. The editor includes a nice feature to allow the cursor to automatically tab to the 10th position of an inserted line, place opcodes at that position, and put labels at the far left, making the code more readable. However, I found no way to change the position to anything other than 10.



There is no native C128 version. Being a rather staunch C64 developer, it doesn't bother me that much, but C128 programmers will likely find the 40 column screen and lack of support for the extra memory in the C128 a significant limitation. After using the editor, I exited out of the editor to BASIC, only to find my cursor was invisible. Maybe it's just me, but I couldn't get the assembler to allow reads or writes to any drive besides device 8. Surely a command exists to change the data drive, but the usual disk commands didn't work, and the manual didn't offer any suggestions either.

Some of these problems can be corrected by a new release, while others could probably be fixed by updating the documentation. Nonetheless, some may find these small oversights to be extremely annoying.

Turbo Assembler is an interesting product. The version that comes on disk contains regular and cross-assembler versions of the assembler, but the manual implies that a version which allows macro definitions and a version which allows an REU to be used as extra RAM for program development both exist. Yet, these two capabilities are not present on the version supplied. In addition, the initial quote claims that the product has underwent "years of ... integration...", but these two innovations are not in the product. One can only guess how useful these two features are.

My recommendation is that, while Turbo Assembler seems like a product with promise, software developers looking for an assembler should buy this product with some hesitation. Some people will find its limitations a non-issue, but some serious developers will find the product lacking in support for large projects. On the other hand, the ease of using the product makes it ideal for small projects or prototype work, and the ability to cross assembler can save valuable time in the software development cycle. Caveat Emptor!

*- Jim Brain*

## Winter Clean-up Sale!

**This year at CMD, we neglected our Spring Cleaning. Now we're cleaning house and giving you some great deals! Call our order line at 1-800-638-3263 now; product quantities are limited!**

| | |
|---|---|
| * = Games 64 | + = Book 64 |
| • = Games 128 | ↓ = Book 128 |
| † = Productivity64 | Δ = Productivity 128 |
| ‡ = Productivity 64/128 | « = Misc. Hardware |
| ◊ = Printer | ♣ = Drives |
| ✓ = Manual (Computer) | ♠ = Manual (Drive) |

**$3.00**

« 1660 Modem (300 Baud)

**$5.00**

→1001 Things to do with your C-128
+ 32 BASIC Programs for C-64
* 4th & Inches
* Ace of Aces
* Adventure Construction Set
* Alice in Wonderland
* Arcade Megahits - Vol. 1
* Archon
* Archon II
* Arctic Fox
* Battle of Normandy
* Bits Pieces & Clues
* Blue Max
+ C-64: Getting the Most from It
+ C-64: Programs for the Home
* Championship Gambler
* Chernobyl
* Clowns Cartridge
* Clue - Master Detective
* Comics
+ Commodore 64 Games Book 2
+ Commodore 64 Game Book
* Countdown to Shutdown
* Cut & Paste
* Defender of the Crown
* Delta Drawing Cartridge
* Dragon Wars
* Dragons Lair
+ Elementary C-64
* Fight Night
+ First Book of Games for C-64
* Hacker
+ Hands on BASIC for the C-64
* Hardball
* Hawaiin Odessey Scenery Disk
* Hitchhikers Guide to the Universe
+ How to Program your C-64
* Infiltrator II
* Kennedy Approach
* Killed Until Dead
* Knight Games
* Kronis Rift
* Lode Runner
+ LOGO Workbook
* Lords of Conquest
† Magic Desk 1 - Type and File
* Mail Order Monsters
* Movie Monsters
* Murder Party 64
* Ocean Ranger
* Omega
† Omniwriter w/ Omni Spell
* Operation Wolf
* Outrun
* Phantasie II
† Printed Word & Calc
* Project: Space Station
* Raid Over Moscow
* Renegade
* Rootin'Tootin Cartridge
* Sargon II
* Sargon III
* Serve & Volley
† Simons BASIC Cartridge
* Sky Fox
* StarTrek - The Promethean Prophecy
* Super Cycle
* Temple of Asphai
* The Dolphins Rune
‡ The Electronic Checkbook
† The Filer
+ The Hackers Handbook
‡ The Home Manager
‡ The Kitchen Manager
‡ The Money Manager
* The Music Studio
† The Planner
* The Seven Cities of Gold
* The Shadows of Mordor
* The Slugger
* The Train
* The Wizard of Oz
‡ The Writer
+ Things to do with your C-64
* Ticket to Washington, DC
• T rinity 12
† WordWriter 3
* Worms
+ Your Commodore 64

**$8.00**

* Advanced Dungeons & Dragons-Dungeon Masters Assistant
* Advanced Dungeons & Dragons-Pool of Radiance
* Bards Tale II - The Destiny Knight
* Bards Tale III- Thief of Fate
* Bridge Made Easy
↓ C-128 BASIC
+ C-64: Graphics & Sound Programming
* Carrier Force - Santa Cruz, Midway
* Carrier Force - Solomons, Coral Sea
* Carriers at War 1941-1945
* Chuck Yeager's Advanced Flight Simulator
+ Commodore Collection Vol. 1
* Decision in the Desert
+ Flight Simulator & FSII 82 New Adventures
* Flight Simulator II
* Flight Simulator II - 64
* Gunship
* Hunt for Red October
+ Illustrated Guide to the C-64
* Indiana Jones - The Lost Kingdom
* Jack Nicholas Greatest 18 holes
* JET - 64
+ JET - 82 New Challenging Adventures
+ Kids and the Commodore 64
* Panzer-JAGD
* PHM Pegasus
+ Programming the C-64
* Project Stealth Fighter
* Redstorm Rising
* Silent Service
* Steel Thunder - Tank Simulator
* The Bards tale-Tale of the Unknown
* The Chess Master 2000
+ The Commodore 64 Users Guide
↓ The Essential C128 Users Guide
↓ The Official Book for the C-128
* Ultima IV
* Ultimate Wizard
+ Using & Programming C-64
↓ Your C128

**$10.00**

« 1350 Mouse
♠ 1541 Manual
♠ 1541 IIManual
♠ 1541C Manual
♠ 1571Manual
† BASIC - A Tutorial
✓C-128 Manual
✓C-128D Manual
✓C-64 Manual
† Certificate Library
† Clip Art Collection
† DataManager 2
† Fleet System 2
† Font Master II
Δ FontMaster 128
Δ Jane 128
« M1 Mouse
« Navarone Cartridge Port Expander
† Paperback Filer
✓VIC20 Manual
Δ WordPro 128

**$15.00**

« Aprospan Cartridge Port Expander
+ C-64 Programmers Reference Guide
Δ DataManager128
Δ Partner 128 Cartridge
Δ Pocket Writer 128
† Print Shop
Δ Sylvia Porter's Personal Finance
‡ Sylvia Porter's Personal Finance
† Vizastar Cartridge
Δ WordWriter128

**$20.00**

↓ 128 Programmers Guide
« 1351 Mouse
◊ CBM 1526

**$25.00**

◊ Epson Homewriter
◊ Okimate 10
◊ Okimate 20

**$30.00**

◊ MPS 801
◊ MPS 803

**$35.00**

♣ BCD-5.25
♣ Enhancer 2000

**$45.00**

« Light Pen 170C
« Light Pen 184C

**$50.00**

♣ FSD-2 Accelorator Plus
♣ Indus GT
◊ MPS 1250
◊ Okidata 120

All software tested and repackaged. All disk drives tested. All other hardware untested but in good condition. All products sold as is. $15.00 minimum order in effect.

## AMIGA

**Repairs • Upgrades • Sales**

*Factory Trained Techs • Flat Rate + parts • 90 Day Warranty*

**Buy • Sell • Trade**

*Amiga Computers • Monitors • Accessories*

**BSP**

HARD-TO-FIND ITEMS

CLOSE-OUTS

**Voice 908 245-1313**

Fax 908 245-9409

WANT MAILINGS? FAX NAME • ADD • PHONE • FAX

## ESCAPE ROUTE

The Adventures of Eric Hawthorne, P.I.
A combination of action, board, and strategy games. Help Eric track the infamous ICELADY who has stolen the rare Acme Diamond! Fun for everyone! For the C-64 or C-128 in 64 mode.

Send check or M.O. for $19.95 to CREATIVE PIXELS, P.O. Box 592, Library, PA 15129

## USED SOFTWARE

We buy, sell, and trade used original software. Lowest prices for C64, C128, Amiga, and IBM. Mention this ad and your computer type for a free list. Call our BBS for a complete list within minutes.

BBS:
8N1 1200-28.8k
616-429-7211

Ask for a list via Internet:
CENTSIBLE@DELPHI.COM

Call or write:

**CENTSIBLE SOFTWARE**

P.O. Box 930
St. Joseph, MI 49085
Phone: 616-428-9096

## *All Aboard the Commodore Information Supersubway!*

The Underground is a grass-roots Commodore publication produced with 8-bit computers, Geos, and a Postscript laser printer. For a mere $11 per year, you'll get helpful features, current reviews, and columns on everything from projects to PD software. A sample issue of the Underground is only $2--why not give it a try?

*THE UNDERGROUND, 4574 Via Santa Maria, Santa Maria, CA 93455*
Can/Mex: US$2.25/sample, $12.50/one year. International: US$3.75/sample, $21.50/one year.

## COMMODORE COUNTRY

Everything for Commodore Computers
Sell • Trade • Repair • Buy
1420 County Rd. 914
Burleson, TX 76028
817-295-7658
817-447-6974 - Voice/FAX line

ALL STORE ITEMS HAVE A 90 DAY WARRANTY. IF, FOR SOME REASON YOU ARE DISSATISFIED IN YOUR PRODUCT, YOU MAY RETURN IT FOR REPLACEMENT OR STORE CREDIT ONLY. SORRY, NO REFUNDS.

C64/1541 Repair - $40
*SPECIAL*
C64, 1541, J-S+Sft. - $99

We carry a full line of hardware, software & magazines, both new & used, including European items. Our flat-rate repairs include both parts & labor. Call for details. Trade in your unwanted items.

Catalog - $2.95
ALL MAJOR CREDIT CARDS ACCEPTED.

We carry a full line of Amiga Products.

INTERCORP COMMUNICATION **Presents**

**CSOFT Wares!**
**{As Seen on Many BBS's}**

**Digi<-->Dox V1.0**
**a SEQ. file reader that will read text as well as art,and supports drive's 8-11.It also has many Digi-sounds which are very entertaining.**
**$7.00 + $2.00 S&H**
**D.S. II "Cartoons"**
**10 of your Favorite Cartoon Digi Samples on one disk ,with a very easy to use Menu System for loading and playing.**
**Great For the Kids! $3.00 + $2.00 S&H**
**Send orders to**
**Intercorp Communication 466 W.Harwood ,Madsion Hts ,MI ,48071**
**Alaska,Hawii,& International add $1.00 to S&h**

Bounce around mazes fixing bricks and gathering bonuses while avoiding obstacles in **Pogo Stick**, a multi-level arcade style game for the 64. Fix all the bricks before the hour glass empties to go to the next level.

**Super Pogo Stick** Adds:
- 25 different mazes (with the ability to use even more)
- More obstacles & bonuses
- Ability to Save and Load games to disk
- High Scores

**Either game only $11.95**

**Both games for $19.95**

To order send a check or money order (in US funds) to:
**Yanney Software**
**P.O. Box 224**
**Lebanon, PA 17042-0224**

Both games are for a standard Commodore 64 or 128 (in 64 mode), a joystick is optional.

* Free shipping in US and Canada, other countries please add $3.00 to order.
** Pennsylvania residents please add 6% sales tax.

## Computer Bargain Store

**(801)466-8084**

**Specializing in NEW and USED Commodore Hardware, Software and Accessories at excellent prices.**

**Send $1.00 for a HUGE list of products. Office Hours: 11:30 - 6:30 MST. Visa, MC, Discover and American Express accepted.**

**3366 South 2300 East, Salt Lake City, UT 84109**

# Graphic Interpretation

*by Steve Vander Ark*

## GETTING STARTED WITH GEOBASIC: PART 2

In the last installment of this column, I started talking about writing programs with geoBASIC. I gave you an outline of a geoBASIC program using an "event-driven" structure. Then I talked through the creation of several of the components from which our program would be built. I ended the column by saying that "everything will plug right in, just like an electronic Lego set." But I also warned you that geoBASIC can throw you for a loop now and then.

So, this time we'll review the structure of our program and write out the geoBASIC code itself, and I'll also introduce you to some of the pitfalls you should avoid.

Here's the program outline from last month's column with my notes added in parentheses:

@titleScreen - a routine to create the title screen with two buttons saved as "icons1":
  title1 - ties to @quit
  title2 - ties to @mainScreen

(This is a subroutine; the name is set apart from other geoBASIC commands by the @ symbol. This screen will have two buttons on it which will, when clicked by the mouse pointer, jump the program to routines called @quit and @mainScreen. The label names of routines can have upper and lower case letters in them, as you can see. Notice that I called the buttons "title1" and "title2." These names show up in the icon editor where I create the buttons themselves.)

MAINLOOP
The program waits at this point for a button or menu item to be selected. The rest of the program will consist of the routines which those buttons and menu items call up. (This is the heart of a geoBASIC program. This is what actually makes the program act like a "real" GEOS program.)
@mainScreen - a routine to draw the main program screen and set pattern and color to their starting values and display the following buttons and menus:

- a drop down menu saved as "mainM" with three submenus
- a "geos" submenu where available desk accessories will be listed automatically
- "program info" - tied to @about
- a "file" submenu with two choices:
  "quit" - tied to @quit
  "start over" - tied to @mainScreen
- an "attributes" submenu with two choices:
  "color" - tied to @colChoice
  "pattern" - tied to @pattChoice
- four icons saved as "icons2":
  draw1 - tied to @drawRect
  draw2 - tied to @drawLine
  draw3 - tied to @sayHi
  draw4 - tied to @scrnClear

(This is the section that actually creates the main screen of our program. Notice all the bits and pieces that are mentioned. In this plan I have included all have names from the editors used to create these pieces, including the various subroutines. This information is very important to have planned out ahead of time, but most of it never appears in your program itself, just in the editors. What follows this are the routines themselves.)

@quit
  ends the program and automatically sends the user back to the deskTop
@colChoice
  changes the value of the background color
@pattChoice
  changes the value of PATTERN for rectangles
@drawRect
  draws a rectangle on the screen
@drawLine
  draws a line on the screen
@sayHi
  places the word "HI" on the screen
@scrnClear
  asks with a dialog box saved as "clrD" with two buttons
    "yes" which clears the screen
    "no" which returns doing nothing
@about
  reports the author's name in a dialog box

(Now if you want to double check, look back through the plan and see where each of these routines is called. Some are hooked to icons, others to drop-down menu selections. They could appear here in any order, of course. When an "event" occurs—when a menu selection is chosen, for example—the program will go to whichever of these routines applies.)

Okay, now that we've walked through the structure of this program, let's turn it into actual geoBASIC commands. Last time we worked on creating some of the building blocks—the drop down menus, the icons, and what have you. Now this time we'll plug them into the program framework.

**10 REM geoBASIC program by Steve Vander Ark**

```
100 @titleScreen
```

(Notice that we still use line numbers, even though the structure is not very linear.)

```
110 CLS
120 PATTERN 10
130 RECT 80,50,290,170
140 PATTERN 1
150 RECT 60,30,270,150
160 PATTERN 0
170 RECT 61,31,269,149
```

(After I cleared the screen in line 110, I created a series of rectangles with the RECT command. Before each one I set the pattern inside the rectangle to a different value. This is a quick way to create a nice-looking title screen. Now all it needs is some text.)

```
180 WINDOW 110,75,269,149
```

(This is a window for my text to go into. I placed it inside of the rectangles I already drew.)

```
190 PRINT" geoBASIC Program"
200 PRINT"by Steve Vander Ark"
```

(Now we have a title screen. It isn't fancy, but it will do. Now we have to add a couple of buttons so the user can go on to the program itself.)

```
210 ICON"title"
```

(This command places a set of icons on the screen. These icons were created using the "icon editor" under the utilities menu of the geoBASIC screen. The images for the icons were created by the "bitmap editor" and were called "title1" and "title2." The routines they call were also specified at that time. All of this is specified in my plan above. Now, once all that was set up for us, our program gets a nice set of icons, all ready to use, with just this one simple command. That's what I meant when I said that everything would plug in like an electronic Lego set.)

```
220 MAINLOOP
```

(Here we are. Our program is set up and now is waiting for something to happen. At the moment, the only possible events are the ones tied to those two icons we just placed on the screen. One of them will move our program into its main screen, which we'll design next.)

```
300 @mainScreen
```

(That's another label. This routine will be called when someone clicks on one of our icons on the title screen.)

```
310 PATTERN 0
320 SETCOL 15
```

(Before the program starts, I want to set these values to their base number. That way if someone changes them during the course of the program and then starts the program over, the changed pattern and color won't be in effect anymore. The SETCOL command is actually more complicated than it looks. I am using this formula: 15+16*0 . That means my foreground color will be 0, which is black, and my background color will be 15, which is light gray.)

```
330 CLS
340 REM MENU "
350 REM ICON "
```

(These two commands place a drop down menu at the top of the screen and a new set of buttons on a toolbar at the top. Again, these were all defined using the editors, so now we just have to plug them in. All that's left for us to do is to actually define our various routines.)

```
400 @quit
500 @colChoice
600 @pattChoice
700 @drawRect
800 @drawLine
900 @sayHi
1000 @scrnClear
1100 @about
```

Before we get those fleshed out, let me fill you in on a few possible trouble spots. First of all, don't use the "update" selection under the "file" menu on the main geoBASIC screen. Doing so will damage your file. Second, be extra careful to write down the names of any and all components of your program. Remember that upper case or lower case letters must be taken into account. The reason for this is that when you try to open an existing bitmap, icon list, drop down menu, or other editor file, the requester box doesn't really list any of the applicable files from the disk. Those files are there, but you need to backspace over the word that appears on the dialog box by the cursor and type in the name of the file you really want. You will need to have the exact filename written down somewhere because the files don't show up in the directory on the deskTop either. Careful planning in advance makes it easier since you can name everything beforehand and make a list for reference. Planning ahead is also handy when you need to assign routines to various objects you create in the editors since you'll know exactly what the labels for those routines will be.

Next time we'll get to the rest of our discussion on geoBASIC and the rest of this program listing.

# Carrier Detect

*By Gaelyne R. Moranec*

## SPINNING THE WEB

When I first wrote about the World Wide Web, I mentioned that in a broad sense, it is a form of advertising in which the consumer goes to the advertising source for information about a company or its products. It has gained in popularity not only among "mainstream" computer users and advertisers, but also by Commodore users and those who in one way or another support us. Commercial companies, program authors and others have been adding CBM support to the Internet in a big way, and it's something you can have, too. This is one of the nice things about the World Wide Web— it's not limited to big commercial companies, nor is it only for those with PC's and Macs. As I explained in CW #8, we access WWW pages using an Internet utility called Lynx, which is a text based web browser. But we can do more than access WWW pages. We can create them!

What does it take to create your own web page? You'll need to be on an Internet service which allows you to create "home pages". If in doubt, ask the System Administrator. If your service provider doesn't offer this capability, there are numerous web presence providers on the web that offer reasonable rates for both commercial and non-commercial purposes.

Web pages are created using plain ASCII text and special formatting codes known as *HTML* or *HyperText Markup Language*. You may have noticed that some businesses are selling commercial PC and Macintosh packages which create Web pages and cost a lot of money. These really aren't necessary. HTML codes (or "tags") are just plain text and can be created in any word processor or text editor which allows you to save text in true ASCII format.

### What's Out There?

Before you jump into creating WWW pages, it helps to know what's already out there, how others have designed their pages, and the types of things they've included. The following list of

Commodore related websites serves two purposes. First, it's a place on the web to find Commodore support. Second, by seeing how others have designed their sites, you can get ideas on creating your own.

One of the most common things you'll notice is that most websites are linked to others. Many times it's a reciprocal thing—if you include a link to someone else's site, they in turn include link to yours. The places you include links to will depend on your interests and the overall content of your website.

There are so many web pages out there, it's hard to know where to begin. But, why not try a few of my favorite sites?

### Q-Link

*http://www.kaiwan.com/~sirfitz/qlink.html*
*http://www.portal.com/~steward/qlink.html*

When Q-Link closed its doors on October 31, 1995, I didn't think I'd miss it, but a recent World Wide Web tour of Q-Link home pages reminded me that what made Q-Link special was the people who made it fun to be there. People like Sir Fritz (Charles J. Fritzhugh), JohnD39 (Steward), and Squirrel's Nest (John Purkey) who have created a place on the Internet for those who enjoyed Q-Link and remember it fondly. These home pages are more than just a shrine to a "dead offline service", due in part to the interactive items they've added to their pages, like a "gRiFiTtI" (graffiti) wall to post short messages and a registry of Q-Link screen names and Email addresses so you can catch up with old friends. Browsing the list of names, it's interesting to note how many people have kept their screen names as part of their Internet addresses. Links to former Q-Link users web sites are also included. The Q-Link sites are integrated together in such a way that they are almost seamless. Nothing is repeated and everything is shared between the pages.

*http://www2.ari.net/home/jpurkey/qscreens.html*

This site has screenshots of the Q-Link load screens that were seen whenever moving to different areas on the service. These are available as a C64 program to download, and within this 63 block program there's a menu with all 37 screens for viewing. The screens can be downloaded (or viewed with a graphic browser) as individual GIFs or you can download the whole collection in a PKZip 1.x archive. Purkey's site also includes a short library of classic Q-Link files to download straight from the web.

Q-Link the service may have died a senseless and agonizing death, but Q-Link is (or was) not so much an online service as a feeling of belonging, and a mutual meeting place for many kindred spirits. The service is gone, but the spirit is alive and well in these web pages.

### SIDS

*http://stud1.tuwien.ac.at/~e9426444/index.html*

I believe that SID music never would have gained its popularity if it weren't for Q-Link and other online services. I'm sure the ability to share ones creations with others contributed to the strong popularity. The SID home page has samples, interviews with composers, a history of SID music, and you can even vote for your favorite tunes. Links to other SID related sites can also be found here.

**Creative Micro Designs, Inc. & Commodore World**
*http://www.msen.com/~brain/guest/cmd/index.html*

You can read about how CMD got started and how it's progressed over the years by selecting "More About CMD" from their home page. There is an online sample of this magazine (I have two articles in the online issue). You can browse the "Table of Contents" and read articles online. CMD's products are highlighted in their Product Information pages, and to make looking up specific items easier, they are grouped by category. This area was under construction when I visited, but may be completed by the time you read this. To help users keep up with the current prices, CMD has a dated Price List of the items they carry and their shipping charges. From CMD's home page, you can also connect to Jim Brain's Commodore WWW Links or the popular Yahoo Index of fun places to visit on the net.

**Softdisk Publishing and Loadstar**
*http://www.webcom.com/~softdisk/c64.html*

Loadstar's home page has information about the two disk magazines published by Softdisk Publishing: *Loadstar* and *Loadstar 128 Quarterly*. When I visited, the software products list had one item, "The Compleat Series", which didn't have a description. Hopefully it will be more "compleat" later. The site has ordering information for the magazines, and software items, with specials for those ordering items from the website. A sample "Table of Contents" shows the type of programs and articles which normally make up an issue of *Loadstar*. Several articles are also available to read, including programming columns ("When is Your Program Finished?") interviews ("Interview with CMD"), reviews ("Dr. Synth Tone Generator"), and others ("Miscellaneous Hints & Tips"). This site has several links to other Commodore related web sites.

**Commodore Format Magazine**
*http://www.futurenet.co.uk/computing/commodoreformat.html*

*Commodore Format* magazine is a British publication. Its home page has a table of contents for recent issues as well as a list of software which they sell. While a bit on the spartan side, its link to FutureNet Computing News makes up for it. FutureNet Computing News offers some of the best computing news available. You can get the latest computer news from the net sometimes weeks or months before you see it in print.

**Software Support International**
*gopher://gopher.soonet.ca/11\COMPUTERSMITH*
*http://www.soonet.ca/~compsmth*

Computersmith offers most of Software Support International's Summer '95 Commodore/Amiga catalog via Gopher, and possibly via WWW by time this article is published, which is why I've included them here. When I asked Computersmith's owner, Allen Smith, if the catalog was sanctioned by Software Support International, he told me "SSI supplied the text to me (as a WordPerfect file) and gave me permission to use it in the way I have. I, in turn, am doing this both as a service to all Commodore (8-bit and Amiga) users, as well as to try and generate a little business for myself. My business being a Canadian Dealer for SSI products." The SSI catalog and other information can also be retrieved via E-mail. To find out more, send a message to info@compsmth.soonet.ca.

**Computer Workshops**
*http://www.armory.com/~spectre/cwi.html*

Computer Workshops is a software company which supports the Commodore 8-bit line as well as MSDOS. Commodore software highlighted at this site are "Flyer" and "MahGong" which are commercial games available directly from the company. Descriptions of the games are given, along with review comments from the now defunct Gazette magazine. Two shareware releases, "NewView" and "HyperLink" are described. The programs are available via FTP, but the FTP site wasn't (at time of research) linked into the web site.

**CNet 64 DS2 BBS**
*http://www.infinet.com/~mbendure/*

Michael Bendure's CNet 64 DS2 BBS home page includes personal background about the man behind the networking software as well as support for his BBS software, the network it uses (DS2 Network v3.04), and ComNet development. ComNet is a dream of Bendure's in which all Commodore BBS's (regardless of software) have an integrated network so they can share messages and files between them. Support for registered CNet 64 DS2 BBS sysops is planned, but in the meantime, those interested in learning about the bulletin board system can check out the features and hardware it supports. This site also has Commodore hardware and software listed for sale and links to other CBM related software.

**RMS Computer Systems (CNet 128)**
*http://www.msen.com/~brain/guest/rms/*

When I visited the RMS Computer Systems home page, it was still under construction. RMS Computer Systems sell and support the CNet 128 BBS system and computer systems parts and accessories. RMS plans to have a support area within the web site so registered sysops can get updates, documentation and other support regarding the operation of the CNet 128 Bulletin Board System. As of this writing, this site has a list of BBS features and hardware requirements as well as information about how to reach the Cave Of Cerberus BBS, which is C-Net 128's support Board.

**Jim Brain's Commodore 8-bit Site**
*http://www.msen.com/~brain/cbmhome.html*

Jim Brain's site has become one of the major US places to check out Commodore developments. Most Commodore related web links can be found here and the links are continually updated. Features here include the ability to read the comp.sys.cbm FAQ file in HyperText format and links to other Commodore related FAQ files such as comp.binaries.cbm and the C64 emulator newsgroup. You can find the answers to the Trivia questions in this magazine , read issues of *C= Hacking* (a Commodore magazine on the net) or you can learn about Jim's business, Brain Innovations, Inc.

**Craig Bruce's ACE and LLR**
*http://ccnga.uwaterloo.ca/~csbruce/index.html*

Craig Bruce is a Canadian programmer. His website has a mixture of information about himself and about some of his popular programs like ACE, Little Read Reader and others. He has links to the FTP site where his programs can be downloaded, and lists of other places of interest to Commodore users. Craig also has a few UNIX utilities which he's written, and help files for UNIX Vi and EMACS editors on his site.

**Rod Gasson's QWKRR128**
*http://www.msen.com/~brain/guest/Gaelyne_Moranec/qwkrr/*
*http://www.msen.com/~brain/guest/Gaelyne_Moranec/qwkrr/browser.html*

Rod Gasson doesn't have a home page, but two of his programs, QWKRR128 and Browser, do have a home on the web, under my wing. When I set up my personal home page, I decided to add support for QWKRR as an experiment in using the World Wide Web to show how it could be used to offer support for shareware and commercial programs.The QWKRR pages consist of updated information about support files which weren't available when QWKRR128 v4.3 was released. You can even register QWKRR while on the Web. The Browser pages let you become a Beta Tester; you can download the work "in progress" of frequently updated versions right from the World Wide Web.

**Gaelyne Moranec's Home Page**
*http://www.msen.com/~brain/guest/Gaelyne_Moranec/*

I've included links to most if not all of the web pages listed here, FTP links to Commodore files, a type-in term program that a reader in Italy asked me for, on-going lists of favorite CBM programs, BBS's which support us, a Guest page to sign, and links to my articles in *Commodore World's* online sample magazine issue. (Oops, my modesty just left me again!) A special site on my page has been added: a tutorial on HyperText, and examples of magazine pages on the web. The URL for the tutorial is *http://www.msen.com/~brain/guest/Gaelyne_Moranec/learn.html*.

## World Wide Web Alternatives

### Web Surfing via Telnet

For those using online services who need to telnet to use Lynx, this is telnet site allows you to use the "Go" command to go to any of the URL's listed in this article:

Telnet to: *fatty.law.cornell.edu*

At the first prompt, type "www". Don't forget to set your terminal emulation to VT100 (or VT102). GEnie users should turn off echo before beginning the telnet session.

### Web Surfing by Email

If your only access to the Internet is via Email, you can receive web pages by Email. Simply send a message with no subject line to *webmail@curia.ucc.ie* on the first line of the message type the word "go" followed by the URL you wish to view, such as:

*go http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html*

## URL Listings in Carrier Detect

**Q-Link**
*http://www.kaiwan.com/~sirfitz/qlink.html*
*http://www.portal.com/~steward/qlink.html*
*http://www2.ari.net/home/jpurkey/qscreens.html*

**SIDS**
*http://stud1.tuwien.ac.at/~e9426444/index.html*

**Creative Micro Designs, Inc. & Commodore World**
*http://www.msen.com/~brain/guest/cmd/index.html*

**Softdisk Publishing**
*http://www.webcom.com/~softdisk/c64.html*

**Commodore Format Magazine**
*http://www.futurenet.co.uk/computing/commodoreformat.html*

**Software Support International**
*gopher://gopher.soonet.ca/11\COMPUTERSMITH*
*http://www.soonet.ca/~compsmth*

**Computer Workshops**
*http://www.armory.com/~spectre/cwi.html*

**CNet 64 DS2 BBS**
*http://www.infinet.com/~mbendure/*

**RMS Computer Systems (CNet 128 BBS)**
*http://www.msen.com/~brain/guest/rms/*

**Jim Brain's Commodore 8-bit Home Page**
*http://www.msen.com/~brain/cbmhome.html*

**Craig Bruce's ACE and LLR**
*http://ccnga.uwaterloo.ca/~csbruce/index.html*

**Rod Gasson's QWKRR128**
*http://www.msen.com/~brain/guest/Gaelyne_Moranec/qwkrr/*

**Gaelyne Moranec's Home Page**
*http://www.msen.com/~brain/guest/Gaelyne_Moranec/*

**The tutorial on HyperText**
*http://www.msen.com/~brain/guest/Gaelyne_Moranec/learn.html*

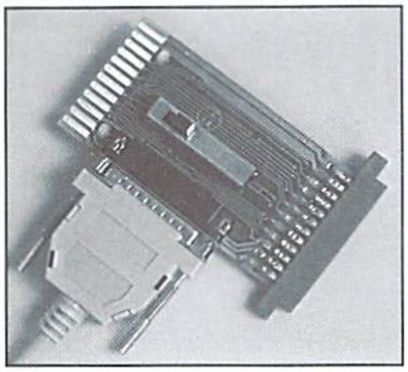
**Web Browse**
*http://www.beta.yahoo.com/*

**FTP files from ccnga.uwaterloo.ca**
*ftp:get//ccnga.uwaterloo.ca/pub/cbm/telecomm/*

**CompuServe Telnet Session**
*telnet://compuserve.com*

**HTML Guide**
*http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html*

## GEOCABLE II
### Parallel Print Cable



**The Most Advanced Parallel Print Cable Ever!**

- **Uses Standard Cable** Equipped with a female DB-25 cable connector to allow use of standard 'PC' printer cables.
- **Switchable Pass-thru** Allows connection of printer and other user port peripherals such as modems without conflicts.

GEOCABLE-II (6 ft.) ONLY $29.00
GEOCABLE-II (15 ft.) ONLY $34.00
Shipping: US $5.00, Canada $7.00, Foreign $25.00.

- **Faster Output** Parallel printing offers up to a 40% increase in output speed.
- **GEOS Drivers Included** Includes GEOS drivers plus useful GEOS utilities like MacAttack II, WrongIsWrite, and GEOS CONVERT.

GeoCable II is compatible with GEOS, Perfect Print LQ, Action-Replay MK VI, Superbase, Superscript, Paperclip III, and all software that supports user port parallel printing.

CMD **Creative Micro Designs, Inc.**
P.O. Box 646
East Longmeadow, MA 01028
Orders: 800-638-3263
Info: 413-525-0023

## CW & RUN BACK ISSUES

For years, *RUN* Magazine provided Commodore Users with a great source of information, and now CMD has given you Commodore World. Don't let this valuable information slip away—fill in the voids in your library now!

### RUN Magazine Back Issues
**Any 3 issues for $12.00, any 6 for $18.00, or any 12 for only $24.00**

| | | | |
|---|---|---|---|
| RMJAN88 | January 1988 | RMOCT89 | October 1989 |
| RMFEB88 | February 1988 | RMSP89 | Special Issue 1989 |
| RMMAR88 | March 1988 | RMJAN90 | January 1990 |
| RMAPR88 | April 1988 | RMMAY90 | May 1990 |
| RMMAY88 | May 1988 | RMJUN90 | June/July 1990 |
| RMJUN88 | June 1988 | RMDEC90 | December 1990 |
| RMJUL88 | July 1988 | RMJAN91 | Jan/Feb 1991 |
| RMFEB89 | February 1989 | RMJAN92 | Jan/Feb 1992 |
| RMMAY89 | May 1989 | RMMAR92 | March/April 1992 |
| RMJUN89 | June 1989 | RMMAY92 | May/June 1992 |
| RMJUL89 | July 1989 | RMJUL92 | July/August 1992 |
| RMAUG89 | August 1989 | RMSEP92 | Sep/October 1992 |
| RMSEP89 | September 1989 | RMNOV92 | Nov/December 1992 |

**Shipping:** 3 or 6 issues - U.S. $3.00, Canada $5.00, Foreign $15.00; 12 issues U.S. $5.00; Canada $7.00; Foreign $20.00.

### Commodore World Back Issues
**$4.95 each, or any 3 for only $12.00**

| | | | |
|---|---|---|---|
| CW1 | Issue 1, Volume 1, Number 1 | CW6 | Issue 6, Volume 2, Number 1 |
| CW2 | Issue 2, Volume 1, Number 2 | CW7 | Issue 7, Volume 2, Number 2 |
| CW3 | Issue 3, Volume 1, Number 3 | CW8 | Issue 8, Volume 2, Number 3 |
| CW4 | Issue 4, Volume 1, Number 4 | CW9 | Issue 9, Volume 2, Number 4 |
| CW5 | Issue 5, Volume 1, Number 5 | CW10 | Issue 10, Volume 2, Number 5 |

**Shipping:** U.S. and Canada $2.00 for first issue, plus $1.00 per additional issues.; Foreign $5.00 per issue.

**TO ORDER CALL 1-800-638-3263**

## Need Input? It Doesn't Get Any Better Than This...

**Smart TRACK**

For years, Commodore set the standard with the 1351 Mouse. Sure, it was vastly superior to using the old digital input devices like the 1350 mouse, joysticks or imitation trackballs. But everything can be improved. Guaranteed 100% 1351-compatible, SmartMouse and SmartTrack do everything the C-1351 does and more! These highly intelligent, three-button input devices include a built-in battery-backed Real-Time Clock, along with double-click and Turbo features for GEOS. Plus, they come with a complete set of utilities for using the clock in GEOS and BASIC applications. Modern ergonomic designs make these devices smooth operators that are a pleasure to use. So, if you're tired of the slow, erratic movement of your current input device, make the intelligent choice and pick up a SmartMouse or SmartTrack today!

**Smart MOUSE**

### SmartMouse and SmartTrack... the Best C-64/128 Input Devices Ever!

- Three buttons means convenience! If you're a GEOS user, the left button is configured as single click, the right as a handy double click and the center button is the TURBO button. When depressed, it doubles the speed at which the pointer moves across the screen. Additionally, programmers can assign their own functions to all three of the buttons.
- Unlike other third party mice or trackballs, the CMD SmartMouse and SmartTrack trackball use the same custom gate array chip as the Commodore C-1351 mouse. This guarantees 100% compatibility.
- Switches easily into joystick emulation mode on power-up by holding down the right button.
- SmartMouse/SmartTrack utilize the same advanced technology used in today's powerful 486 and Pentium PC's, providing you with unparalleled accuracy and smoothness.
- Built-in battery-backed Real-Time Clock automatically sets the GEOS clock, displays time and can be used in your own programs.
- Includes utilities disk and detailed manual explaining the utilities and programming information.
- Attention Lefties! SmartMouse can be altered for left handed use.
- Don't be fooled by the old style digital "trackballs". These only perform as well as a joystick. SmartTrack uses analog inputs for greater accuracy.

# Over The Edge

*By Don Radler*

## VIRTUAL REALITY: A PLANNED DISASTER?

Technology, as you might have noticed, often has a few unexpected consequences.

ITEM: When Henry Ford figured out how to put each of us into one of his automobiles, he didn't plan on helping to launch the sexual revolution. But the freedom the automobile provided for many young couples, and all those back seats hidden from sight by sheets of steel were just too tempting.

ITEM: When Dave Lennox sold the public on putting his air conditioners into more and more rooms, he didn't mean to help criminals. Besides cooling off Mom and Dad inside (instead of on their front porches or stoops, leaving the streets un-watched) it was also easier for burglars to gain entry to the house by popping the units out of the windows. And there went the neighborhood.

ITEM: When Lee De Forest laid the groundwork for television, he wanted to beam culture into our homes. Look what we ended up with! Gangsta rap at breakfast, tabloid-talk shows and soap operas starting at lunch, feminine hygiene commercials and graphic scenes of warfare accompanying dinner, and pornography on into the wee hours of the morning.

Even smaller technological changes can have large effects. For example, we derived a sense of community and connection to each other when there were only three networks, and all of us were watching Ed Sullivan at the same time. The next day, in every office and factory and restaurant, last night's "shew" (Sullivan's much-imitated pronunciation of "show") was the topic of conversation. Now, with over 50 cable channels as the norm and 500 promised, what you looked at last night doesn't figure to be what I watched at all.

Similarly with Alexander Graham Bell's invention: the "someone" AT&T tells you to reach out and touch is more than likely going to be an answering machine, and MCI's

"friends and family" are usually not people at all, but the gadgets that speak and listen for them. In the absence of real, live people, community can't exist; the motivation for inventing the telephone has already been defeated by the way newer technology enhances its use.

Moving from analog cyberspace to digital, we come to the much-vaunted information superhighway. Here, there are literally thousands of Special Interest Groups (SIGS) or Round Tables (RTs), where people of narrow focus type messages back and forth in the belief that they are thereby broadening their horizons. Thus, the ever-expanding roster of online enthusiasts becomes not a community, but a collection of inward-looking electronic tribes. Ultimately, those tribes can be expected to do battle with one another over the available bandwidth, just as tribes are proving their humanity in places like Bosnia and Somalia today.

Finally, we arrive at the ultimate anti-community, virtual reality. Here, hardware and software conspire to bring you inside a computerized environment, experiencing it with several senses at once and responding to it as a participant rather than a mere observer. You may wear a headset or special glasses to see scenes in full-color 3-D and hear things in surround sound; you may wear gloves that act like a mouse or a joystick but also feed back the feel of things; you may even climb into a full body suit to generate these experiences. However, our understanding of human perception meshes with our ability to design hardware and software to fool ourselves. VR will pull us into the machine and make us experience simulated events rather than just see and hear them from outside. With this ability, we can play - or even learn.

Bennett Davis, writing in Discover Magazine back in June 1990, said this about it: "Many researchers see complete modestly-priced 'personal reality simulators' little more than a decade away, and they already worry about the consequences."

Davis quotes Thomas Furness, pioneer of simulators for the military: "It's not like television or a personal computer. With those, you're still on the outside. Once that field of view surrounds you and controls everything you see, you're inside. The social implications are of great concern to us."

The downside is the creation of socially immature people. Virtual realities will do what people want them to do, and that's not the way the real world works. This can be a tremendous medium through which to learn, but it can also hinder people from learning other real things in the real world. Are we ready for a world dependent on artificial experiences?

You could "virtually" strangle someone. You could have a virtual chain-saw massacre. You could use this to experience all kinds of expressions of deviant behavior, and through it people could become inured to violence or perversion in real life, much like children confused by the concept of make-believe versus real life. Many years earlier, Aldous Huxley foreshadowed this concept in his book *Brave New World*. Here's one example:

COULD IT BE THAT THE CONSEQUENCES OF THIS TECHNOLOGY WERE PLANNED ALL ALONG? IS OUR NEED FOR GADGETS AND SPEED WHAT THE DIGITAL FUTURE IS REALLY ALL ABOUT?

"Three weeks in a helicopter. An all-super-singing, synthetic-talking, colored, stereoscopic feeling, with synchronized scent-organ accompaniment."

Sounds more than a little like where VR is headed, doesn't it? And these quotes from a Huxley character just might have explained what's driving it:

"Industrial civilization is only possible when there's no self-denial...otherwise the wheels stop turning...Imagine the folly of allowing people to play elaborate games which do nothing whatever to increase consumption...We don't want people to be attracted by old things. We want them to like them to like the new ones."

Could it be that the unexpected consequences of this technology were planned all along? The proverbial "world run by computers"? Is our own refusal to deny ourselves anything in the name of "technological advancement" being exploited by the richest among us? Is our need for gadgets and speed and their greed for money and power what the digital future is really all about?

Available NOW

*Click Here Software's*

**geoFAX**

***Send and Receive faxes on your C-64/128***

**Features**

- **SEND & RECEIVE FAXES to/from any fax machine or computer running a fax modem**
- **User-friendly and easy to operate**
- **Auto-answer mode for unattended operation**
- **Manual send/receive mode lets you share phone line**
- **Built-in functions convert faxes to geoPaint files**
- **geoPaint files may be faxed without conversion**
- **Built-in printing functions for printing fax documents and geoPaint files**
- **Printer drivers may be selected directly from within the program**
- **Full-page scanning into geoPaint is possible using a standard fax machine as a scanner**
- **Built-in simple terminal program lets you browse telecommunications services**

**System Requirements**

- **Commodore C-64, C-64C, C-128, 128-D computer**
- **GEOS 64 or GEOS 128 version 2.0**
- **SwiftLink RS-232 cartridge**
- **Group III, Class 2 fax modem**
- **GEOS compatible disk drive (large capacity drive or RAM disk recommended but not required)**

**Prices & Special Bundles**

| | |
|---|---|
| *geoFAX* ........... *$39.95* | *geoFAX & SwiftLink* ........ *$75.00* |
| *SwiftLink* ........ *$39.95* | *geoFAX, SwiftLink & BOCA 14.4 Fax Modem* ......... *$195.00* |
| *BOCA 14.4 Fax Modem* .. *$129.95* | |

**Creative Micro Designs**

See our main ad in this issue for complete ordering information.

**1-800-638-3263**

# HARD TIPS

## CHANGING DISK DRIVE DEVICE NUMBERS

*By Doug Cotton*

One of the most often asked questions we get is how to change device numbers on disk drives. While the 1541-II and 1571 drives were created with DIP switches for convenient device number configuration, Commodore left this convenience out of previous 1541 drives, and the 128D. They did, however, at least make such changes possible directly on the circuit boards. Yes, you've got to get under the hood to change the device number.

Commodore also provided information in the 1541 User's Manual on making device number changes. Unfortunately, the instructions were wrong in some editions, and often didn't match the revision of the 1541 circuit board in the drive that the manual came supplied with.

So, after a little prodding, we've put together this illustrated guide to changing device numbers on each of the older 1541 models, as well as on the 128D. In addition, we've covered installing switches that will let you make device number changes whenever you like.

### Tools & Other Required Goods

As a bare minimum, you'll need the following items in order to change your 1541's hardware device number:

- A Phillips-head Screwdriver (#2 should work nicely)
- An Exacto Knife (or other precision utility knife or blade)

In addition to these items, I'd also suggest a magnifying lens of some type (for the close work), and a good light source. If you want to install switches, you'll also need the following:

- One or two SPST Miniature Toggle Switches (SPDT will work as well)
- A Soldering Iron or Pencil
- Solder (electronic circuit variety)
- Hookup Wire (28 gauge stranded)
- A small pair of Diagonal Cutters (Wire Strippers would be handy, too)
- A Drill and Drill Bit (to provide the hole for mounting the switches)

### Here we go...

We'll start off with disassembly. Start by removing the screws that hold the case together (these are located in recessed holes in the bottom half of the case on the 1541 models, and on the back of the 128D case). After you have the screws removed (and safely tucked away where you won't lose them), remove the upper half of the case. If you're disassembling a 1541, you'll need to remove the metal shield that covers the circuit board (there are two screws located on one side). Those of you with 128D's will have to remove the drive itself by removing the three screws that hold it in place (one on the left side, two on the right), pulling the lever off the front, and sliding the mechanism backwards into the case.

The next step for 1541 users is to determine which revision of the circuit board you have. There are three main types to be concerned with, and these are shown in Figures 1, 2 and 3. If your drive (like most) has the "short" board (Fig. 2), then check around the edge of the board for the revision level (A, B or C).

Now check Figures 6 through 10 to locate the Jumper Pads used to program the device number for your unit. Jumper Pads look like two silvery circles, each having a split down the middle, except at the very center where a trace bridges the two halves of each circle together (see Figure 4 for a closer view). To change the device number, we need to cut away the bridge on one or both of these Jumper Pads (this is detailed in Figure 5). Which pads do what? The chart below shows you. Note that the pads have numbers by them, either a 1 and 2 or J1 and J2 (there are two exceptions where the pads aren't marked—the 1541 Rev. C short board,

**DISCLAIMER**

Information in this column is presented for educational purposes only. In no event will *Commodore World* or Creative Micro Designs, Inc. be liable for any direct, indirect or consequential damages resulting from the use or misuse of this information.

**Jumper Pad Device Number Programming**

| Device Number | 8 | 9 | 10 | 11 |
|---|---|---|---|---|
| Jumper 1 or J1 | Closed | Open | Closed | Open |
| Jumper 2 or J2 | Closed | Closed | Open | Open |

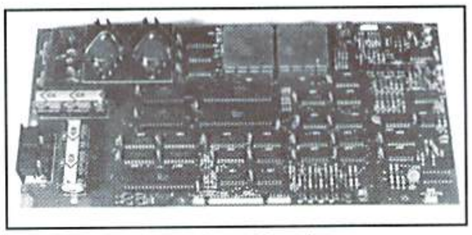


Figure 1. 1540/1541 Long Board

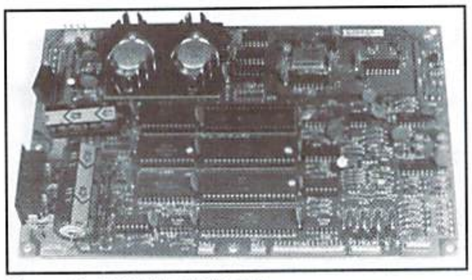


Figure 2. 1541 Short Board

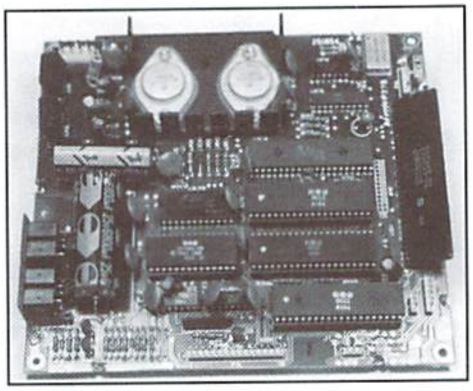


Figure 3. 1541C Board

and the 128D—so we've added numbers to the pictures that show the pad locations).

Note the terminolgy used in the chart: "Open" indicates that the bridge trace has been cut, while "Closed" indicates that it has not (or that the two pad halves are connected by other means, like a switch or solder bridge).

If you're making a permanent change (and not installing switches), pick the device number you want and cut the appropriate traces (see Figure 5) with an Exacto knife. If you're installing two switches, cut both; if you're installing one switch, decide what device numbers you want available by referring to the chart, and cut one or both pads accordingly. (There are a lot of possible device number combinations that can be achieved using a single switch, but most users usually just cut J1, or Jumper 1, and install a switch on it to provide switching between devices 8 and 9.)

Be sure to examine your work to make certain the cuts completely seperate the pad halves. If you're not installing switches, reassemble and test your unit. If you're installing switches, continue on.

Select a location to mount the switch(es), and drill the appropriate mounting hole(s). Now prepare two wires for each switch, long enough to reach from the Jumper Pad locations to the switch mounting hole(s) with a couple of extra inches of extra slack. Strip off an eighth of an inch of insulation from both ends of each wire.

Solder two wires to each switch; one to the center terminal, and the other to the outside terminal. (If you're using SPDT switches, use either of the outside terminals and ignore the other.) Now solder the other ends of the wires to the two halves of the pads, one wire to each pad. Make sure that you don't short the pads together with your solder, and cut away any excess wire. Also be sure that the wires from each switch go to the same number Jumper Pad.

Once you've finished, examine your work carefully, and after you're certain that all is correct, reassemble and test your unit.

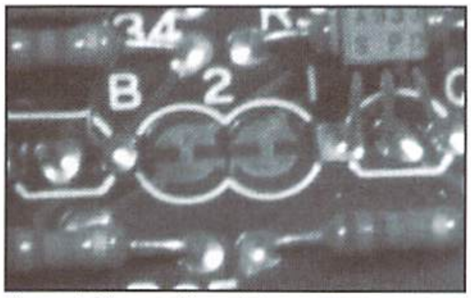


Figure 4. Closeup View of Jumper Pads

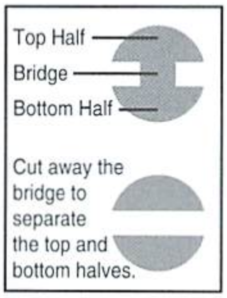


Figure 5.

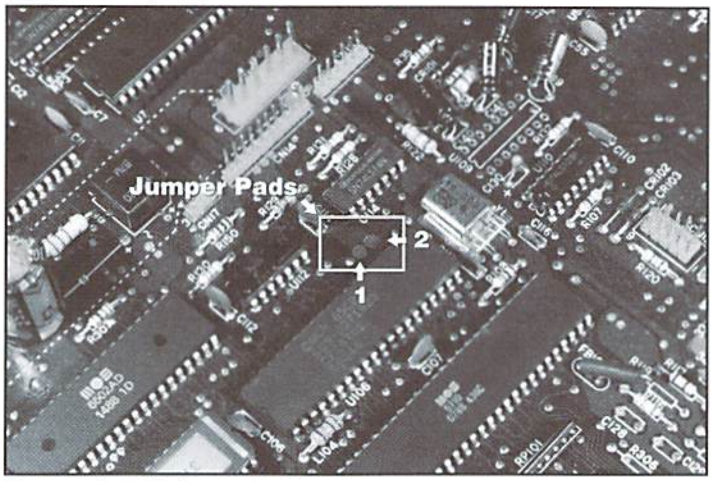


Figure 6. C-128D Jumper Pad Location

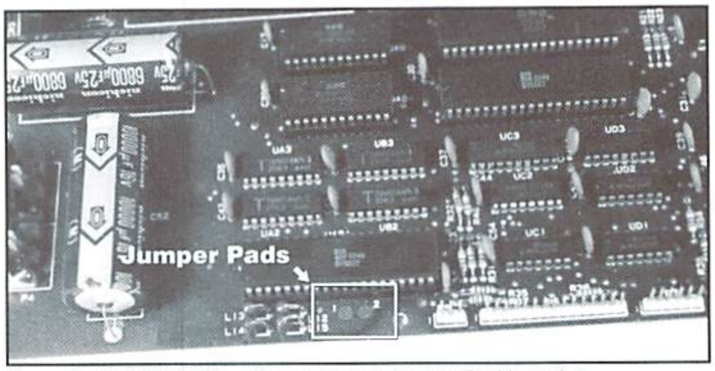


Figure 7. 1540/1541 Rev. A Long Board Jumper Pad Location

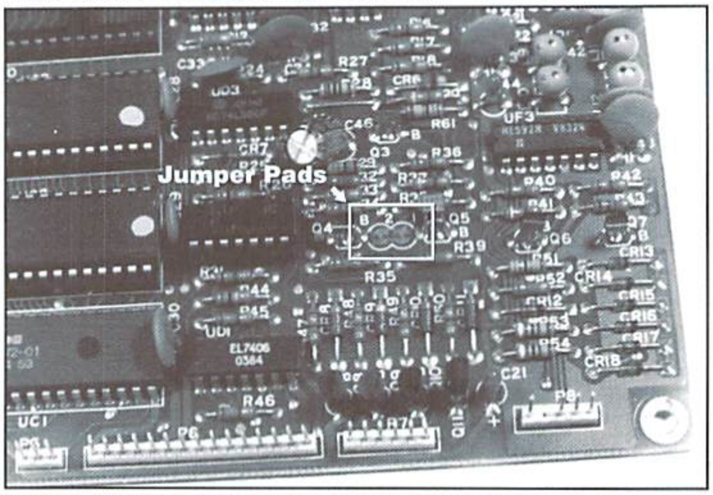


Figure 8. 1541 Rev. A/Rev. B Short Board Jumper Pad Location

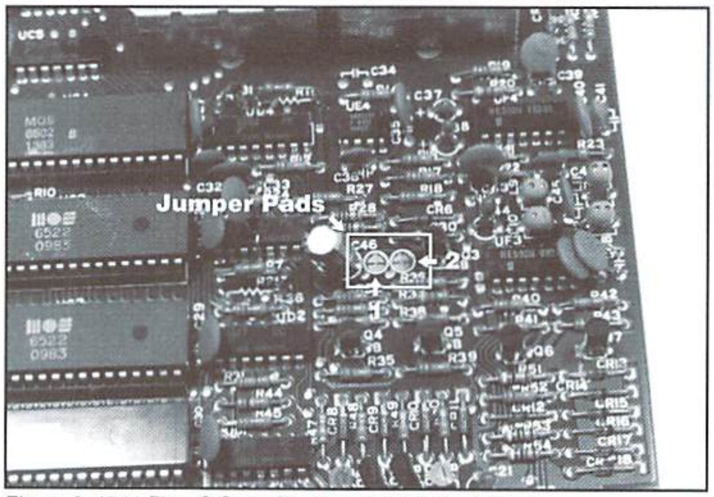


Figure 9. 1541 Rev. C Short Board Jumper Pad Location

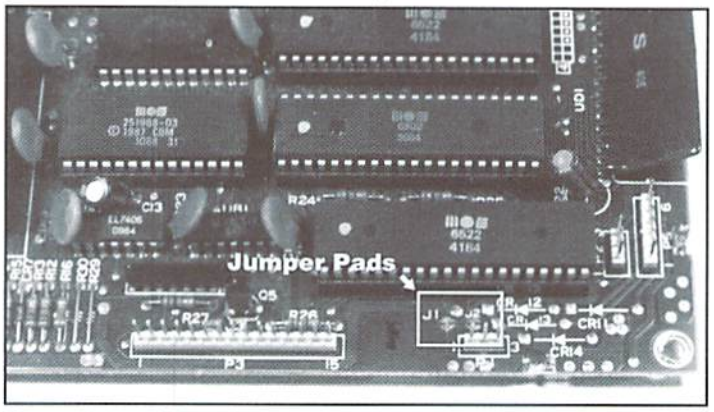


Figure 10. 1541C Rev. A Jumper Pad Location

# GEO PROGRAMIST

*by Maurice Randall*

Let's start getting into some of the GEOS Kernal routines and see how we can use them effectively in our programs. In this issue, I will cover some of the more general routines that you might be likely to use. These will be routines that perform actions on memory or help out with math calculations. In a future issue, we will get into the routines that involve graphics and other screen display routines.

I have always felt that a computer programmer should be comfortable with math. After all, isn't that basically what a computer is doing? It is always calculating addresses and offsets. That is all taking place at the processor level of the machine. At the programming level, we might need to calculate a game score, or multiply a dollar amount by a quantity, or perform some other typical calculation. Sometimes, developing a math routine to perform these jobs can be quite tedious. The GEOS Kernal provides us with some built-in routines that will help us out in many cases. Let's take a look at some of the more helpful ones.

### It's Dividing Time

Perhaps the most difficult of the standard math functions to write in machine language is division. Once the concept is understood, it is not that big of a deal. But comprehending it and understanding it can be tricky, and it is also easily misunderstood or forgotten. Take a simple math problem such as 379 divided by 23. How would we write a routine to perform this calculation? In GEOS, it is simple to use the Kernal routine called Ddiv. Ddiv gives us the ability to divide a 16-bit number by another 16-bit number as the following example shows:

```
LoadW r0,#379        ;divide 379
LoadW r1,#23         ;by 23
ldx #r0              ;x points to r0
ldy #r1              ;y points to r1
jsr Ddiv             ;perform the division
```

Upon return from Ddiv, the zero page register that x was pointing at, which was r0, will be holding the quotient result of the division and r8 will be holding the remainder, if any. Since we were dividing by 23, which is less than 256, our code could assume that r8H is zero and only need to check r8L for the remainder. In this example, r0 will now contain 16 and the remainder, which is 11, will be in r8L.

Ddiv does not limit you to using r0 or r1. You could use any of the available registers such as a0 through a9. Just be sure not to use r8, obviously, or r9 which is also used by the routine. The important thing is that x is pointing at the number that will be divided by the number that y is pointing at.

### Multiplying Is Just As Easy

The nice thing about using the GEOS routines is the similarity in how they are setup. This makes it easy to remember each time you need to use the routines. Let's see how multiplying looks the same as dividing. This time, we will multiply 379 by 23. The routine that is used is called DMult and works exactly the same as Ddiv, except that it gives us the result of a multiplication instead of a division.

```
LoadW r0,#379        ;multiply 379
LoadW r1,#23         ;by 23
ldx #r0              ;x points to r0
ldy #r1              ;y points to r1
jsr Dmult            ;perform the multiplication
```

This time, the 16-bit result will also be found in r0 upon return from DMult. Obviously, there won't be a remainder in this case, however the result of a multiplication could very easily exceed 16 bits. If the result is greater than $ffff (or 65,535), the carry flag will be set. As long as the carry flag is clear, you know the result is entirely contained within r0. In our example, r0 will contain $220d, or 8717 in decimal.

There are also two other multiplication routines available. One is called BMult for multiplying an 8-bit byte times a 16-bit word, and the other is BBMult which will multiply one 8-bit byte by another 8-bit byte. Each routine will return a 16-bit value. These two routines operate almost exactly like DMult except that where a single byte is used, only the low-byte of the register will be needed. The register that is pointed at by x always gets the result, and it is always a 16-bit result. In the case of BMult, x must point to the 16-bit value, while y points to the 8-bit value. Once multiplied together, the result will be in the register pointed at by x.

### Counting Down To Zero

Ddec can be a handy routine when you need a 16-bit counter. Let's say you are reading bytes from a buffer and you want to make sure that you only read a specific number of bytes. Maybe the buffer begins at $5000 and you want to read in 1000 of those bytes and send each one individually to another routine for whatever reason. You could put the value of 1000 into a register and let Ddec decrement the value each time a byte is read and processed. Here's how it might look:

```
startOfBuffer =$5000

   LoadW r0,#startOfBuffer
   LoadW r2,#1000           ;count down from 1000
10$
   ldy #0                   ;read the byte pointed at
   lda (r0),y               ;by r0
   jsr ProcessByte          ;go do whatever with it
   inc r0L                  ;point r0 to the next byte
   bne 20$
   inc r0H
20$
   ldx #r2                  ;point x at r2
   jsr Ddec                 ;decrement r2
   bne 10$                  ;branch back if not zero
```

As you can see in the example, all we have to do is set x to the value of #r2 and let GEOS decrement it for us. By testing the zero flag, we will know if we have processed 1000 bytes yet. Of course, in this example, the routine that does the processing will have to preserve both r0 and r2, or we will have a bug on our hands.

There are a few other math routines, but I find these to be the ones that I use the most.

## Clear Your RAM

Here's a pretty important routine, it's called ClearRam. I use this one mostly when a program first gets loaded and run. I use it to clear out a variable area. It's common for you to establish a variable area that begins just after the end of your program code, in an area that you might define as ramsect with GeoAssembler. Well, when your program is first loaded, there is no guarantee as to what is in this area. If you just start letting your code run, you might get to a routine that checks one of your variables and if it contains the wrong byte, it could mean trouble. So, try to make a point of clearing out this area before getting too far into the running of your program. It is also simple to use, as you will see.

```
  LoadW r0,#50                ;zero out 50 bytes
  LoadW r1,#startOfVariables  ;beginning here
  jsr ClearRam
```

In the above example, startOfVariables is the beginning of the area that we will clear out. In this area, we have 50 bytes that need clearing. You can use ClearRam to clear out any size area anywhere within the computer's memory.

As you can see, GEOS has routines to help us programmers out. Make use of them and your own code will be easier to write and will also be more compact. Let's take a look at some more of these easy to use routines next issue.

**Mad Man Software**

**CKit 94!**

**It's Cool**

CKit 94 is a powerful collection of backup utilities designed to take advantage of the expanded Commodore systems of the 90's. In fact the CKit is so advanced, we recommend it only for certain Commodore systems.

**If You Have One Of These**
**Commodore 64, 64C, 128, or 128D**
**And One Of These**
**CMD RAMLink, PPI RAMDrive, or a 17XX REU with JiffyDOS**
**And At Least One Of These**
**Commodore 1541/1571/1581 or any CMD Storage Device**
**◆ YOU NEED THE CKIT 94! ◆**

**It's Mean**

CKit 94 can view, select, and copy over 1,000 files! It can also copy entire 41/71/81 disks in ONE pass. It can even make the next copy directly from RAM. With the CKit you can squeeze over 700 files into an archive. Even more, you can squeeze your 41/71/81 disks and CMD partitions into backup files. All of this power is at your command through an easy to use menu driven interface.

**File Utilities**
**Filemaster file copier and the Archiver file squeezer**
**Disk Utilities**
**Procopy disk copier and the 41/71/81/Native Boa disk squeezers**
**BBS Utilities**
**New Dissolver SDA maker and fast PD Decompers**

**It's Better**

CKit 94 has 2 X faster and tighter compression than in previous versions. All of the utilities have been enhanced for power and speed. Plus, it has three new powerful utilities. And it takes full advantage of Commodore 128's and REU's. Wow!

**Check Out The Support**
- ✓ Commodore 1541/1571/1581 Drives
- ✓ CMD Hard Drives/Floppy Drives/RAMLinks
- ✓ PPI RAMDrives
- ✓ Commodore 128 VDC RAM Support (16K and 64K)
- ✓ Commodore 128 2MHz Enhanced Mode Support
- ✓ Commodore 17XX REU Support (up to 16Meg)
- ✓ RAMLink and RAMDrive DACC Support (up to 16Meg)

**Get Your Copy Now!**

CKit 94 is available at fine Commodore mail order outlets such as Creative Micro Designs. You may also order directly from us.

Enclose Check or Money Order for:
CKit 94 $39.95 (Update $9.95) plus
Shipping $5 US and Canada ($10 Foreign)
Update uses Key from earlier version

Mad Man Software, Inc.
1400 East College Drive
Cheyenne, WY 82007
(307) 632-1178 Information

# BASIC INSTINCTS

## USING ML SUBROUTINES

*by Gene Barker*

If you've been following along the last couple of issues, you've probably noticed that BASIC programs lack the speed of their machine language counterparts. Lets take a look at how you can go about bridging this speed gap in your own programs.

### Speed Techniques

There are a number of ways to speed up your BASIC programs. Some programmers prefer to write short concise code, excluding all possible extras. For example an experienced programmer may code:

```
10 fori=0to15:poke53280,i:next
```

Instead of:

```
10 rem * flash the border all 16 colors *
20 for i=0 to 15
30 :   poke 53280,i
40 next i
```

Notice how the programmer left out the REM statement and the variable (i) after the NEXT statement. The programmer also chose to use a compound statement through the use of the colon. Despite the speed advantage of this one line statement, I prefer to use the latter. Readability can be quite valuable during development, debugging, and maintenance. However, don't discount this method just because I don't like it. It's sometimes necessary to sacrifice readability to make your program the best it can be.

Some programmers compile their programs with a BASIC compiler. This is a wonderful solution. However, it should be considered the last step in speeding up your BASIC programs. This method can also present problems should you decide to integrate your programs with machine language in the future.

### Machine Language Subroutines

Often times, the best way to speed up your BASIC programs is through the strategic use of machine language subroutines. Fortunately, there are a number of great pre-written machine language subroutines available. These can be found in BASIC programming books (*Compute's Programming the Commodore 64* and *Mapping the Commodore 64* are two fine examples), LOADSTAR's various ML tool boxes (excellent tools available via Softdisk Publishing), and in the public domain. So it's not necessary to know machine language in order to harness its advantages. Which brings us to...

### This Issue's Example

This month's example focuses on the strategic use of a pre-written machine language subroutine. As a bonus you will have a handy subroutine that you will use again and again. The subroutine is a directory listing routine. It is written in BASIC and has a small machine language subroutine of its own. I included a 100% BASIC version of the directory lister to demonstrate the performance advantage of the machine language. The two subroutines are:

Lines 1000 - 1199 : BASIC Directory Listing Subroutine
Lines 2000 - 2299 : BASIC+ML Directory Listing Subroutine

These two subroutines are complex; please don't spend too much time deciphering them. When you use pre-written subroutines in BASIC or machine language this comes as no surprise. The key concept here is not how the subroutines work, but how you use them. You will find that using pre-written subroutines can save hours of work.

### Project

See if you can add the directory listing subroutine (the ML version of course!) to one of your programs. Should you change the line numbers, be sure to change them in the THEN and GOTO statements as well. If you have any previous DATA and READ statements, makesure they won't interfere with the subroutine's initialization; you may have to move lines 2025 to 2099 near your other READ loops. Expert programmers: see if you can edit the subroutine so that you can view different partitions and sub-directories on CMD devices. Hint: Build a path string in the OPEN statement.

### Notes

As you type in this issue's program, take it one section at a time. Try to get a general idea of what the section is trying to accomplish. See if you can follow how the section is using its variables. If you see an unfamiliar BASIC statement, take a quick look at it in your BASIC manual. If you are still confused, move on to the next section; often times the next section helps explain the previous one. Above all, *back your work up frequently!*

### Entering The Program

Before entering this issue's program, load and run the CHK-LIST utility (located elsewhere in this issue). CHK-LIST insures that you enter the program correctly the first time. Also, remember to SAVE the program before you attempt to RUN it. It never hurts to be safe.

**MLDIR.BAS**

```
5000  100 rem--------------------------------
6dc2  105 rem commodore world magazine
c67c  110 rem basic instincts w/gene barker
c38b  115 rem using ml subroutines
0e6b  120 rem
2d24  125 rem (c)1995 creative micro designs
92ad  130 rem--------------------------------
1315  135 rem-
63fa  140 rem display program name
c243  145 rem-
6be1  150 print"{CLEAR/HOME}{CRSR DN}sample di
          rectory driver"
```

**MLDIR.BAS** *(cont.)*

```
cØe8  16Ø rem get the desired drive #
1315  165 rem-
78Øe  17Ø print"{CRSR DN}enter drive # (8-31):
      "
f9bb  175 input xd
88d7  18Ø if xd<8 or xd>29 then end
c243  185 rem-
92fc  19Ø rem get desired directory method
dØ7b  195 rem (pure basic or ml enhanced)
c1cc  2ØØ rem-
67b7  2Ø5 print"{CRSR DN}use which directory r
      outine:"
e6bØ  21Ø print"{CRSR DN}(1) pure basic"
6ab5  215 print"(2) basic w/ml enhancement"
d992  22Ø get x$:if x$="" then 22Ø
554c  225 if x$<>"1" then 255
Øb3d  23Ø rem-
ee65  235 rem use pure basic option
da6b  24Ø rem-
657b  245 : gosub 1ØØØ
f41c  25Ø : goto 3ØØ
eØ46  255 if x$<>"2" then 22Ø
5291  26Ø rem-
a1d2  265 rem use basic w/ml
5291  27Ø rem-
27c3  275 : gosub 2ØØØ
5291  3ØØ rem-
213e  3Ø5 rem ask if user wants to try again
4936  31Ø rem-
ef2a  315 print"{CRSR DN}do you wish to try ag
      ain (y/n)?"
f939  32Ø print"{CRSR DN}note: be sure to try
      the ml version"
d1b3  325 print"at least twice."
9673  33Ø get x$:if x$="y" then 185
966b  335 if x$<>"n" then 33Ø
b13f  34Ø end
d85a  1ØØØ rem-----------------------------
f58Ø  1ØØ5 rem display directory (basic)
3d5a  1Ø1Ø rem
cØ54  1Ø15 rem given: xd - device number
dbf5  1Ø2Ø rem-----------------------------
8fØ5  1Ø25 print chr$(147);
d87Ø  1Ø3Ø open 5,xd,Ø,"$"
1de5  1Ø35 get#5,xx$,xx$
4e32  1Ø4Ø get#5,xx$,xx$
6647  1Ø45 print
6c4e  1Ø5Ø get#5,x1$,x2$
Ø3c6  1Ø55 if st=Ø then 1Ø85
81a9  1Ø6Ø : close 5
21d3  1Ø65 : print
7ØbØ  1Ø7Ø : print"[return]"
6dØ2  1Ø75 : get xx$:if xx$<>chr$(13) then 1Ø7
      5
cd4c  1Ø8Ø : return
c143  1Ø85 if x1$="" then x1$=chr$(Ø)
f6fØ  1Ø9Ø if x2$="" then x2$=chr$(Ø)
79c9  1Ø95 print mid$(str$(asc(x1$)+(asc(x2$)*
      256))+"{5 SPACES}",2,6);
Ø9ce  11ØØ get#5,xx$:if xx$=chr$(32) then 11ØØ
f774  11Ø5 print xx$;
a9d3  111Ø get#5,xx$:if xx$="" then 1Ø4Ø
f774  1115 print xx$;
4985  112Ø goto 111Ø
9c11  2ØØØ rem-----------------------------
229c  2ØØ5 rem display directory (basic/ml)
f19e  2Ø1Ø rem
52Ø4  2Ø15 rem given: xd - device number
9fbe  2Ø2Ø rem-----------------------------
5c88  2Ø25 rem-
b2f8  2Ø3Ø rem make sure ml is setup
```

**LOAD2.BAS** *(cont.)*

```
472f  2Ø35 rem-
Ø596  2Ø37 xa=52864
1a4c  2Ø4Ø if xv=999 then 21ØØ
86ba  2Ø45 : print:print"initializing ml..."
8f58  2Ø5Ø : xi=Ø:xc=Ø
fab1  2Ø55 : read xv
3591  2Ø6Ø : if xv=999 then 2Ø8Ø
b7dd  2Ø65 : poke xa+xi,xv
3585  2Ø7Ø : xi=xi+1:xc=xc+xv
6511  2Ø75 : goto 2Ø55
4b79  2Ø8Ø if xi=28Ø and xc=36966 then 21ØØ
fdØe  2Ø85 : print"!!!error in data statements
      !!!"
4363  2Ø9Ø : end
7eØ6  21ØØ rem-
ba38  21Ø5 rem setup ml call
7eØ6  211Ø rem-
66e5  2115 print chr$(147);
Øbc7  212Ø open 5,xd,Ø,"$"
ae1d  2125 sys xa
57a6  213Ø close 5
662e  2135 rem-
e51b  214Ø rem wait for a return
b778  2145 rem-
c37d  215Ø print
124f  2155 print"{CRSR DN}[return]"
dbØe  216Ø get xx$:if xx$<>chr$(13) then 216Ø
f5ba  2165 return
b4f7  22ØØ rem-
5419  22Ø5 rem ml code
af5Ø  221Ø rem-
9ea9  2215 data 162,5,32,198,255,162,Ø,134
1a14  2216 data 144,32,193,2Ø6,32,193,2Ø6
8227  2217 data 32,193,2Ø6,32,193,2Ø6,169
ecc4  2218 data 13,32,21Ø,255,32,193,2Ø6
595a  2219 data 141,151,2Ø7,32,193,2Ø6,174
f381  222Ø data 151,2Ø7,168,169,6,32,24Ø
f233  2221 data 2Ø6,32,193,2Ø6,2Ø1,32,24Ø
56f3  2222 data 249,32,21Ø,255,32,193,2Ø6
2c3d  2223 data 24Ø,212,32,21Ø,255,76,182
3de7  2224 data 2Ø6,32,2Ø7,255,166,144,24Ø
24f4  2225 data 24,224,64,2Ø8,6,1Ø4,1Ø4
cc5e  2226 data 32,2Ø4,255,96,1Ø4,1Ø4,32
7821  2227 data 2Ø4,255,169,13,32,21Ø,255
9ec4  2228 data 32,21Ø,255,96,17Ø,96,142
5cae  2229 data 147,2Ø7,162,Ø,142,148,2Ø7
8e31  223Ø data 142,149,2Ø7,76,254,2Ø6,142
a8fØ  2231 data 147,2Ø7,14Ø,148,2Ø7,162
41db  2232 data Ø,142,149,2Ø7,76,254,2Ø6
955Ø  2233 data 141,15Ø,2Ø7,173,147,2Ø7
cdb7  2234 data 174,148,2Ø7,172,149,2Ø7
Ø6b7  2235 data 133,4Ø,134,41,132,42,169
7cb9  2236 data Ø,141,146,2Ø7,162,21,16Ø
baØØ  2237 data 255,2ØØ,165,4Ø,72,56,253
21a9  2238 data 122,2Ø7,133,4Ø,165,41,72
7968  2239 data 253,123,2Ø7,133,41,165,42
8fff  224Ø data 72,253,124,2Ø7,133,42,144
182d  2241 data 6,1Ø4,1Ø4,1Ø4,76,25,2Ø7
4d2b  2242 data 1Ø4,133,42,1Ø4,133,41,1Ø4
b14d  2243 data 133,4Ø,152,172,146,2Ø7,2Ø8
feØØ  2244 data 7,2Ø1,Ø,24Ø,11,141,146,2Ø7
983a  2245 data 9,48,32,21Ø,255,2Ø6,15Ø
Øf78  2246 data 2Ø7,2Ø2,2Ø2,2Ø2,16,185,173
7356  2247 data 146,2Ø7,2Ø8,8,169,48,32
aØ82  2248 data 21Ø,255,2Ø6,15Ø,2Ø7,173
98af  2249 data 15Ø,2Ø7,24Ø,9,17Ø,169,32
beab  225Ø data 32,21Ø,255,2Ø2,2Ø8,25Ø,96
d972  2251 data 1,Ø,Ø,1Ø,Ø,Ø,1ØØ,Ø,Ø,232
dc31  2252 data 3,Ø,16,39,Ø,16Ø,134,1,64
6e6a  2253 data 66,15,128,15Ø,152,Ø,Ø,Ø
62d7  2254 data Ø,Ø,Ø,999
```

# Peripheral Vision

*By Jim Butterfield*

## MUCH ADO ABOUT NULL

### The BASIC Null

There are times when you read information from a device but get nothing. The two types of "nothing" are the NULL character (character zero) or no character at all. These are not as sharply defined as you might think.

When a BASIC program reads a binary file or a program file from disk, it's likely to input Null characters, what BASIC would call CHR$(0). It's guaranteed if you're reading another BASIC program. But BASIC does a nasty thing when it sees such a character: it throws it away and gives you a "null string"—no character at all. There's an easy work-around for this, but first let's see why BASIC behaves this way.

Suppose you're reading the keyboard using the BASIC GET command. This command returns right away, even if there's no key input waiting in the keyboard buffer. The BASIC interpreter asks the operating system for data from the keyboard buffer; the operating system either delivers a character or a binary zero, which means "no character" (you can't usefully generate a binary zero from the keyboard). When the BASIC interpreter sees the binary zero, it correctly identifies this as no-key, and converts what would be a single character string, CHR$(0), into no-character, a null string. But the same mechanism is built into the GET# command that may be used to read disk files, with the result that binary zeros received from such a file are thrown away.

### Fixing Those BASIC Nulls

You always get a character when you're reading a file from disk. Even if you've foolishly gone beyond the end of the file, you'll still get back a RETURN character, CHR$(13). Since the computer will never encounter a no-character condition during a disk file read, there are several ways to fix the BASIC "null-character to null-string" anomaly. Assume we're about to read a character from logical file 1. We might code:

```
GET#1,C$
IF C$="" THEN C$=CHR$(Ø)
```

We'll work this coding into a program, but let's look at another alternative. Often, we want to analyze a binary file in terms of the numeric value of the bytes. To extract the number, we would use the function ASC(C$). But—except on the 128—that function won't work with a null string: the program will stop with an error. We could fix this problem with the same IF C$="" trick given above. Or try this:

```
GET#1,C$
V=ASC(C$+CHR$(Ø))
```

If C$ is not null, the ASC function extracts its value; it looks only at the first character of a string. But if C$ is a null string, then the expression inside the parentheses becomes simply CHR$(0), and the function returns the correct value of zero. It's faster to define CHR$(0) as a variable near the beginning of your program. You'll get better speed with a variable in the working loop.

### Demo Program 1

Program 1 is a brief program which will copy a file and not be bothered by any binary zeros. If you save it as MYPROG, it will copy itself when run.

Without line 210, those binary zeros would not copy and you'd have a mess instead of a duplicate program. Now let's write a program to analyze another BASIC program, digging into the numeric values of the bytes (see Program 2).

The program will report each line within the BASIC program, and where the line will be placed in memory (assuming the load address is honored).

### Machine Language

The above anomaly reverses if you're coding in machine language. The call to GET (at $FFE4) returns a value in the A register. The program knows if it's seeking information from the keyboard, in which case a zero value in A means "no key" and may be handled accordingly. If the program is reading a disk file, a zero byte in A is valid data, and will be handled in the usual way. Incidentally, a call to INPUT (at $FFCF) works exactly the same as GET for file reading, but not for keyboard/screen input.

### The RS-232 Conundrum

There is one peripheral: the RS-232 communications port, that wants it both ways. There might not be any characters waiting; or a legitimate CHR$(0) might be delivered as input. So we must look at how the computer deals with

**Program 1**

```
100 Z$=CHR$(Ø)
110 OPEN 15,8,15
120 OPEN 1,8,2,"Ø:MYPROG,P,R"
130 INPUT#15,E,E$
140 IF E<>Ø THEN PRINT E$ : STOP
150 OPEN 2,8,3,"Ø:MYCOPY,P,W"
160 INPUT#15,E,E$
170 IF E<>Ø THEN PRINT E$ : CLOS
E 1 : STOP
180 REMARK: MAIN LOOP HERE
190 GET#1,A$
200 S=ST   :REM SAVE STATUS FOR
EOF TEST
210 IF A$="" THEN A$=Z$
220 PRINT#2,A$;
230 REMARK: DO NOT FORGET THE SE
MICOLON ABOVE
240 IF S=Ø GOTO 190
250 CLOSE 2
260 CLOSE 1
270 PRINT "FILE COPY FINISHED."
280 CLOSE 15
```

**Program 2**

```
100 Z$=CHR$(Ø)
110 INPUT "BASIC PROGRAM NAME";P$
120 OPEN 15,8,15
130 OPEN 1,8,2,"Ø:"+P$+",P,R"
140 INPUT#15,E,E$
150 IF E<>Ø THEN PRINT E$ : STOP
160 GET#1,A$,B$
170 X=ASC(A$+Z$)
180 Y=ASC(B$+Z$)
190 IF X<>1 THEN PRINT "MAYBE NOT BASIC?"
200 L=X+256*Y
210 PRINT "LOAD ADDRESS = ";L
220 REM: MAIN LOOP, LINK AND LINE NUMBER
230 GET#1,A$,B$
240 KØ=ASC(A$+Z$) + 256*ASC(B$+Z$)
250 IF KØ<=L GOTO 360
260 GET#1,A$,B$
270 LØ=ASC(A$+Z$) + 256*ASC(B$+Z$)
280 PRINT "LINE";L0;"AT ADDRESS";L
290 L=L+4
300 IF LØ < L1 GOTO 360
310 L1 = L0
320 GET#1,A$
330 L=L+1
340 IF ASC(A$+Z$)<>Ø AND ST=Ø GOTO 320
350 IF ST=Ø GOTO 230
360 IF KØ<>Ø THEN PRINT "FILE PROBLEM!"
370 CLOSE 1
380 CLOSE 15
390 PRINT "FINISHED."
```

this port. The following deals with the "standard" RS-232 interface of the VIC-20, Commodore 64, Plus-4, and the C128. Special interfaces, such as SwiftLink, are not part of the description.

In all of these machines, characters that arrive at a "live" RS-232 port are placed into a rotating buffer, usually 256 bytes in size. The interrupt system stores the next character by means of a "put-'em-in" pointer. When asked for a character from the RS-232 port, the system uses a "take-'em-out" pointer to select the next character from the buffer. If the two pointers match, the system knows that there are no characters waiting, and returns a binary zero... which you might confuse with a received NULL character.

Figure 1 shows the idea. One pointer puts received characters into the buffer, and the other is used to get the characters when they are needed. The buffer for the Plus-4 is 64 bytes in size, versus 256 bytes on all other machines. That seems unfair; the Plus-4 with its ACIA chip has the highest communications capability of any of the 8-bit Commodore computers.

The two pointers: put-'em-in and take-'em-out, are each one byte in size. Their logic is carefully arranged: if the buffer fills up, the pointers won't "cross" so that you'll lose everything. Of course, you'll lose some characters since there's no place for them. The solution is to check to see if the two pointers are equal. If they are, you have no characters waiting, and your program can go on to other work. If the two pointers are not equal, you may command GET# and be sure—even if it's a NULL—that you have received a valid character.

The only useful test you can perform on these pointers is for an equals condition. The characters are placed into a "rotating" buffer, so either pointer might have a value higher or lower than the other. Also, the put-'em-in pointer is interrupt driven: it might change as you are testing it!

**Pointer Location**

The address of the two pointers is not the same in all machines. The table below is a summary of their addresses. The addresses are given in decimal, suitable for BASIC use. Remember in BASIC, you still have to convert any received null strings to CHR$(0) as discussed above. The code would go something like this for a Commodore 64 or VIC-20:

| | put-'em-in | take-'em-out |
|---|---|---|
| VIC - 20 and Commodore 64 | 667 | 668 |
| Commodore 128 (bank 0) | 2584 | 2585 |
| Plus-4 | 2001 | 2001 |

```
100 OPEN 1,2,3,CHR$(6)  :REM
START THE COMMS LINK
```

... then, within a loop:

```
500 IF PEEK(667)=PEEK(668) GOTO
600 :REM IF NOTHING
510 GET#1,A$
520 IF A$="" THEN A$=CHR$(Ø)
```

... eventually, when the program finishes:

```
900 CLOSE 1
```

The approach in machine language is almost identical. The substitute for lines 500 to 520 above might read as seen in Program 3, below.

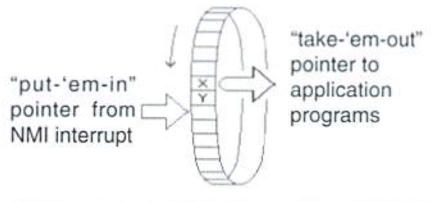
**Short Notes**

I should mention that there is no problem with outputting NULL characters, as you may have guessed from programs above. Send a CHR$(0) and it will get to its destination unless you're sending it to the screen, where it will do nothing.

Some users are confused about the difference between NULL, binary zero, and the '0' character on their keyboard. Just keep in mind that the '0' character is really character number 48 (hexadecimal 30), so it's distinct from the NULL.

**Figure 1**

The RS-232 receive sequences store incoming characters into a rotating buffer until the application program takes them out. If the two pointers are equal, there are no characters waiting.


"take-'em-out" pointer to application programs
"put-'em-in" pointer from NMI interrupt
X
Y

**Program 3**

```
PUTEMIN =    $29B
TAKEMOUT         =        $29C
...
         LDA  PUTEMIN
         CMP  TAKEMOUT
         BEQ  ...               ; (branch if no character)
         LDX  #$Ø1              ; logical device 1..
         JSR  $FFC6             ; ..CHKIN, connect input stream
         JSR  $FFE4             ; ..GETIN, get the character
         PHA                    ;    save the character on stack
         JSR  $FFCC             ; ..CLRCHN, restore input stream
         PLA                    ; restore input character
```

# ASSEMBLY LINE

## FIRST REGISTERS, FIRST DEBUGGING

*by Jim Butterfield*

Our first steps in machine language will emphasize debugging. It's good to confirm that your program works correctly, step by step, and to explore how instructions work.

Our universal tool for this work will be a Machine Language Monitor (MLM). On the Commodore 128 or Plus-4, you have one built in; on other machines, you'll need to load one. I suggest you try the public domain MLM, "SuperMon". Its operation is very close to that of the built-in monitors, so we can all use the same commands and see comparable display data.

"SuperMon" is largely for the Commodore 64. But other versions are around on networks and bulletin boards. There's a tiny one for the VIC-20; tiny because of the small memory on that machine, and also trimmed to fit into the limited screen width. The old PET/8032 computers have a built-in machine language monitor, but these are not fully featured. For example, there's no assembler/disassembler built in. So there's a version of SuperMon for those machines, too.

With an MLM, you can enter program code, you can check code, you can save it, and you can examine how it runs. Most MLM packages contain a "tiny" assembler, which we'll use here. If you happen to have a full-featured "symbolic" assembler, you can use it to prepare the program if you wish, but for debugging you'll need to come back to the MLM. By the way, other platforms call the MLM a "debugger"—it's really the same thing.

### About Registers

The 6502-class chip that powers all Commodore 8-bit computers contains storage areas—called "registers"—within the chip. Three are used for data: they are named A, X, and Y, and we'll talk about them in a moment. One register (SP, "stack pointer") is used for the stack; we won't worry about it right now. Another is called PC, for "program counter"; it shows us the address in memory where the processor will look for its next instruction. On other platforms, the PC may be called IP ("instruction pointer").

Finally, there's a register called the SR ("status register"). It holds the results of recent tests, and also certain processor control information. We may need to look at this one during debugging sessions.

Figure 1 shows the registers that are inside your processor chip. All except the PC are eight bits long; that means they can hold a value from 0 to 255 (hexadecimal FF), or, if you like, an ASCII character.

Register A is sometimes called the "accumulator"; registers X and Y are often called "index registers". Any of the three can be used to hold data; you can load it in, test its value, and store out a copy. We'll use a couple of them in the short program given here.

### Planning the Program

Our first program will reverse the contents of two locations in memory. It's a start, and you'll have a chance to do programming and debugging.

We need to pick a location for this short program. Because of the wide variety of machines that might be used, I'll pick the area around address 8192, hex 2000, which is available on most machines. (On an unexpanded VIC-20, you'll need to pick a lower address, say 7424, or $1D00).The two data locations can be any convenient place in RAM. If you have a "memory-mapped" screen, you might pick a couple of addresses there, so that you can actually see the characters being swapped,. but that won't work on the 80-column C128, of course. For the moment, I'll choose $2100 and $2101.

### Getting Started

If you don't have a built-in Machine Language Monitor, load one in. With SuperMon, for example, you load the program, command RUN, and you're in business. With the Plus-4 or Commodore 128, just command MONITOR.

You'll see a display of the processor's registers. Ignore them for the time being. Type the following command:

```
A 2000 LDA $2100
```

When you press [RETURN], you might be surprised to find that the above line has been changed. First, let's look at what you have typed:

A - stands for Assemble. The instruction you type will be translated into machine code.

2000 - is the address at which you wish to assemble this instruction.

It's 2000 hex, or decimal 8192. If you don't tell it otherwise, the MLM will always assume hexadecimal. Most versions of SuperMon and other monitors will allow you to enter decimal by putting a '+' sign ahead of the value, so that we could code: A +8192 LDA +8448 and produce the same result.

LDA - a command to Load the A register. The data loaded will be a copy; the original value will still be in memory.

$2100 - the address from which to load. You could omit the '$' symbol, since the MLM assumes hexadecimal.

When you press RETURN , the line you have typed changes. To the left of the LDA command, you'll see three bytes of information—the translated command as stored in memory. And if you happened to put in any values in decimal, using the '+' sign, you'll find they have been changed to hex.

Additionally, you'll find that the MLM has typed part of the next line for you: 'A 2003'. This helps guard against you miscalculating the address where the next instruction is to go. Complete this next line with the command NOP. That's an instruction that does NOTHING. Seems like a waste of processor time and memory, but it will help with our testing later. Our plan is to load A and X with the two values to be swapped, and then store the register contents back, the other way around. Here's what the lines you type in for the rest of the program should look like:

```
A 2ØØ4 LDX $21Ø1
A 2ØØ7 NOP
A 2ØØ8 STA $21Ø1
A 2ØØB NOP
A 2ØØC STX $21ØØ
A 2ØØF NOP
A 2Ø1Ø RTS
```

You can guess LDX to be "load X", and STA, STX to be "Store A, Store X" respectively. RTS is "Return from Subroutine"; it will take the program back to whoever called it; that will normally be Basic.

Even though you're finished, the MLM will continue to prompt you with 'A 2011' , just press RETURN to signal that you're finished. The program has been stored in memory as you type. You can double check it by commanding: D 2000 2010 ('D' stands for Disassemble).

Keep in mind that we have not executed any of the instructions; we've just put them in memory, ready to run. Type R to see the registers again and you'll see that nothing has changed. Now return to Basic with command: X (for 'exit').

## Running

Set the two locations in memory with POKE commands:

```
POKE 8448,Ø
POKE 8449,2ØØ
```

If you wish, you may check these values using PEEK(). Or you can return to the MLM, and check the memory locations with M 2010 2011; you'll see the first two bytes as 00 C8.

From Basic, command SYS 8192. You'll get READY right away, the program has run in an instant. PEEK addresses 8448 and 8449 , you'll see their contents have been reversed! SYS 8192 again, PEEK again, and the values are back where they started.

## Testing and Debugging

Let's go back to the MLM and disassemble the program with D 2000 2010. Move the cursor up carefully, and change each NOP to BRK; just type over. Press RETURN on each line you make the change. BRK stands for "Break"; we're going to stop the program at each point. When you're finished, back to Basic, and command SYS 8192 again.

This time the program stops at the first BRK instruction. That's at $2003, and the PC should be pointing at the following instruction, at $2004. As you'll see from the register display, it's pointing at $2005 instead. That's an anomaly in the BRK instruction, nothing to worry about.

Again, it's convenient to type D 2000 2010 again if you want to see your program. You can see that we have executed the LDA instruction, and the A register (AC) contains the value copied from address $2100. It's probably zero, if you've been following the instructions closely, and that allows us to check something else: the Status Register (SR).

## Status Register

Figure 1 shows that the Status Register is made up of eight bits, each of which has a distinct meaning. We'll be concerned only with the high bit, marked N ("negative"), and the two lowest ones, marked Z ("zero") and C ("carry"). You'll likely see a value of $32 in the register, so we must change this to binary to see the bits. That's easy with hexadecimal; 3 is 0011, and 2 is 0010, so we see that N is 0, Z is 1, and C is 0.

After every register load or change, the Z flag is adjusted to reflect if the value is zero (yes, so flag Z is 1.), and the N flag is adjusted to reflect if the value's high bit is on (no, so flag N is 0.). A register load command won't change C, so we can ignore that flag ; it won't change during this program.

We expect the next value to be loaded (to X) will be $C8, binary 11001000. That's non-zero, and its high bit is on, so we expect to see flag Z off and flag N on. Continue the program with: G 2004. You're likely to get a SR value of B0—binary 10110000—which confirms flag N as set and flag Z as clear. The value in XR will have changed, since data has been loaded in there; the value in AC will be the same.

"Store" commands never affect any flags in the status register. So try continuing with commands: G 2008 and G 200C and you'll see that the value in SR stays the same. Finally, G 2010 takes us back to Basic.

## Summary

Writing a program is only part of the job. Testing it, to make sure it does the right thing, is the other part. If you know your way around the MLM, you can not only test your programs effectively. You can also see in detail how commands work. The logic need not be fuzzy. With good use of the MLM, the computer will always tell you what it is doing.

Incidentally, if hexadecimal and binary still fog you, take some time out and try to learn it. You'll be that much more effective in programming and testing if you become a "hex nut".

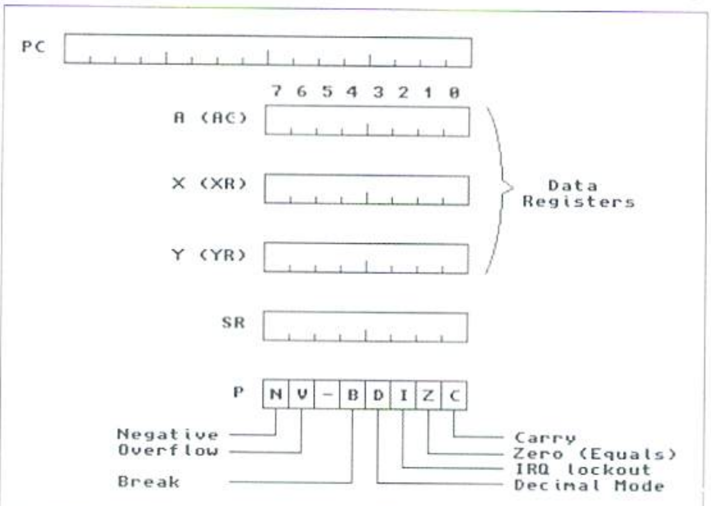


Figure 1. Registers inside the processor chip.

# How to Type In Program Listings Appearing in Commodore World

While *Commodore World* currently doesn't make it a habit of publishing type-in programs, a number of our columns do require entering sample routines. For this purpose, we have created our CHK-LIST utility for the Commodore 64 and 128. This utility uses a 16-bit CRC checksum method to verify that you have correctly entered each program line, and that each of the characters in the program lines are in the correct order.

You'll notice that program listings appear with a column of values to the left of the program lines. These values are the CHK-LIST values and are not to be entered as part of the program. A similar set of values are generated by the CHK-LIST utility to allow you to verify that everything has been entered correctly.

Enter the CHK-LIST program from BASIC. You can use either a C-64 or a C-128 computer. If you use a C-128, it can be in either 64 or 128 mode. Be sure to enter each line carefully to avoid mistakes—until you actually have CHK-LIST working, finding errors in program entry won't be easy. After you have finished entering the program, be sure to SAVE a copy to disk before you attempt to RUN it, just in case. If you aren't familiar with how to save a program to disk, you can use the following command:

```
SAVE"CHK-LIST",8
```

To use CHK-LIST, load it into your computer and type RUN. Make sure that any program you are currently working on is saved first, or start CHK-LIST before you begin typing in a new program. After you have CHK-LIST in memory and running, type NEW. You may now either load or begin typing the program you wish to have CHK-LIST check on. Whenever you want to check your program, type in the appropriate SYS command given below:

| | |
|---|---|
| C-64 or C-128 in 64 mode: | SYS49152 |
| C-128 in 128 mode: | SYS4864 |

Note that when typing in listings, some special characters will appear in braces. For example, {CLR/HOME} means that you should enter the Clear key, which is done by holding down the SHIFT key while you press the HOME key. Other times you may see a number ahead of the key name, such as {3 SPACES} or {5 CRSR L}. This means you should press the key indicated the number of times shown. Most special keys are easy to identify, since the text shown will generally match the text on the key. Exceptions are the space bar {SPACE}, and cursor keys which include directions ({CRSR UP}, {CRSR DN}, {CRSR L} and {CRSR RT}). Be sure to use the correct key combinations for color keys, such as <CTRL><2> for {WHT}.

**CHK-LIST**

```
A454  1Ø F=ABS(PEEK(65533)=255):M=49152:IFFTHE
      NM=4864
6E2F  12 C=Ø:PRINT"{CLR/HOME}WORKING";
E35Ø  2Ø READD:IFD=-256THEN4Ø
AD2Ø  3Ø C=C+D:IFD<ØANDF=ØTHEN2Ø
3316  31 IFD<ØTHEND=Ø-D:M=M-1
Ø7FØ  32 POKEM,D:M=M+1:PRINT".";:GOTO2Ø
578A  4Ø PRINT:READCK:IFC<>CKTHENPRINT"ERROR I
      N DATA STATEMENTS!":END
Ø679  5Ø PRINT"DONE.":END
8D92  6Ø :
E7FE  49152 DATA 165,43,-45,133,251,165,44,-46
      ,133,252
B2AE  4916Ø DATA 169,Ø,141,36,193,-2Ø,169,147,
      32
CD5Ø  49168 DATA 21Ø,255,32,194,192,-19,16Ø,Ø,
      14Ø
C9CD  49176 DATA 37,193,-2Ø,177,251,133,253,2Ø
      8,3
2Ø58  49184 DATA 238,37,193,-2Ø,2ØØ,177,251,13
      3,254
EA9C  49192 DATA 2Ø8,3,238,37,193,-2Ø,173,37,1
      93,-2Ø
6C15  492ØØ DATA 2Ø1,2,2Ø8,1,96,2ØØ,177,251
E7ØE  492Ø8 DATA 17Ø,2ØØ,177,251,32,2Ø5,-5Ø,18
      9,-142,169
6795  49216 DATA 6,133,211,-236,169,61,32,21Ø,
      255
F8ØF  49224 DATA 169,32,32,21Ø,255,16Ø,2,177
9735  49232 DATA 251,32,213,192,-19,2ØØ,177,25
      1,32
Ø734  4924Ø DATA 213,192,-19,2ØØ,177,251,24Ø,6
      ,32
D99D  49248 DATA 213,192,-19,76,9Ø,192,-19,173
      ,191,192,-19
AC3Ø  49256 DATA 32,167,192,-19,173,19Ø,192,-1
      9,32,167
```

**CHK-LIST** *(cont.)*

```
B343  49264 DATA 192,-19,169,13,32,21Ø,255,165
      ,253
DF3A  49272 DATA 133,251,165,254,133,252,238,3
      6
A6E2  4928Ø DATA 193,-2Ø,173,36,193,-2Ø,2Ø1,2Ø
      ,24Ø,3
936E  49288 DATA 76,18,192,-19,162,Ø,189,1,193
      ,-2Ø
8C3A  49296 DATA 24Ø,6,32,21Ø,255,232,2Ø8,245,
      32
EB74  493Ø4 DATA 228,255,2Ø1,13,2Ø8,249,32
6Ø95  49312 DATA 228,255,2Ø8,251,76,8,192,-19,
      72
AØØ1  4932Ø DATA 1Ø6,1Ø6,1Ø6,1Ø6,32,18Ø,192,-1
      9,1Ø4
FAA2  49328 DATA 32,18Ø,192,-19,96,41,15,17Ø,1
      89
EBFD  49336 DATA 2Ø,193,-2Ø,32,21Ø,255,96,Ø,Ø
E9Ø7  49344 DATA Ø,Ø,169,Ø,141,19Ø,192,-19,141
E8EA  49352 DATA 191,192,-19,169,33,141,192,19
      2,-19,169
A7D7  4936Ø DATA 16,141,193,192,-19,96,162,8,7
      2
6Ø4Ø  49368 DATA 41,127,77,191,192,-19,141,191
      ,192,-19
D24B  49376 DATA 24,14,19Ø,192,-19,46,191,192,
      -19,144
D52F  49384 DATA 18,173,192,192,-19,77,19Ø,192
      ,-19,141
DCA6  49392 DATA 19Ø,192,-19,173,193,192,-19,7
      7,191,192,-19
6Ø32  494ØØ DATA 141,191,192,-19,1Ø4,1Ø,2Ø2,2Ø
      8,215
37C5  494Ø8 DATA 96,13,8Ø,82,69,83,83,32
9A2A  49416 DATA 6Ø,82,69,84,85,82,78,62
AC9Ø  49424 DATA 13,13,13,Ø,48,49,5Ø,51
FE71  49432 DATA 52,53,54,55,56,57,65,66
Ø17E  4944Ø DATA 67,68,69,7Ø,Ø,Ø,-256,37944
```

## CMD Service Center

**AFFORDABLE • FAST • DEPENDABLE**

**Call Today 1-800-638-3263**

Our Team of Technicians are Among the Most Qualified in the Industry!
Why Settle for Anything Less Than the Best?

We repair the following equipment: Commodore C-64, 64C, SX-64, C-128 and C128-D computers; 1541, 1541C, 1571 and 1581 Disk Drives plus CMD Devices. JiffyDOS Installations a specialty. All repairs warranted for 30 days. Minimum charge $35.00 plus parts and return shipping. Contact CMD for authorization before sending any equipment.

**Creative Micro Designs, Inc. P.O. Box 646 E. Longmeadow, MA 01028**

## SODAK ELECTRONICS INC.

**Nintendo**® and **Sega**® Repair and Parts
Authorized **Commodore** Qualified Service Center
Computer Monitor and Printer Repair - All Types
Flat Rate on Most Repairs

Nintendo® and Sega® are Registered Trademarks of Nintendo of America and Sega of America respectively.

**800-201-3004**

**Lamar Nance**
603 S. Mable, Sioux Falls, SD 57103
**(605) 335-3004**

## UPGRADE YOUR COMMODORE!!!

**Refurbished Hardware**

| **Monitors** | | **Drives** | | **Other** | |
|---|---|---|---|---|---|
| 1701 | $169.95 | 1541/C | $99.95 | C64 - | $99.95 |
| 1702 | $169.95 | 41 w/Dips | $114.95 | 64C - | $119.95 |
| 1802 | $189.95 | 1541-II | $129.95 | 128 - | $179.95 |
| 1802D | $209.95 | 1571 | $169.95 | 128D | $329.95 |
| 1902 | $229.95 | 1581 | $179.95 | SX-64 | $359.95 |
| 1902A | $259.95 | MSD-2 | $149.95 | 1660 | $24.95 |
| 1084 | $299.95 | 1001SFD | $129.95 | 1670 | $39.95 |
| 1084S | $339.95 | B.I. Buscard-II | | | $59.95 |
| Mono's | $49.95+ | 1530 Datasette | | | $39.95 |

**Miscellaneous**

| | | | |
|---|---|---|---|
| Books | $10-$15 | Printer Interfaces | $49.95+ |
| Prog's | $5-$20 | Epyx Fastload Cartridge | $34.95 |
| Repairs | $ASK | Super Snapshot v4 | $49.95 |
| Manuals | $7 | Super Grafix Jr. | $199.95 |

***Ask For Anything! We May Have It!***

**New Hardware**

**New APROTEK Accessories**

| | |
|---|---|
| C24-2400 Baud (64/128) | $118.95 |
| User Switch | $44.95 |
| Convert-A-Com | $46.95 |

**New CMD/LMS Accessories**

| | |
|---|---|
| JiffyDOS C64/SX64 'System' | $69.95 |
| JiffyDOS 128/128D 'System' | $79.95 |
| 128 Kernal $59.95 64 Kernal | $49.95 |
| Additional JiffyDOS Drive ROM | $39.95 |
| RAMLink Bs. $239.95 c/w 0 MB | $289.95 |
| c/w 1 MB $319.95 c/w 4MB | $449.95 |
| Real Time Clock (Optional) Add | $29.95 |
| FD-2000 $249.95 FD-4000 | $349.95 |
| **NEW** Super Snapshot v5.22 | $89.95 |
| 1750 Superclone REU 512K | $199.95 |

**J.P. PBM Products By Mail**
Box# 60515, N. Sheridan Mall P.O.
Downsview, Ont. Canada M3L 1B0
Tax-Canada + 7%GST, Ontario +8% PST

Send CDN Funds/15% USA Exchange
15 day Warranty On Refurbished Hdwr.
Allow 4-6 weeks for delivery
**1995** Catalogue Disk (64 Format) — $2

**Shipping**-($0-$25=$4, $26-$99=10%, $100-$199=8%, $200-$499=7.5%, $500+=6%, USA=15%)

## MOVING?

➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔ ➔

Don't forget to let us here at Commodore World know!

Call or write with your change of address 6 to 8 weeks prior to your move so that you won't miss a single issue!

## E X P A N D YOUR SYSTEM

**3 SLOT CARTRIDGE PORT EXPANDERS**

The EX2+1 and EX3 Cartridge Port Expanders bring new expandability to you Commodore 64 or 128. Combine the use of compatible cartridges. Disable cartridges not in use. Change the address your computer finds a cartridge at. Avoid the constant changing of cartridges that puts extra wear on your cartridge port. The EX3 offers 3 vertical expansion slots, while the EX2+1 provides 2 vertical slots and 1 horizontal slot. Both units offer 7 switchable signals per slot, address mapping on slot 2, and a reset button.

**CALL 1-800-638-3263**

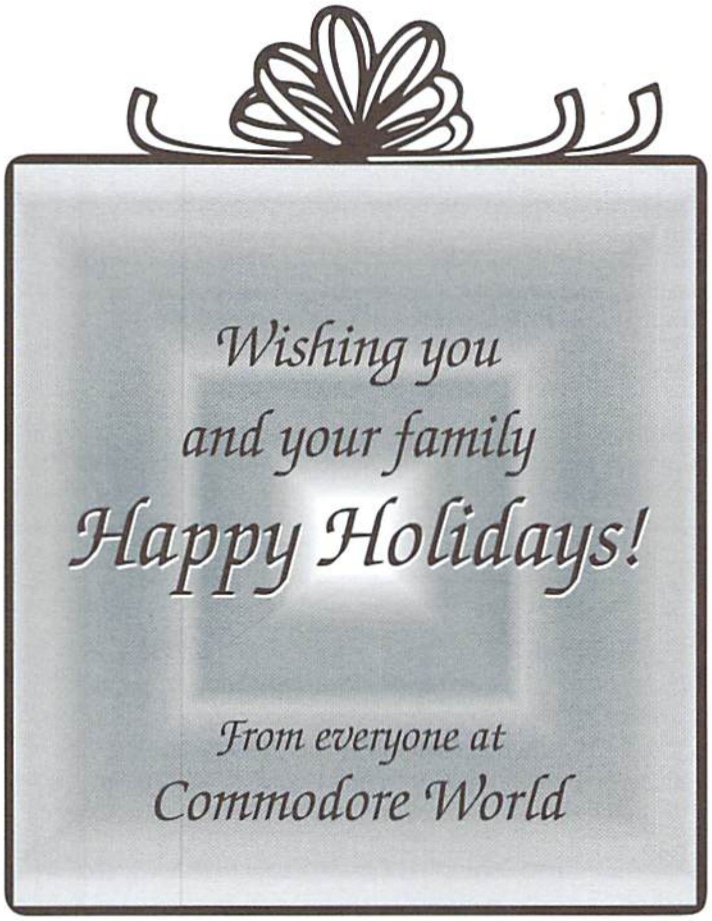
EX3

**$29.95**

EX2+1

**$39.95**

**Creative Micro Designs, Inc., P.O. Box 646, East Longmeadow MA 01028**



## * CLASSIFIED ADS *

**C64/128 PUBLIC DOMAIN.** REQUEST FREE CATALOG OR SEND $2 FOR A DEMO & CATALOG. CALOKE IND., P.O. BOX 18477, RAYTOWN, MO. 64133. VISA-M/CARD ACCEPTED.

**C-64 FOREIGN-AMERICAN Utilities, Graphics, Hacker, Arcade.** 32¢ stamp gets catalog. Home-Spun Software, POB 1064-CW,Estero,FL. 33928

**GEOS PUBLICATION.** The exclusive GEOS Monthly publication. $8.50 yearly, $16.00 for two years. Feb. 1st 1996 rates become $12.00 yearly and $20.00 for two years. Join NOW and save!

**GRASSROOTS #1.** C= history, hardware, help on full 2 sided info disk. Send $3. & system info to Donald Ayers, 75 State Rd. 270W, Sturgis, KY 42459.

**Reconditioned C64 and 1541 Disk Drive** also some used Commodore parts. For information send a SASE to Chuck 30102 Pacific Island Dr., Laguna Nigel, CA 92677.

**RUN, Ahoy, Commodore, COMPUTE!'s Gaz.,** Transactor, Home Comp., High Tech., INFO, etc. D. Marquis, 477 Church Rd., Palmetto, FL 34221-8426, 941-722-8426.

**SUBSCRIBE** to Commodore Gazette; Christopher Ryan; 5296 Devonshire; Detroit, MI 48224-3233; 1 yr. $12 / 2 yr. $24 / 3 yr. $36.

**FOR SALE:** 1541 DISC DRIVE $45, CMD 4-MB RAM LINK $290, SMART TRAK TRACK BALL $45, PERFECT CONDITION. 815-259-2816.

**RUN magazine,** all issues 1987 thru 1992. Commodore MPS 801 Printer, 1541 drive. R. Elliot 228 Star Hill, Swansboro, NC 28584

**Wanted to Buy** Voice Synthesizer for the C64. Prefer Hearsay 1000, but will take any that can speak AND hear voice commands. 813-914-5410 (beeper).

**WANTED:** Leader Board Tournament Disk 1; also World Class Leader Board. Reply to: 810-744-4093 (Roger).

**COMMODORE WORLD**

### Classified Advertising

Subscribers may place non-commercial classified advertising in Commodore World at a cost of $10.00 per issue. Your advertisement may contain up to 150 characters (including spaces). Commercial ads are $10.00 per line (45 characters). Send your advertisement with payment to: CW Classified Advertising, c/o Creative Micro Designs, Inc., P.O. Box 646, East Longmeadow MA 01028-0646.

## ADVERTISERS INDEX

| | |
|---|---|
| BSP | 35 |
| Caloke Industries | (Classified) 56 |
| Centsible Software | 35 |
| Commodore Country | 35 |
| Commodore World | 41, Inside Back Cover |
| Computer Bargain Store | 35 |
| Creative Micro Designs | 5,30,31,34,41,43,55, Back Cover |
| Creative Pixels | 35 |
| Home-Spun Software | (Classified) 56 |
| GEOS Publication | (Classified) 56 |
| Intercorp | 35 |
| J.P. Products by Mail | 55 |
| Loadstar | 3 |
| Mad Man Software | 47 |
| Paxtron | 11 |
| Sodak | 55 |
| Software Support Int. | Inside Front Cover |
| The Underground | 35 |
| Unique | 11 |
| Yanney Software | 35 |

## MOVING?

Don't forget to let Commodore World know. Call or write with your change of address 6 to 8 weeks prior to your move so that you won't miss a single issue!

## DON'T WAIT UNTIL IT'S TOO LATE— RENEW EARLY!

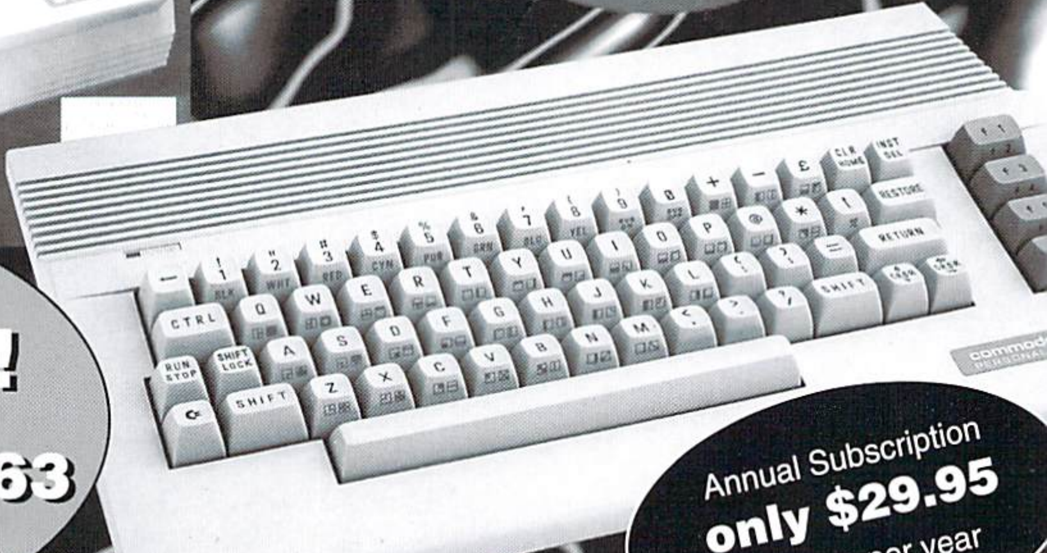
Is your Commodore World Subscription getting close to running out? There's an easy way to check. Look at the mailing label on the front of your copy. There you'll find your subscription number and the expiration issue number. For example:

```
James Smith          12345EXP12
123 Home Street
Grand Rapids, MI  49502-0123
```

Jim's subscription will run out with Issue 12, as indicated by the EXP12 in his subscription code. Jim would be wise to re-subscribe early to avoid missing a single issue of Commodore World!

# COMMODORE WORLD

THE NEWS MAGAZINE FOR COMMODORE 64 & 128 USERS



**STAY IN TOUCH!**

CALL CMD

**1-800-638-3263**

**To Subscribe**

Annual Subscription
**only $29.95**
8 issues per year

**Commodore World** is the publication that will keep you informed in these times when up-to-date information on Commodore computing is so hard to find. Published by Creative Micro Designs, the industry leader in development of Commodore-related products for over six years, Commodore World will supply you with information on what's new, what's still available, and above all else—*where* to get it. If you felt you had nowhere to turn to for Commodore support, turn to the pages of Commodore World for a wealth of resources ready to help you get the most from your computer!

You'll find Commodore World feature articles informative and easy to read; what's more, they're written by leading authorities and experts, many of whom have written for other Commodore-related publications in the past. And Commodore World has something for everyone, whether you're a novice or an experienced programmer.

And while our feature articles help to cover different subjects in each issue, regular columns provide on-going insight into topics of interest to most users. You'll find columns that cover BASIC and advanced programming, and even a column for GEOS programming. And if you prefer being a GEOS user to being a GEOS programmer, you'll find another column devoted to helping you get more out of GEOS. If you want to learn more about using and programming the various peripherals on your system—you guessed it, we've got a column for that as well. Even first-time Commodore users will find a column devoted specifically to their needs.

And there's even more. Departments that cover news, telecommunications, reviews of available hardware and software; even news of what's happening in other Commodore-related publications!

So, if you really want to get the most from your Commodore, there's no better way to get it than Commodore World!

## Columns

Just For Starters - An introduction to the C64/128 by Steve VanderArk

Foreign Exchange - An inside look at the market in Europe by Joseph Gaudl

Graphic Interpretation - GEOS, GEOS and more GEOS by Steve VanderArk

geoProgrammist - GEOS programming techniques by Maurice Randall

BASIC Instincts - BASIC tutorials and type-in programs by Gene Barker

Jim Butterfield's ML Column - Probably the best known name in our industry, Jim covers every aspect of programming in ML (coming soon)

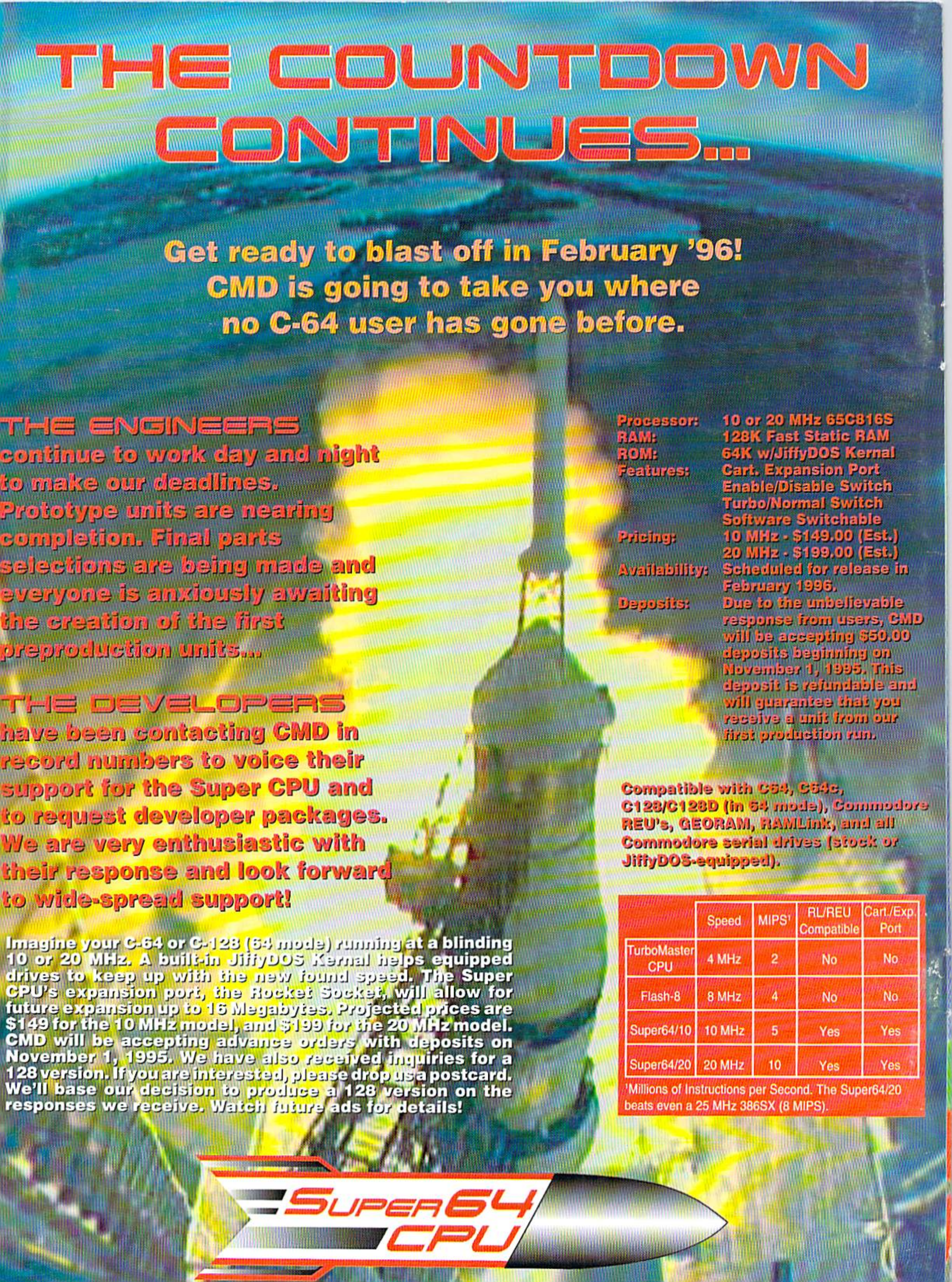
Peripheral Vision - Technical insights to C-64/128 hardware peripherals

Carrier Detect - Exploring every facet of the Telecommunications experience

Over The Edge - Editorial covering various computer related topics and news

## Departments

From The Editor • BackTalk • On The Horizon
Just Asking • The Connection • Top Tips
User Group Connection • Commodore Trivia
BBS Spotlight • Classified Ads



# THE COUNTDOWN CONTINUES...

**Get ready to blast off in February '96! CMD is going to take you where no C-64 user has gone before.**

## THE ENGINEERS

**continue to work day and night to make our deadlines. Prototype units are nearing completion. Final parts selections are being made and everyone is anxiously awaiting the creation of the first preproduction units...**

## THE DEVELOPERS

**have been contacting CMD in record numbers to voice their support for the Super CPU and to request developer packages. We are very enthusiastic with their response and look forward to wide-spread support!**

| | |
|---|---|
| Processor: | 10 or 20 MHz 65C816S |
| RAM: | 128K Fast Static RAM |
| ROM: | 64K w/JiffyDOS Kernal |
| Features: | Cart. Expansion Port<br>Enable/Disable Switch<br>Turbo/Normal Switch<br>Software Switchable |
| Pricing: | 10 MHz - $149.00 (Est.)<br>20 MHz - $199.00 (Est.) |
| Availability: | Scheduled for release in February 1996. |
| Deposits: | Due to the unbelievable response from users, CMD will be accepting $50.00 deposits beginning on November 1, 1995. This deposit is refundable and will guarantee that you receive a unit from our first production run. |

**Compatible with C64, C64c, C128/C128D (in 64 mode), Commodore REU's, GEORAM, RAMLink, and all Commodore serial drives (stock or JiffyDOS-equipped).**

**Imagine your C-64 or C-128 (64 mode) running at a blinding 10 or 20 MHz. A built-in JiffyDOS Kernal helps equipped drives to keep up with the new found speed. The Super CPU's expansion port, the Rocket Socket, will allow for future expansion up to 16 Megabytes. Projected prices are $149 for the 10 MHz model, and $199 for the 20 MHz model. CMD will be accepting advance orders with deposits on November 1, 1995. We have also received inquiries for a 128 version. If you are interested, please drop us a postcard. We'll base our decision to produce a 128 version on the responses we receive. Watch future ads for details!**

| | Speed | MIPS[1] | RL/REU Compatible | Cart./Exp. Port |
|---|---|---|---|---|
| TurboMaster CPU | 4 MHz | 2 | No | No |
| Flash-8 | 8 MHz | 4 | No | No |
| Super64/10 | 10 MHz | 5 | Yes | Yes |
| Super64/20 | 20 MHz | 10 | Yes | Yes |

[1]Millions of Instructions per Second. The Super64/20 beats even a 25 MHz 386SX (8 MIPS).

SUPER64 CPU