TWIN CITIES 128

THE COMMODORE 128 JOURNAL

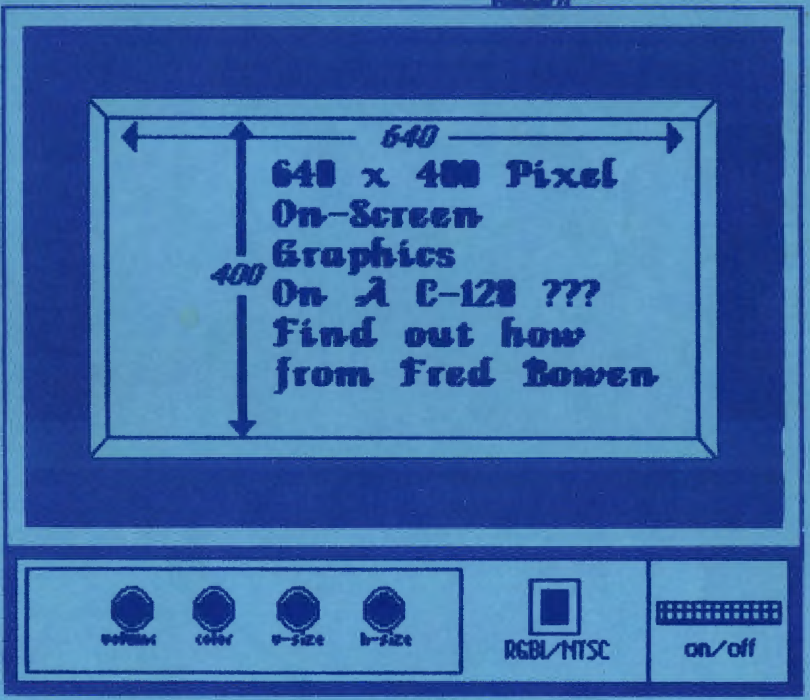
PROUDLY PRODUCED COMPLETELY ON A C-128 IN NATIVE MODE !!!!

$2.50 u.s.

Twin Cities 128 #20

MOUSE MANIA!

Our Mouse Doctor shows you how to repair and prevent Mouse damage!
Plus Mouse & Joystick programming from BASIC 8!



640 x 400 Pixel On-Screen Graphics On A C-128 ??? Find out how from Fred Bowen

Software Reviews: Fontmaster 128, Quick Brown Box, Big Blue Reader 128 (update), C-128 Helper, The Super Chips

Plus...

RUmor/Opinion/Mayhem The C-128 Price & Progress Report

The Assembly Line

We apologize to our subscribers for the lateness of this issue. We thank you for your patient support.
--LJL

Twin Cities 128 is North America's only C-128 specific publication

TWIN CITIES 128: THE COMMODORE 128 JOURNAL

North America's only C-128 specific publication, Issue #20, Copyright 1988 by Voyager Mindtools Inc.

TABLE OF CONTENTS

STAFF

Loren Lovhaug, Managing Editor
C-128 Industry News & Developments, BASIC 7.0, BASIC 8, & COMAL programming, Productivity Applications, Graphics & Cover design, Telecommunications, Special Projects, Child Care, Mental Health Coordinator

Avonelle Lovhaug, Associate Editor
Productivity Applications, Proofreading and Layout, Business & Correspondence, Educational Applications, Child Care, Diciplinarian, Ruthless Telephone Call Screening, Bubble Gum Music Expert

Bill Nicholson, Assistant Editor
Productivity Applications, Proofreading and Layout, Music Applications, Business & Correspondence, Graphics, Entertainment software, Telecommunications, Special Projects, Transportation Specialist, Cubs Fan

Miklos Garamszeghy, Staff Writer
BASIC 7.0 and Assembly Language Programming, Productivity Applications, CP/M Applications, Foreign Correspondent

Bruce Jaeger, Staff Writer
BASIC 7.0 and Assembly Language Programming, Hardware hacking, Photography, Blue Grass Music, Funny Jokes

Twin Cities 128 is published as often as possible and is available for $2.50 per single issue or $25.00 for 12 issues (Only U.S. funds please). To order write: Twin Cities 128, P.O. Box 4625, Saint Paul MN 55104

Information Processing Engines:
Commodore 128 computer, Commodore 128D computer, Tandy Model 100 portable computer, Franklin Spelling Ace Computer
Mass Storage Devices:
Commodore 1571 5.25 inch floppy disk drive, Commodore 1581 3.5 inch floppy disk drive, Commodore 1750 RAM expansion Unit
Printed Output Engines:
Okidata Laserline 6 Laserprinter, Commodore DPS 1101 Daisy Wheel Printer, Star Micronics NX-1000 Dot matrix printer
Video Displays:
Commodore 2002 RGBI/A/NTSC video monitor, Commodore 1902a RGBI/NTSC video monitor
Interface and Communications Devices:
Commodore 1351 proportional mouse, Commodore 1670 modem, Xetec Super Graphix Printer Interface, GeoPrint printer cable
Information Processing Software:
PaperClip III, from Electronic Arts, Pocket Writer 2, Planner 2, and Filer 2, from Digital Solutions, GEOS 128 from Berkeley Softworks, BASIC 8 from PATECH, Superbase 128 from Precision Software, Bobsterm Pro 128 from RML labs, Colorez 128 from B-ware

Over the past three years I have gained a reputation as being somewhat of a Commodore cheerleader, especially for the C-128. Of course such a reputation is a dangerous one for any journalist to acquire because it can have a severe effect on one's credibility. So to balance out my "mild mannered" nature I will create some mayhem by presenting my version of the Academy Awards: The Dead Sparrows.

Our first Dead Sparrow goes for "The most hype surrounding any Commodore 128 product." The winner? Berkeley Softworks. If you read any other Commodore magazine besides Twin Cities 128 you probably have noticed the multiple page ads for GEO-this and GEO-that. There is nothing wrong with slick advertising (and being able to afford it), however, there are things that are disturbing about the Berkworks ads. The GEOS ads are quite long on flowery text and promises and pretty short on content. This would not be such a big deal if the text was not so condescending. The underlying theme found in Berkworks ad campaign, can be synthesized into the following: "Gee, your Commodore 128 was obsolete and without any powerful software until we came along. If you don't buy our software you are probably just a casual user who does not need or deserve good software". The irony here is that although I really like GEOS 128 and its companion programs, they are far from the most powerful 128 software I own. In fact they don't even come close to the sophistication of Superbase 128, The Pocket Series, PaperClip III and several other packages I use daily. GEOS is good, and does some things that no other C-128 package can do but as far as being revolutionary or necessary...

I don't want this column to sound like a beat on Berkworks tirade, but Berkeley gets our second Sparrow, this time for the company best emulating the philosophy of "kicking them while they are down". I am all for making a profit, and I am trying very hard to make sure we make one, but some of the "strategies" Berkworks employs are questionable. First, there are several GEOS add-on products that really ought to have been included with the GEOS 128 system package instead of being sold as separate packages. Programs such as Geowrite 128 v2.1 and the text grabber (found on the Geowrite Workshop 128 package), seem to be more in the vein of upgrades and bug-fixes as opposed to full blown products. I think providing them as inexpensive upgrades (say $15) as opposed to marketing them separately would be more appropriate (this is what they seem to be doing with their Apple II version of GEOS), especially considering GEOS is already a pretty expensive system to use since you really do need a RAM expansion unit and mouse to get the most out of it. Another kick in the side is Berkeley's copy protection. I fully support any software company in their war against software theft, and it seems that copy protection is a necessary measure, but most companies do it unobtrusively. In the case of Berkeley, I have received a number of letters, e-mail, and phone calls that suggest their copy protection is doing a better job of stopping legitimate users as opposed to stopping software pirates.

Speaking of software theft, the Dead Sparrow award for "shooting one's self in the foot" goes to all those that complain that they want more unprotected C-128 software and yet either pirate software or turn their backs on software theft. It is these folks that will destroy the Commodore 8 bit market long before any Commodore executive ever will. Consider BASIC 8. Lou Wallace and Dave Darus (Walrusoft), and Peter Patel (Patech) purposely avoided copy protecting BASIC 8 because they wanted to trust their customers and provide them with a quality product. What most people have failed to understand is that such a move represented an incredible gamble, involving thousands of dollars, and many software companies, including PATECH itself, looked at BASIC 8 as the experiment to see if unprotected software was viable in the Commodore 8 bit world. Unfortunately, the experiment has failed. BASIC 8 has been heavily pirated and it is unlikely that either Patech or Walrusoft will ever produce any unprotected C-128 product again. In addition, many other companies have cited what happened with BASIC 8 as confirming their worst fears about copy protection and the Commodore 128 community. Some of them have also chang their minds about producing 128 products for this reason.

Of course one of the excuses so often cited for software theft is high prices, so with that in mind I award our next Dead Sparrow to the wholesale software distribution companies, who are primarily responsible for inflating the cost of software. Lately I have had a crash course on how the Commodore software industry really works. And the truth is that neither software developers nor software retailers are truly responsible for or benefit from high software prices. It is the distributors (middlemen) who order in large volumes, and demand high percentages, while selling in guaranteed multiple unit lots that really determine how much you will pay for software. In fact, I now know of at least five instances of distributors who demanded that developers increase the price of C-128 products or they would simply not carry the products, even though they were already taking in a bloated 50% to 75% of all profits made on the products.

Our last Dead Sparrow award, for the ultimate extension of the laissez-faire approach to the Commodore 128, goes to Commodore themselves. Commodore will continue to market both 8 bit computers as long as it is profitable to do so, which is a prudent thing to do. But what earns Commodore the bird is their willingness to make money from the eight bit lines without any thought of dropping any money back towards C-128 and 64 owners who have made the lines so successful, even though it has been demonstrated many times over that doing so is very lucrative. Things like dedicating only one person to continued 8 bit development, not actively exploring more 8 bit peripherals, delaying almost indefinitely projects like the RAMdisk software, and the DevPak, as well as demonstrating little excitement over the 8 bits really goes along way to turn off C-128 owners. Well I am off to go bird hunting...I wonder how much postage it is going to cost?

DEVpak Finally sees the light to day!
The long awaited DEVpak is finally available from Commodore. It is hard to believe that we first reported about the existence of this West Chester project two full years ago! Briefly, the DEVpak is a 6502 macro assembler and DEC-EDT style editor for the 128. These tools are totally compatible with the ones we used on the VAX to create the C128, 1571, 1581, etc. operating systems. Also included as part of the package is the RAMdisk software, sprite editors, C64 fast loaders (including source code), mouse drivers (including source code), burst stuff, and extensive documentation on the C-128 math package and ROM routines which can be utilized in your programs. It is available only from Commodore directly. Send $49.95 (check or money order only, US funds) to:

Commodore Business Machines, Inc.
1200 Wilson Drive
West Chester, PA 19380
ATTN: CATS-ORDERS - C-128 Developer Package

Knowing how Commodore fills these kinds of orders, I'll warn you patience will be a virtue...(allow six weeks for delivery).

Solderless Video RAM Upgrade
If you have wanted to upgrade your C-128's video RAM to 64K (as in the C-128D) but have been reluctant to do so because it involves actually soldering chips to your C-128's motherboard, Software Support International may have a solution for you. By late May 1988, they will be offering a solderless video RAM upgrade kit for $34.95. Extended video RAM is ideal for BASIC 8 programming and is being supported in several new graphics and game packages which are under development for the C-128. Software Support International, 2700 NE Anderson Road, Vancouver WA 98661 Suite D13, 206-695-1393.

The Drive Box
Microteq Systems is now manufacturing a new product for the Commodore C128D, 1541 and 1571. The Drive Box will allow setting the device number from 8 to 11, and will also defeat the write-protect without having to cut a notch into the floppy disk, so you can write to the un-notched sides of your disks. The Drive Box has three switches, two for changing device number, and the other for Write-Protect ON/OFF. The unit measures 3 1/8" x 2" x 1". To install the Drive Box requires opening your disk drive, or C128D, and making 6 solder connections to your drive circuit board, but Microteq also offers to install it for you, if you send your drive to them, along with the Drive Box already purchased, for $10 plus shipping. Other than the soldering described above, installation of the Drive Box is completely external and does not require cutting of the C-128D's system unit. The Drive Box, $29.95, Microteq Systems, 1430 9th Avenue South, Fargo, ND 58103, 701-232-4033.

Two new titles from PATECH software
Patech software is proud to announce two new C-128 80 column graphics packages. Page Illustrator 128 and Page Builder 128 are BASIC 8 compatible packages which support, but do not require, extended video RAM as well as the 1700/1750 RAM expansion units. Page Illustrator 128 is a drawing package and Page Builder 128 is a page layout and design program. Page Illustrator is available now, and lists for $39.95 and Page Builder is scheduled to ship in mid-May 1988, at $49.95. PATECH Software, P.O. Box 5208, Somerset NJ 08873, 201-238-5959.

Photogenic Fred..
A picture of the C-128's co-designer, and frequent contributor to Twin Cities 128, Fred Bowen, appeared on the cover of the April 1988 Q-link update newsletter. He is the more diminutive fellow (in stature only), wearing glasses. Now that Fred has been a "cover-boy" for Q-link, what's next...Time, Newsweek, Playgirl..who knows maybe even a guest shot on Geraldo..

Twin Cities 128 T-Shirt Blowout...
I still have a couple of boxes of those stylish blue and gold Twin Cities 128 T-shirts which proclaim one life's more evident truth's: Commodore Built it...We Support it. Also displayed proudly is the Twin Cities 128 banner. These 50% cotton/50% polyester T-shirts are ideal for hitting the beach or hacking out some code and are on sale for just $8.75! When ordering specify: small, medium, large, or extra large. Send your order right away (Avonelle says I have entirely too many T-shirts in my wardrobe already): Twin Cities 128, P.O. Box 4625, Saint Paul MN 55104.

C-128 Hardware Prices, Spring 1988

	New High Price	New Low Price	Used Price
C-128	$265.00	$199.00	$165.00
C-128D	$499.00	$409.00	n/a
1571	$225.00	$195.00	$185.00
1581	$229.00	$169.00	$185.00
1750 RAM	$199.00	$149.00	$129.00
1700 RAM	n/a	n/a	$75.00
1902A RGBI	n/a	n/a	$200.00
2002 RGBIA	n/a	n/a	$200.00
1084 RGBIA	$349.95	$259.00	n/a
1351 Mouse	$49.95	$26.95	$20.00

Note: The data on new equipment is compiled via a random sampling of 30 Commodore dealers and mail order firms across the US. Low prices often represent sale prices at mail order firms and do not include shipping or other fees. Additionally, many local dealers have package pricing for buyer of multiple items which are not reflected in our numbers. The Used Prices are averages calculated from listings in newspapers and on telecommunications networks.

(Editor's note: This article is presented for information purposes only, and efforts have been made to validate the accuracy of its content. As with all projects that involve modification of your hardware we accept no legal, moral, or financial responsibility for the accuracy of the article or consequences of attempting the modification. The modifications described below are not sanctioned or warranted by Commodore, and should be attempted only by those confident with electronic projects. When in doubt we advise that you seek the help of a professional technician.)

The Commodore 1351 Proportional Mouse is a welcome addition to the input devices available for the Commodore 128 / Commodore 64. It seems to respond as well as other comparable mice I've used on other systems, and Commodore was kind enough to make it possible to use the mouse in joystick mode too, for all that older software that doesn't support the proportional rodent.

On the other hand, they seem to have a reliability problem. We have heard of a lot of people suffering mouse and I've had two dead or wounded mice brought to me recently, one that would only work vertically, and one that wouldn't work at all. (My cat Slider also brings dead mice to me, but that's different!) I was able to get both mice to work again, just by wiggling and resoldering the wires from the cord, where they attach to the electronics inside the mouse. Switching the mouse to joystick mode and back again is also, well, clumsy. You're supposed to switch to joystick mode by holding down the right mouse button, and turning your computer on and off. What most of us end up doing, of course, is holding down the right button, unplugging the mouse from the computer, then plugging it back in, all the while the power is on. Besides being awkward, we risk blowing our CIA chips every time we do it. (Or was that FBI chips? I forget...)

I reasoned that switching the +5 volts to the mouse on and off would work the same as turning the computer on and off, and some experimentation proved that it worked. Most of this article deals with installing a power-interupting "reset" switch to the front of your mouse, making for quick and painless switching between proportional and joystick modes.

What you'll need:
Repair only:
Phillips screwdriver
Small, 25-watt soldering iron
Rosin-core (electronic) solder. Thin is better. (An inexpensive ohmmeter would also be helpful for testing.)

Modification:
2.5" piece of thin, flexible wire (Around 24 gauge is fine.)

One Radio Shack normally closed pushbutton switch, #275-1548. (The ones with the black buttons, not the red.) Unlike normal pushbuttons, these interrupt or break a circuit when you press them, so that if we install one in the +5 supply to the mouse, the mouse will always get current until we press the button. The Radio Shack switches come five to a package (for $2.69 as of April, 1988), so get your friends together.

These aren't the greatest quality switches in the world, and are a very tight fit in the 1351 mouse, so if you want to substitute a smaller, higher-quality switch, feel free to do so. You'll obviously have to modify the hole-drilling instruction later on, to match the switch you're using. (Make sure it fits in the mouse case!)

Taking the mouse apart and checking the wires:

Step 1. With a small phillips screwdriver, remove the two screws from near where the cord enters the mouse body.

Step 2. The top of the mouse can now be removed; it rotates on plastic tabs on the side of the mouse opposite the cord.

Step 3. Note the small PC board with the button contacts (Figure 1). Remove the two phillips-head screws that hold it in place, and move this small circuit board out of the way. (Figure 2)

Step 4. If you're just trying to get your mouse to work, that bunch of wires in the lower left hand corner of Figure 2 (where the wires from the cord attach to the bottom circuit board) is the first place to look. Turn on your computer, install a proportional mouse driver program, and try your mouse now. If it doesn't work, try pushing, pulling and otherwise moving these wires, and see whether or not the mouse pointer responds. If it does, you've found a broken wire or bad connection. (I've fixed two out of two dead mice by re-soldering the connections here. It may not work for you-but it's worth a try!)

Step 5. Getting at the solder connections. To get at the solder connections for repair, or to add a "reset" switch to the mouse, disconnect the mouse from the computer again, and remove the rest of the mouse from the case by unscrewing the main mounting screw at the end of the mouse (by the ball - see Figure 2). Try not to let the ball fall out of the mouse and roll across the floor!

Step 6. The solder side of the cord-wire attachment to the PC board looks like Figure 3.

Repair: Testing the Connections. If you have access to an ohmmeter, you can also test the continuity of the wire from the plug-in connector to the circuit board itself. The holes in the 9-pin DIN plug are numbered (get a string light in order to read them!) and connect to the circuit board in the order printed on the top of the PC board.

One of my mice had a wire that broke away when I moved it. That required stripping away a little (less than 1/8") of insulation, and re-soldering it to the board. Be careful here, as you don't have that much wire to work with!

A word about soldering irons: Use a small, pointed 25-watt iron only for this kind of close work. Do not use a big, clunky soldering gun, or one of those 100-watt pipe-melters. If you don't already have a small soldering iron, then it indicates that you're not already an electronics tinkerer, and you probably shouldn't be trying to do this stuff. Also, be careful that you don't accidentally connect ("bridge") any of the solder joints on the circuit board.

Step 7. If you're trying to repair a mouse, test it now by replacing the main circuit board in the lower case, and trying it with a program that drives the proportional mouse.

If it still doesn't work, and the ball moves freely along with the X and Y "wheels," then the problem is electronic, and beyond the scope of this article. (That means that I don't have the slightest idea of what's wrong!) (There is sometimes an electro-mechanical problem with the ball/wheel/sensor mechanism that I can't seem to nail down. If your "X" or "Y" movement stops working, try cleaning everything, make sure that the shafts turn freely, and that the plastic end "bearings" are fully snapped in pace. In other words, futz with it, and it usually works!)

The rest of these instructions have to to with adding a power-interrupting "reset" switch to the 1351 mouse.

Step 7. Drilling a hole for the switch: Screw the mouse case back together, without the "mouse innards".

Step 8. Using a 1/4" drill bit and a variable-speed drill at a relatively slow speed, drill a hole in the front of the case, as shown in Figure 3. (Center the hole on the slot where the two case halves meet.) Do not use a high-speed drill to do this, or you'll have a melted plastic mouse-coffin on your hands!

Step 9. Test fit the switch in the hole. A 1/4" hole is kind of tight for the Radio Shack switch, but we'll use that to our advantage later. (In fact, the Radio Shack switch itself is also a pretty tight fit in the mouse body.)

Step 10. Unsolder wire #7 (as marked on the top of the PC board) by unsoldering it underneath the PC board. (This wire brings the +5 volts from the joystick port to the mouse). See Figure 4.

Step 11. Solder a short piece of flexible wire (about 2.5") to the circuit board at position 7, where you just unsoldered the wire from the mouse cord.

Step 12. Solder one terminal of the Radio Shack switch to the other end of the new wire--see Figure 5. (Remove the nut and lock washer; we won't be using them.)

Step 13. Reinstall the larger PC board in the lower mouse case, and install the single screw that holds it in.

Step 14. Solder the original wire from the mouse cord (the one you'd unsoldered earlier) to the other terminal of the switch; see Figure 6. (Notice that my new wire is stretched kind of tight. That's because I made it too short. It works, but you'll be better off with a 2.5" piece)

Step 15. Wrap the end of the switch with a 1" piece of electrician's tape (to cover the contacts). Lay the switch in the hole you cut in the bottom half of the mouse case.

Step 16. Re-install the upper PC board (the one that has the two mouse button switches on it) that you moved out of the way back in Step 3.

Step 17. Mounting the switch and closing the case. This is the trial-and-error part. Lay the switch as far into the case as it will fit, so that the threads of the switch just reach the edge of the case. This is necessary to ensure that the body of the switch clears the top mouse case when it's put back together. Test-fit the top mouse cover!

We won't be able to use the switch's mounting nut to hold it in place, but the friction of the 1/4" hole does a good job once the top cover is put on. (The right side of the mouse won't close completely using this method.) See Figure 7.

While that worked just fine, it wasn't "elegant" enough, so I used an X-Acto brand knife to slightly trim the holes in the top and bottom cases, so that the switch would nestle in a little more, and the case would close better. This reduced the grip of the case on the switch, so I lightly glued the switch to the bottom half of the case. I used glue from a hot glue gun, because I didn't want to wait for it to dry, it's also easy to pry loose from plastic. "Five-Minute" epoxy, or a silicone-rubber type adhesive would also work. Whatever you do, make sure you only apply glue to the bottom half of the switch, you don't want to glue the mouse case halves together! (Do you see now why a smaller switch would be less of a headache? Make sure the pushbutton switch is of the normally closed variety, so that unpressed it supplies current.)

Step 17. Screw the mouse top case back on.

How to use the modified mouse to change modes: To go from proportional to joystick mode: Hold down the RIGHT mouse button, and momentarily depress the new switch. This will interrupt the current to the mouse, and simulates turning on the computer with the mouse button down. To go from joystick mode to proportional mode: Without pressing the right mouse button, momentarily depress the new switch.

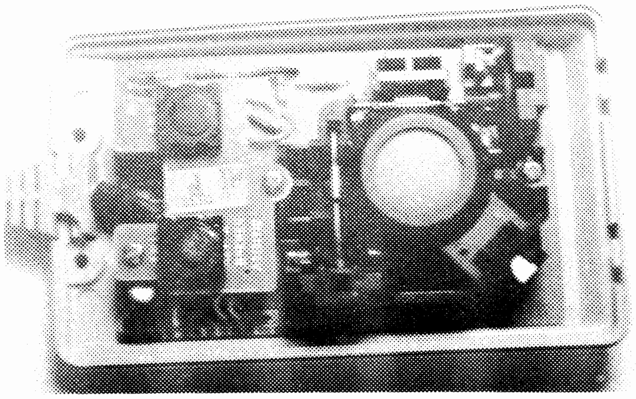


Figure 1

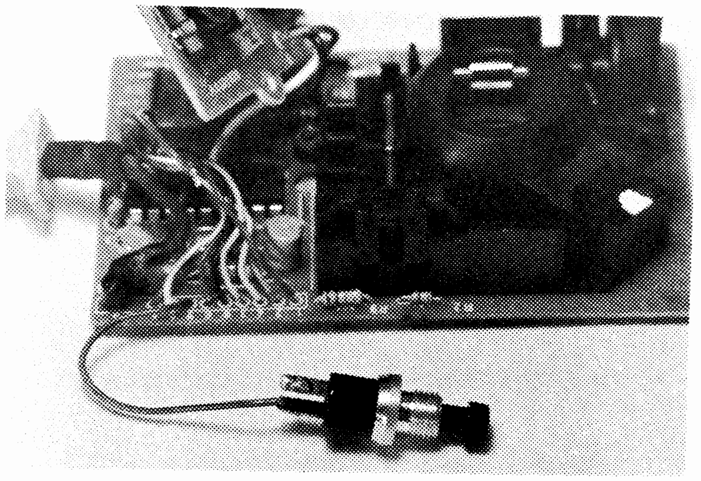


Figure 5

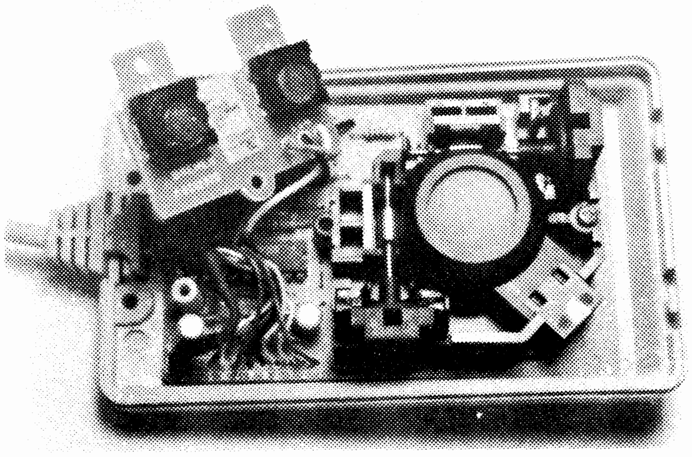


Figure 2

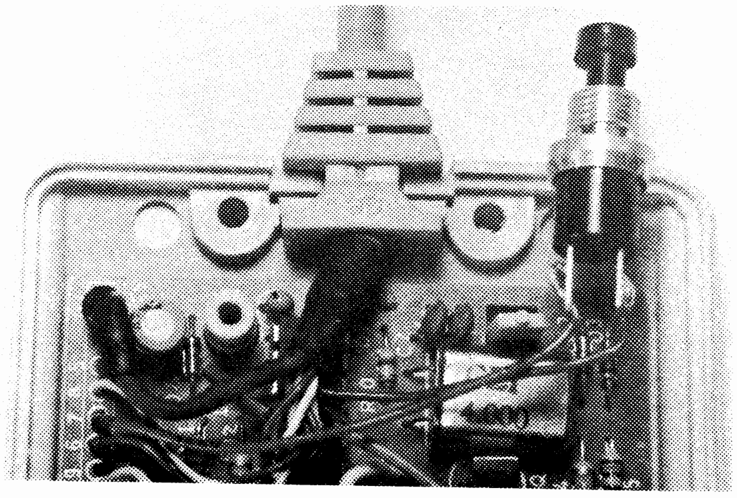


Figure 6

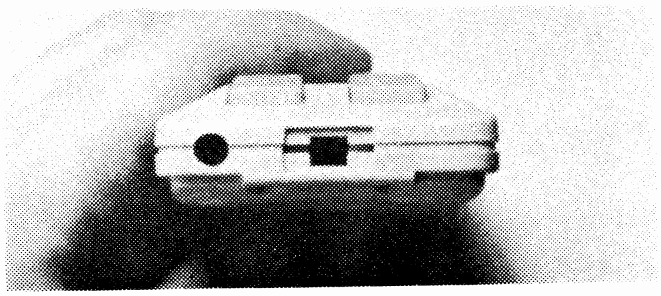


Figure 3

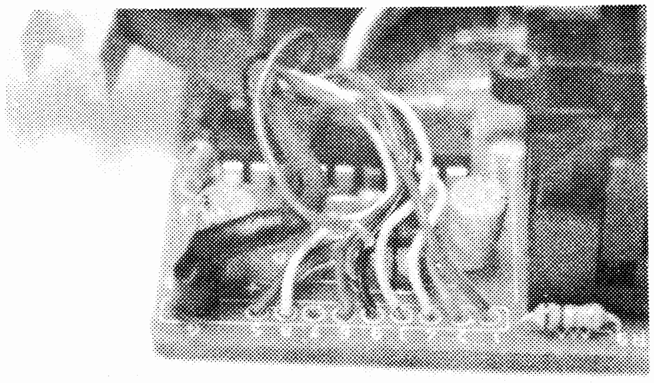


Figure 4

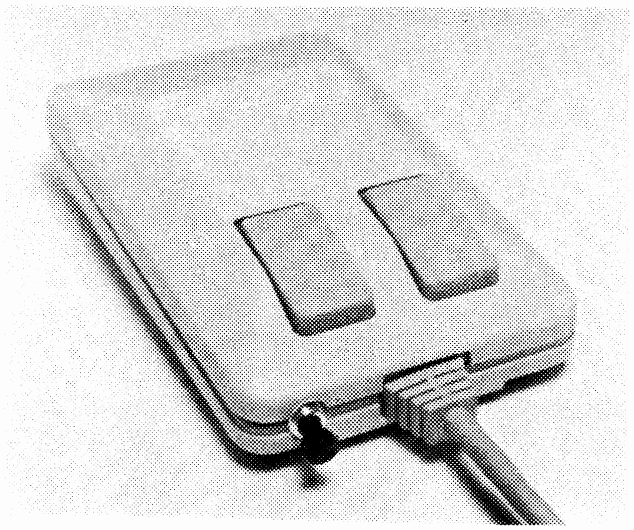


Figure 7

This is a simple method for quick'n dirty interlaced displays on C128's with 64K video RAM and C128D's. It is pretty forgiving of monitors, too. But there is a drawback: it requires the processor to maintain the display and it is not very tolerant of sloppy timing. This particular example will work only on C128's with 64K display RAM and requires that you own BASIC 8, folks with only 16K memory can get a hint of what's going on: just change the 400 in line 20 to 200, the 399 in lines 60, 70, and 80 to 199, and the 1 in line 10 to 0. You will see TWO copies of your screen.

Here's how it works. The typical video display is refreshed 60 times a second. Each frame is delimited by a vertical blanking interval, which gives the hardware time to move the beam back to the top of the frame. The VDC signals this occurrence by setting a bit in its status register, bit 5. During this time, our processor can execute an amazing number of instructions, and the VDC will still do whatever we desire. We desire an interlaced graphic display. What does interlaced mean? Well, simply put it means we want to double our display resolution by displaying twice as much data. We can do this if we sacrifice something: a steady image. Displays are made up of lots of lines of dots, which must constantly be redrawn or "refreshed", or else they lose luminescence, dim, and go out. The persistence of these dots is a function of the display tube in your monitor. Anyhow, the VDC can be instructed to display data in between the normal scan lines on alternate frames. This is interlace mode. When this happens, the normal lines are displayed on even frames, and the half-scan lines are displayed on odd frames. The refresh rate for any given line is effectively cut in half causing it to dim more than usual, and hence you see the display flicker. As many of you have heard, the VDC has some problems displaying interlaced screens; it wants to display the same data on odd & even frames, which does not add to our total resolution. This hack gets around that by manually changing where the VDC gets its display data from every frame. For the convenience of BASIC 8, I use the address increment per row feature of the VDC. This lets me draw on one contiguous hunk of memory. Even frames will display even lines of my bit map on the normal scan lines, and odd frames will display odd lines of my bit map on the "in between" or half scan lines. Well, almost. There is one more difficulty, and that is we cannot tell which frame is odd and which is even. This means we have a 50-50 chance of getting an image that has the odd lines displayed before the even ones, yielding a blurry sort of image. That is why the ML routine looks at the keyboard: only you, with your eyes, can tell the program if it got things backwards. Simply bang the (return) key and the program will switch things around for you.

This hack can be improved or optimized for your purposes. This is just to get you going, if you have a C128D or have already increased your C128's display RAM to 64K, you have everything else you need! Later, we'll explore other ways to get more resolution outta the VDC- I've got a 752x600 display on the C128D beside me right now...

```
10  @walrus,1:rem init BASIC-8
20  @scrdef,0,0,0,640,400,0,0:  rem define screen
30  @screen,0: @clear,0,1,0:    rem clear screen
40  a$="abcdefghijkmoprstuwxyz This is an INTERLACED Bit Map!"
50  b$="ABCDEFGHIJKWXYZ 1234567890"
60  for y=0 to 399 step 8:@char,254,0,y,1,1,2,a$+b$: next y
70  @line,0,0,0,639,399,0,1:    rem draw line from TL to BR
80  @box ,0,0,0,639,399,0,0,0,1:rem draw box around screen
90  @circle,480,100,0,75,1:     rem draw circle
100 sysdec("cdcc"),1,8:         rem set interlace mode
110 sysdec("cdcc"),80,27:       rem set adr incr/row to 80
110 sysdec("b00"):              rem begin display

00B00  A2 0D     LDX #+13     ;reg 13 is display enable begin
00B02  AD 00 D6  LDA $D600
00B05  29 20     AND #%10000
00B07  F0 F9     BEQ $0B02    ;wait start of vertical blank
00B09  20 DA CD  JSR $CDDA    ;read current display start addr.
00B0C  49 50     EOR #+80     ;toggle between 1st/2nd scan line
00B0E  20 CC CD  JSR $CDCC    ;write it back
00B11  AD 00 D6  LDA $D600
00B14  29 20     AND #%10000
00B16  D0 F9     BNE $0B11    ;wait for end of vertical blanking
00B18  A5 D0     LDA $D0      ;check keyboard
00B1A  F0 E4     BEQ $0B00    ;nothing, loop forever
00B1C  AD 4A 03  LDA $034A    ;get the key
00B1F  C9 0D     CMP #+13     ;is it a RETURN?
00B21  D0 0E     BNE $0B31    ;no, so exit
00B23  20 DA CD  JSR $CDDA    ;toggle display start again to
00B26  49 50     EOR #+80     ;sync up with odd/even frames
00B28  20 CC CD  JSR $CDCC
00B2B  A9 00     LDA #$00     ;clear the keyboard buffer
00B2D  85 D0     STA $D0      ;ndx
00B2F  F0 CF     BEQ $0B00    ;continue display always
00B31  A9 00     LDA #$00     ;user hit a key, exit
00B33  20 CC CD  JSR $CDCC    ;restore start to 1st scan line
00B36  60        RTS          ;return to BASIC program
```

Since I first used a RAM expander cartridge on my VIC-20 many years ago, I have been intrigued with the idea of battery powered RAM which would retain its contents when the power to the computer was switched off. Such a device could be used as a non-volatile RAM disk or to store often used utilities which could be available almost instantly. Well, as with all dreams of youth, the technical challenge proved to be too much for me and I quickly moved on to other ideas. When I finally gave up the trusty old VIC for a C-128, I was no longer concerned with RAM expansion cartridges, after all the 128 had over twenty times the amount of RAM in the basic VIC -- more than plenty to play with.

All this pre-amble brings me indirectly to the point of this review. At the recent World of Commodore Show in Toronto, I came across an interesting accessory being sold for the C-64 and C-128: it was a battery backed RAM cartridge! Externally, the Quick Brown Box, as it is known, looks like any other C-64/128 program cartridge but contains 16k, 32k or 64k of battery backed CMOS static RAM (depending on which size you buy) instead of the usual ROM chip. The Brown Box (which is named after its developer) plugs into the cartridge slot at the rear right of the computer and is compatible with either the C-64 or C-128 (in both 64 and 128 modes but not directly in CP/M mode) by selecting a switch on the box. When used with a C-128, the switch will determine the operating mode of the computer. For example, if the switch is set to C-64 mode, the C-128 will start up in C-64 mode. If the switch is set to C-128 mode, then the computer will come on in that mode. The box can be used as a RAM disk or as sort of an EEPROM cartridge (electrically erasable read only memory) to store your custom programs on. Of note to C-64 users is that the box contains a reset switch and can be made invisible to the C-64 by the flick of a switch. (The only way to make it invisible to the C-128 is to disconnect it or to turn off the box control register at $DE00 by a POKE 56832,128.)

This little gem is truly a programmer's dream. For people who like instant access to their software, the box can be used as a RAM disk with a difference: the programs are retained when the power to the computer is turned off. The internal 3 volt lithium battery of the box is said to have a life of ten years or more (probably longer than the expected life of the computer, but there does not appear to be any way of changing the battery, short of breaking open the case). The battery will maintain the contents of the box even when the box is disconnected from the computer.

The file management software provided with the box (both C-64 and C-128 versions are provided) can be used to load programs from a floppy disk into the box (both BASIC and machine language programs can be loaded into the box) or save files from the box to disk. Machine language program addresses can be designated in either decimal or hex. With the C-128 version, you can also designate the BANK that the machine language program should execute in. With both the versions, you can even designate any program as an autoboot program. This means that your selected program will automatically start when you turn on your computer.

When in RAM disk mode, you can LOAD or SAVE the current contents of the computer's memory into a RAM disk file using the box's command wedge with a few simple keystrokes. Programs stored in the RAM disk are accessed by a user assigned a two character ID code (such as P1 or ZX) which is used to LOAD, erase or RUN the program from the RAM disk, along with a six character descriptive filename. Programs can be made to execute automatically (default) when loaded or marked as load only by including a "<" character in the filename portion. Multi-part BASIC programs can be chained using a "+" as the first character in the filename portion.

To save the BASIC program currently in the computer's memory you would type in, for example: ^ZX "PROGRM" <return> where ^ is the up arrow key to the left of the restore key. To run this program later, all you need to do is type in: *ZX <return> This allows rapid switching of program files if, for example, you are editing several BASIC programs at one time. Later, the files saved to the RAM disk can transferred to a floppy disk if desired.

You can save the contents of the box in two ways: as individual files or as one large block of RAM. The latter method is used to backup the entire contents of a box and can also be used swapping all of the files in one box to those of another. (The manual suggests that you make a backup of the contents of your box when you first get it. That way if you decide to do any custom programming you will always be able to return the box to its "as bought" state with the default software. Note: This initial backup must be done in C-64 mode even if you are using the box with a C-128.) The utilities required to perform these loads and saves are provided on the box itself or in the form of an update disk. Separate versions are provided for the C-64 and the C-128.

The current version of the user manual is well written and contains a great deal of information for both the novice and advanced user. It not only gives you the basic instructions for operating the box with the included software, but tells you how to adapt existing programs to run in the box environment and how to program the box memory control registers directly. The print may be small, but everything you need to know to use the box or develop you own applications for it is contained in the manual.

For the advanced programmer, the box makes an excellent EPROM emulator. You can even partition the box so that most of it is used in RAM disk mode, but one or two 16k blocks are reserved for anything else that you wish. You can set it up as a simple external cartridge and write your program to the box for testing in its normal address space before burning in the final EPROM. It also makes an excellent home

When I first began word processing only a few years ago, the most exciting thing to me was that I could type a document, save it, load it later and make changes to it, save it again, always knowing that any time I wanted, I would edit my document and reprint it. However, since the advent of personal computers such as the Macintosh and the Amiga, the focus has moved from mere text processing to text and graphics processing combined. Now, instead of worrying only about the quality of our text, we must also worry about fonts, graphics, borders, and other sophisticated paraphanalia. And although many current word processors on the market for the C-128 allow us to manipulate text in an impressive manner, there is only one word processor that I can think of which is just as sophisticated as the best such as Pocket Writer 2 and Paperclip 3, and also includes these important graphics and fonts: Fontmaster 128.

To be totally honest, I was not looking forward to examining Fontmaster 128. I don't tend to be a person who is excited by fonts and graphics, but I do enjoy a full featured word processor. I was pleasantly surprised to find that Fontmaster 128 is just that, a complete word processor, but with some additional features which might prove very useful to C-128 owners.

The Fontmaster 128 package comes with 3 disks: the program disk, the supplemental disk (including foreign languages) and the Spell Master 128 disk. It also includes a manual for the program, a manual for the spell checker, a quick reference guide, and a dongle key. Since the program is dongle protected, there is no dangerous head banging to your disk drive. However, the dongle key doesn't fit the joystick port, but instead fits into the cassette port, although it is made so that an interface power plug will fit into the dongle.

Upon loading the program, Fontmaster 128 brings you immediately to the main menu. The system main menu has seven catagories: Word processor, System setup, Font Creator, Character set creator, Text translator, Graphics converter, and Back to Basic. The system setup area allows you to choose the different colors for your screen, different default drives for your fonts, text, and character sets, whether the sound is on or off, the type of printer and interface you are using, and the capabilities of your printer/interface. You can set your own defaults, and then test the printing right from this menu before you save it to disk. You can also save different setups, in case you use different ones in different situations. I would advise changing the colors immediately, as I had trouble seeing the help screens on both my Commodore 2002 monitor and my 1902A monitor. Most printers are supported, and they are constantly adding new ones, from the looks of my update file located on one of the disks. ["Update.info is a file provided to the owner which includes any new features or changes which are not a part of the manual.]

When you enter the word processing area, there is a few moments for the computer to set up. However, the program displays an interesting graphic of the fontmaster logo, which changes a few times while you wait. (It looks really neato!) The word processing screen then appears. There is a three line help and information area at the top of the screen, and then a one line margin/tab underneath the help area. The rest of the screen is devoted to the entering of text on the screen. The standard editing features are all in play here. Block copy and move, find and replace, etc. You name it, you got it.

Not only are the standard features of most word processors implemented here, but also the best features are included. One example of this is the ability to create a form letter, with which you could merge variable information such as names and addresses to do a mass mailing, etc. Another feature implemented is multiple column output. Fontmaster 128 allows up to four columns at a time, all of which can be variable widths, and can overlap, if you wish. However, the method that Fontmaster 128 uses means that if your printer cannot automatically roll the paper back up to the top, you will have to manually roll the paper back up to the top. Since many of the newer printers have added this feature, perhaps it is only a matter of time before this becomes a standard practice.

The disk commands supported by Fontmaster 128 are fairly complete. I mention this because I despise software which limits you to "save" and "load". In addition to these commands and directories, you can also display an error from the error channel, erase files, send any disk command, merge files together, verify the file in memory against the one on disk, save files or blocks to SEQ format, or insert a file in the middle of the current text. Pretty thorough stuff, huh? I was disappointed that Fontmaster 128 does not allow you to load a file directly from the directory. This is one of my favorite features of Pocket Writer 2 and PaperClip, and although I'm getting tremendous memory retention practice, that's hardly what I'm interested in when I'm word processing. Instead, you must load your file by typing in the name of the file exactly, without the ability to even look at the directory at the same time.

Fontmaster 128 is a command based word processing program, which means that in order to see how the text will actually look on paper, you must use the "view" mode, or just print it out. Since the program has to adjust for fonts, etc., the view mode tends to be a little slow. I was disappointed with this, but the program does offer a "quick view", which is faster but less comprehensive, showing only the text currently on the screen.

Like all good word processors, Fontmaster 128 allows you to add several special features to your text, such as pitch changes, boldface type, underlining, italics, superscripts and subscripts. In fact, Fontmaster 128 will also allow you to print with compressed or expanded print. However, even features. Fontmaster 128 will print extra tall height and micro height. If you want to print your text proportionally instead of standard, you don't even have to choose a special font. Fontmaster 128 will automatically switch to proportional. If you are using a script font, Fontmaster 128 includes a "Konnect" pitch of 13 cpi which allows the characters to run together, thus creating a smooth output.

Judging by the name of Fontmaster 128, you would guess that it is largely a word processor which can create great looking fonts. However, to merely say this would be greatly underestimating the abilities of this program. The program comes with several different fonts on disk for the owner to choose from. When working with a document, you simply load a font into a "font slot". Then, if you want to switch to that font, a simple command code will start the font that you choose. There are nine font slots, which means you would use up to nine fonts in a document at any one time. This would seem to be a more than adequate amount. There is also a border font included which will produce great borders for your documents.

In addition to different font styles, there is also several foreign languages supported, including: Arabic, Danish, German, Greek, Hebrew, Inuktitut, Italian, Korean, Norwegian, Russian, Spanish, and South African. (Character sets to see these fonts on the screen are provided as well.) Yes, but (I can hear disbelievers saying) many of those languages write their text from right to left, not left to right as we do in the Western half of the planet. How will this program cope? If you need to type your text from right to left, simply tell Fontmaster to reverse the screen, and it will enter your data backwards. And, Fontmaster 128 also offers a "push" option if the foreign language reads from right to left, unless you are reading a number, which is read conventionally left to right. I must be totally frank, these problems had never even occurred to me, but the the author of Fontmaster really gave some thought to his topic.

If you prefer, you can create your own font with Fontmaster 128's great font editor. This editor allows you to shift your character's pixels to the left, right, up or down, and will also inverse its position left to right or up to down. Fontmaster supports some light pens (Flexidraw's is the example they give), so you can create and edit fonts with this peripheral if you choose. Fontmaster has an option for all fonts to be printed proportionally if you choose, and this feature is controlled by the way it reads individual characters. In the font editor facility, an arrow point up at the character moves across the bottom of the character grid. This arrow determines where Fontmaster sees the width of the character if it is printed proportionally.
Fontmaster 128 also has a buffer for a character which can be stored and recalled for future use. You may also create character sets using Fontmaster 128 which will be seen on the screen, but not printed.

If you are looking for a program which will allow you to incorporate graphics into your word processing documents, Fontmaster 128 can help you. Fontmaster 128 has a special formatter which allows you to name a graphics file you wish to print with your document. You can place this where ever you want in the text. You can also print your graphic double width with a slightly different command. You can also use a command to specify the exact column you wish the graphic to begin at. And if you wish white to print black, and black to print white (reverse video), you can choose the "negative option".

Of course, a graphics option would hardly be complete without a file converter so that you could convert your graphics into a format Fontmaster 128 could use. Fontmaster gives you the ability to convert Print Shop graphics or any hi-res picture to its format. I missed the ability to convert a GEOS file here, I think that a GEOS converter would have added an important section of graphics files. After starting the graphics converter, Fontmaster 128 reads the file you specify. You then "lasso" the picture you wish to save, and give the file a new name. Fontmaster 128 will also allow you to build pictures of several smaller graphics (currently only PrintShop graphics). Still, this is yet another unusual option from Font Master 128.

Spellmaster 128 is the spell checking program which is included with the Fontmaster 128 program. It is, to say the least, impressive. The main dictionary contains over 100,000 words to check against the owner's document. There is also an option for a dictionary to be created by the user, for words that the owner might use on a regular basis, like his name, etc. Spellmaster 128 divides this dictionary into five parts: the main dictionary, abreviations, British words, proper names, and vulgar language. You can request Spellmaster 128 to ignore dictionary sections (other than the main dictionary) to help speed up the checking process. For instance, if I am fairly sure there are no vulgar words in my document, I can request Spellmaster 128 to ignore that dictionary part, and it won't have to check that dictionary part for words, thereby speeding up the check. Spellmaster 128 also takes advantage of disk drive differences. For those folks who own a 1571 or a 1581, greater spell checking speed can be obtained. RAM Expansion owners can really increase their spell checking time to up to 15 seconds (WOW!) for a 1750 REU. If you own a 1541 clone (with your C-128?), Spellmaster 128 will adapt itself to your drive.
If it has trouble reading your drive, adjustments can be made to get your drive to work, and this is all documented in the manual, which is very thorough. [This reminds me to praise Xetec for using dongle protection on this software. Software using this kind of protection is less likely to cause problems for person with Commodore drive clones. Not many of us, I admit, but they should be considered, too.]

Spellmaster 128 allows the user to ignore a misspelled word, replace it with the correct spelling, add it to the user dictionary, or look it up in Spellmaster's dictionary. This

Fontmaster 128 Review Continued...
program will allow you to "lookup" word in its dictionary without spell checking the entire document! If you are unsure of how to spell a word, just pull up the spell checking option menu. Then choose "S" for word search. The program will search for a word based on the pattern you give it (wild cards, etc. can be used.) What a great idea! The spell checking facility of this program is one of the best I've seen for C-128 software, and certainly a good reason for purchasing Fontmaster 128.

I must admit I had a few problems trying to print with Fontmaster 128, at least with our laser printer, the Okidata Laserline 6 (which Fontmaster now supports). When I tried to incorporate graphics into a document, my printout would always garble. And I had even bigger problems when I tried to print proportionally - what a mess! I felt that the manual could have offered more in the way of troubleshooting printing difficulties. Although the manual does cover this topic, it is pretty limited to: "If you printer doesn't print, try turning it on" type of stuff. Perhaps a suggestion that some printers aren't very friendly when it comes to printing graphics with the text, or something of that nature.

To be totally honest, I was also somewhat disappointed with the fonts. There are sure a lot of them, and if I was working with foreign languages I'd say that was all I ever needed, but I didn't care that much for any of the normal english fonts. There weren't bad, just not spectacular. However, I would imagine that this is a minor point since you can create your own fonts, and I'd bet with some effort and a good programmer at your side, you might even be able to convert other fonts from other programs to a Fontmaster 128 format.

My only other gripe about Fontmaster 128 is that it doesn't word wrap the text as you type it in. I know, faithful readers, that you have heard me say this a million times, but it seems to me that it isn't that complicated to incorporate this feature into your software, especially a word processing program. Perhaps I am way off the mark in expecting this feature, but I don't think so. I believe most folks prefer word wrap when word processing. I'm sure that given time, the folks at Xetec will come around.

Fontmaster 128 has many of the items C-128 owners have been craving: full featured word processing combined with multiple fonts and text enhancements, and the ability to incorporate graphics, not to mention a superior spell checking facility. Persons who need to work with foreign languages will find it unbeatable, and disillusioned GEOS users may find a new home. In fact, I might even say that Fontmaster 128 is as close as we have come thus far to desktop publishing software for the C-128 in its native mode.

Brown Box Review Continued...
for custom ROM based software such as word processors, BASIC extensions, disk wedge utilities, etc. My only concern is that a lot of copy protected software will not work from the box, which is a pity. However, one very good word processor for the C-64, called the "Write Stuff", has been created in a version specifically for the box. A nice feature of this word processor when used with the box is that your document is automatically saved in the box as you type it. Thus even if you have a power failure in the midst of creating your magnum opus, you still have a copy of all your work to date in the box! A C-128 version is expected soon.

The Brown Box is very simply controlled from one 8 bit write only latch location at $DE00 of the I/O space of both the C-128 (BANK 4 to 13 or 15) and C-64. On the C-128 the external memory of the cartridge can be mapped into the computer's address space in 16 k byte chunks (8 or 16 k on a C-64) in the normal external cartridge memory area of $8000 to $BFFF in BANKs 8 to 11 and 13. In these BANKs, the box overlays the BASIC ROMs. Commands are provided in the latch to select which 16k bank of box RAM is in context and whether the RAM has been enabled as read/write or read only. The read only mode protects the box contents from being accidentally altered because once it has been mapped in, the contents can be changed with a simple POKE. The manual cautions against having too many cartridges (with an expansion chassis) connected to the cartridge port at once. If more than one cartridge is turned on and/or the box is turned on at the same time, the conflicting data signals may corrupt the contents of the box. It will also increase the battery drain. This situation should not occur because it is likely to cause a complete system crash anyway.

One additional note for C-128 mode users of the Brown Box: When you press the C-128 reset button to boot up auto-booting disk software (such as the CP/M boot disk, for example) you will normally be sent directly into the box menu software instead of your intended application software. However, this minor inconvenience can be over come by typing in "BOOT" to do a soft re-boot of the disk. Or, if you have no need to use the autoboot feature of the box, you could write a simple one line BASIC program consisting of the line "10 BOOT" and assign it as the autoboot program in the box. This will perform a soft boot from the disk automatically and behave almost as normal.

I mentioned previously that the box was not directly compatible with CP/M mode. This may be true for the supplied software, but I can see no reason why its extra memory cannot be mapped into the CP/M memory space using custom programming. Perhaps it would be a good place to stash some pop-up type CP/M utilities.

Available from: Brown Boxes Inc., 26 Concord Road, Bedford MA 01730, (617) 275-0090: 16K box for $69.00, 32K box for $99.00, 64K box for $129.00 (+ $3.00 shipping and handling, Today, every major brand of computer today has a mouse input device available for it, including our own beloved Commodore 128. In fact, the Commodore 128 has not just one, but in fact two mouse devices available for it. Both mice look identical to the two button mouse created by Commodore for the Amiga computer but in reality they behave quite differently. The first mouse, model 1350, emulates a standard 8 direction joystick, and although it has two buttons, only the left button is really connected. The second mouse, model 1351, is a more accurate device which allows a complete range of motion and is much more responsive and both buttons are active.

(Note: From this point on in this article whenever I use the word mouse I will be refering to either the 1350 or the 1351 mouse. Since the 1350 emulates a joystick, if you do not own either mouse you may use a standard joystick as a substitute for the 1350 mouse.)

Before BASIC 8, using these devices in your BASIC programs was often a tedious task, especially on the 80 column screen. But BASIC 8 includes two very convenient commands for adding mouse control to your programs.

The @mouse command is used to communicating with BASIC 8's interrupt drive mouse reader. Once activated, BASIC 8's interrupt drive mouse reader automatically checks for mouse movement 60 times per second. The @mouse command has four different ways that it can communicate with the interrupt driven mouse reader depending upon the particular parameters and syntax used as described below:

@mouse,0 - Turns off the interrupt driven mouse reader.
@mouse,1,device,x,y,joystick increment - Activates the interrupt driven mouse reader and sets the initial location of mouse at a specific coordinate as defined by the x and y parameters. The device parameter can be defined as either 0, representing a 1351 mouse connected in port 1, or as 1, representing a 1350 mouse (or joystick). The joystick increment parameter is necessary only when using a joystick or a 1350 mouse. This parameter controls the number of pixels a pointer will be moved when the 1350 mouse or joystick is moved in any direction. This parameter is designed to make 1350 mouse/joystick use a little faster.
mx = @mouse,2,0 - Defines the variable mx as the current horizontal location the mouse.
my = @mouse,2,1 - Defines the variable my as the current horizontal location the mouse.
(Note: Any numeric variable can be used as a substitute for mx and my.)

The @ptr command controls the placement of a pointer (pointers are moving pictures which are used mostly to graphically represent the on-screen mouse position) and like the @mouse command has multiple syntax variations:
@ptr,0 - Turns off the pointer
@ptr,1,x,y,pointer definition,height - Draws a pointer at a specific coordinate as defined by the parameters x any y. BASIC 8 allows for multiple pointer definitions at any given time. By default BASIC 8 is configured with 8 different 16 x 8 pixel pointer definitions, each with a different shape. Using the pointer definition and height parameters you can control the vertical size and which of a possible 16 definitions (numbered 0 - 15) you use. The pointer does not disturb any graphics data beneath it.
@ptr,2,x,y,pointer definition,height - Acts the same as above but leaves a trial of pointers.

The program on the following page will operate on any C-128 regardless of VDC RAM size, after BASIC 8 has been loaded and demonstrates the use of these commands and a some event programming concepts. Event programming is the current buzzword for writing computer programs that respond to certain specific external events, such as pressing a special key or moving mouse or pressing a mouse button. I suggest that you type in this example and run it a few times as I think it will enhance your understanding of my explanation.

The key to effective event programming is the development of specific rules which determine what is going to happen when a certain "event" occurs. Often these rules can be expressed as conditional statements. In my example the "event rules" are as follows: 1. If the mouse pointer is on a box then change the box colors to purple and black. 2. If the mouse pointer leaves a box then return it to its original colors. 3. If the left mouse button is pressed while the mouse pointer is one of the boxes then pleasant music is played (the music is different for each box). 4. If the left button is pressed while the mouse pointer is in the far left box, the program ends after the pleasant music is played. 5. If the right button is pressed at anytime a rather unpleasant sound is played. (Note: Since the 1350's right button is not active and a joystick does not have a right button this option is not available unless you have a 1351. *Special tip: When writing event code, keep in mind that it is sometimes more important to keep track of what is not happening as opposed to what is happening.*

Here is how the program works: Lines 100-270 perform various setup functions. Line 110 sets up a trap statement to redirect the program to line 350 in case an error occurs. Line 350 simply gets us out of graphics mode and shuts down the program. Lines 120-130 setup and clear a 640 x 176 pixel 8 x 8 color cell graphics screen with a background color of black and a foreground color of yellow. Lines 140-200 draw four boxes (windows) on the screen with the colors specified in the data statement (line 140). The colors are stored in the arrays bc() (background color) and fc() (foreground color) for reference later when we need to restore the default colors of a box. Lines 210-240 display the instructions on the screen. Line 250 defines four variables which are crucial to our "event programming". These variables denote the location (xo and yo) and the size (dx and dy) of the area where the boxes are located. These

```
100 rem ** setup screen, variables, pointer, and mouse **
110 trap 350
120 @walrus,0:@mode,0:@screen,2
130 @color,0,13,0:@clear,0
140 data 8,13,2,15,15,4,4,13
150 for i=0 to 3
160 read bc(i),fc(i)
170 @windowopen,48+(i*136),72,136,32,1:@clear,0,bc(i),fc(i)
180 @char,254,4,12,1,2,2,chr$(3)+"Box"+str$(i)
190 @windowclose
200 next i
210 @char,254,3,8,1,2,2,chr$(3)+"Move mouse pointer to a box and push"
220 @char,254,3,16,1,2,2,chr$(3)+"the left mouse button. Pressing the"
230 @char,254,3,24,1,2,2,chr$(3)+"left mouse button when the pointer is"
240 @char,254,3,32,1,2,2,chr$(3)+"on box 3 will terminate the program."
250 xo=48:yo=72:dx=544:dy=32
260 @ptr,1,320,45,0
270 @mouse,1,0,320,45
280 :
290 rem ** main logic loop **
300 gosub 370
310 if bp=1 then sound 1,500,4:goto 300
320 if of=1 then 300
330 on (b+1) gosub 660,670,680,690
340 if b<3 then 300
350 @mouse,0:@ptr,0:@text:poke 208,0:end
360 :
370 rem ** universal mouse handler **
380 bp=0
390 do until (bp>127) or bp=1
400 mx=@mouse,2,0:my=@mouse,2,1
410 if mx=zx and my=zy then 470
420 @ptr,1,mx,my,0,8
430 if mx<xo or mx>(xo+dx) then gosub 600:of=1:goto 460
440 if my<yo or my>(yo+dy) then gosub 600:of=1:goto 460
450 gosub 500
460 zx=mx:zy=my
470 bp=joy(1):loop
480 return
490 :
500 rem ** process highlight event **
510 b=int((mx-48)/136)
520 ck=(b*136)+48
530 if b=ob and of=0 then return
540 @windowopen,ck,72,136,32,1:@clear,1,0,10:@windowclose
550 if of=1 then 570
560 @windowopen,(ob*136)+48,72,136,32,1:@clear,1,bc(ob),fc(ob):@windowclose
570 ob=b:of=0
580 return
590 :
600 rem ** turn box off **
610 if of=1 then return
620 @windowopen,(ob*136)+48,72,136,32,1:@clear,1,bc(ob),fc(ob):@windowclose
630 return
640 :
660 play"aabc":return
670 play"cdef":return
680 play"fedc":return
690 play"cccc":return
```

variables are crucial because four of our five events depend about whether the mouse pointer is located in these region. Lines 260-270 turns on the interrupt driven mouse and display a pointer at the same location so they start in synchonization. Please note the listing is for the 1351 mouse if you are using a 1350 or joystick, change the program as follows:

```
260 @ptr,1,320,45,5
270 @mouse,1,1,320,45
470 bp=joy(2):loop
```

Lines 290-350 are the program's main loop. Line 300 begins the main loop by calling the subroutine beginning at line 370. This subroutine does most of the program's work. It constantly monitors and updates the mouse pointer, decides if the box colors need to be changed, and checks for button presses. The variable bp is used to keep track of button press status and is set to zero in line 380 (no button pressed) before the do loop beginning in 390 is started. This loop continues to cycle until a button is pressed (if the left button is pressed bp will become greater than 127, of the right button is pressed then bp will be defined as 1). Lines 400-420 determine whether the mouse has moved or not. If the mouse has not been moved we jump to the end of the loop and check for a button press. Lines 430 and 440 determine whether or not the mouse pointer is on a box or not. If the pointer is not on a box we make sure that the last box the pointer was on is restored to its original color by calling the subroutine beginning at line 600. If the mouse is on a box we proceed to the subroutine in line 500 which colors the box we are in black and purple. Then in line 460 we store the current mouse location so that next time we loop we can see if the mouse was moved.

Once a button is pressed, we return to the main loop where we make the unpleasant noise if the right button has been pressed (line 310). If the left button has been pressed, but the mouse pointer is outside of the box area the we simply start the main loop over again in line 330, however, if the button has been pressed inside of a specific box we branch to play the appropriate music in line 330 and begin again. Due to space constraints I have been purposely vague about the workings of the subroutines at line 500 & 600, however if you study the code and review my BASIC 8 article in issue #19 I think it will become fairly clear.

The expanded keyboard of the C-128 is one of its nicest features, but unfortunately Commodore didn't give us much help with the 128's cursor keys. They are better than the C-64's cursor keys, but the straight-line layout and position at the top of the keyboard are only a slight improvement. Fortunately, the numeric keypad, which in my opinion is drastically under-utilized, provides the solution to this problem. I have re-defined the numeric keypad as cursor controls and other cursor movement features.

Although I have made modifications to use this routine with the Power Assembler and RunScript 128, one of the most useful versions is included here and is for use under the Basic programming environment. In addition, because I do a lot of Basic 8 programming, I wanted the version to fit into area unused by Basic 8, at $0C00. Also due to the constraints of fitting the wedge into the small amount of memory unused by Basic 8, I felt it necessary to keep the wedge under one page of memory, that's 256 bytes, folks! However, the unfortunate result of trying to keep the memory used to a minimum also results in more cryptic code.

HOW IT WORKS.

This wedge re-defines the keypad to cursor keys, as well as a variety of other editing functions. Diagram 1 shows the new definitions of the keypad. I have set up the new cursor keys in the inverted T layout, '1', '2', '3', and '5' becoming left, down, right, and up respectively. Additionally you may move to the beginning or end of a line with '8' or '9'. Keys '7' and '4' have been re-defined as top of screen and bottom of screen respectively. In addition to the movement controls just described, two important editing features have been added. The ' + ' is defined as equivalent to ESC Q, delete to the end of the line. Likewise, the '-' is defined ESC I, insert a line. Of course, because the numeric keypad is very useful for entering data, the pound key (\) is a toggle for the cursor/numeric modes. You will notice when the numeric mode is in effect the cursor is only half the normal height. This is the on-screen indicator of the numeric mode.

THE SOURCE CODE

There are two parts to this wedge. The first part, called MODKEY, is wedged into the KEYCHK vector at $033C. This portion of the code does the work of re-defining the keypad keys. The other part of the wedge called MARKER, toggles the cursor between half and full height and is wedged into the IIRQ vector at $0314. The very first bit of the wedge called INSTALL, rewrites these vectors to point to the routines just described and saves the old vectors for an eventual jump once the wedge has completed its task.

When an interrupt strikes, it does an indirect jump to the vector in KEYCHK, which holds the address of MODKEY after the wedge has been installed. The routine enters with the accumulator loaded with the character value of the keypress, the .y register is loaded with the keyboard matrix code, and the .x register containing shift-key information. These values are first saved, and the wedge determines whether the '\' key was pressed as the toggle. If so, the variable TOGGLE (which is either 1 or 0) is checked, and changed to the other. The wedge then compares the keycode with $45 (which is a keypad '4'). This keypress is defined as Bottom of Screen, for which there is no single character code or escape sequence. The cursor is first moved to the Home position, then moved down 24 times.

The next little routine compares the incoming keycodes for the keypad '8','9',' + ', or '-' which, to implement the aforementioned effects, were escape key sequences. If one of these keys are detected, the escape key is drawn from an indexed table and a JSR to the ROM JESCAPE routine is taken. This routine takes the value of the letter in the accumulator and processes an escape key sequence. If the incoming keypress is not one of the four keys that require an escape key, the wedge branches to KEYEVAL, which also compares the incoming keycodes in an indexed loop. The value is compared to the INKEY table which holds the keyboard matrix code of each of the remaining keypad keys (listed left to right, from the top left). Note that the index value in the .x register also points to the corresponding OUTKEY, which is the character code for the replacement keypress.

If no match is found, the incoming values are restored to the corresponding registers and an indirect jump is made to the address previously in KEYCHK. If the incoming code matches one of the specified keycodes, the corresponding character code is drawn from the OUTKEY table indexed by the .x register. The status of the shift key is restored to the .x register and the indirect jump back to the vector contained in KEYCHK is taken.

As this occurs, the MARKER routine is operating in the background. You should notice that the first eight lines of MARKER was previously described in TC128 (#17). It allows access to the 80 column screen during an interrupt. Then routine then checks the polarity of TOGGLE, and if the numeric keypad mode is in effect, a $03 is written to register 10 (Cursor Start Scan Line) of the VDC chip. If the keypad is in cursor mode, a zero is written to the register. Note that when you run this wedge, the cursor no longer blinks. This is because bits 5 and 6 of register 10 control the cursor mode. I am leaving the implementation of retaining the cursor mode as an exercise for the reader.

ADDING THE WEDGE TO BASIC 8

As I mentioned before, I wanted the wedge to fit into memory not used by Basic 8. This free block is at $0C00. To have the wedge load automatically with Basic 8, add the following line to the loader program of the Basic 8 Editor called BOOT B8:

```
5 BLOAD"BASKEY.O":SYSDEC("0C00")
```

Of course, the wedge works fine programming in Basic 7.0, too!

```
**********************************
*                                *
*    KEYPAD CURSOR CONTROL       *
*       BY JOHN D. CLARK         *
*            4/06/88             *
*                                *
**********************************

*   EQUATES & SYSTEM ROUTINES

IIRQ = $0314
KEYCHK = $033C
BSOUT = $FFD2
VDCADR = $D600
VDCDAT = $D601
JESCAPE = $C01E

*
* KEYPRESSES COMING IN VIA KEYCODES ARE KEYBOARD
* MATRIX CODES...
* OUTGOING KEYS SHOULD BE IN .A AND SHOULD BE
* CHARACTER CODES...
*

 ORG $0C00

INSTALL SEI
 LDA KEYCHK ; SAVE THE KEYCHK VECTOR
 STA KEYRET
 LDA KEYCHK+1
 STA KEYRET+1

 LDA #<MODKEY ; REWRITE THE KEYCHK VECTOR TO
 STA KEYCHK ; POINT TO MODKEY
 LDA #>MODKEY
 STA KEYCHK+1

 LDA IIRQ ; SAVE THE IIRQ VECTOR
 STA RETIRQ
LDA IIRQ+1
 STA RETIRQ+1

 LDA #<MARKER ; AND POINT IT TO MARKER
 STA IIRQ
 LDA #>MARKER
 STA IIRQ+1
 CLI
 RTS

*   MAIN LOOP

MODKEY STX SHIFTKEY ; SAVE INCOMING INFO
 STA CHARKEY
 STY KEYCODES

 CPY #$30 ; IS IT A "\"?
 BNE CONTINUE ; IF NOT, CONTINUE

 LDA TOGGLE ; OTHERWISE TOGGLE TOGGLE...
 EOR #%000000001
 STA TOGGLE
 JMP RETURN

CONTINUE LDA TOGGLE ; IF TOGGLE IS 0 THEN NUMERIC
 BEQ NOMATCH ; MODE, AND PASS BY REDEFINES.

NEWKEY LDA KEYCODES
 CMP #$45 ; IS IT A '4'?
 BNE EXTRAKEY ; IF NOT, GO ON...
 LDA #$13 ; OTHERWISE, PRINT HOME,
 JSR BSOUT
 LDX #24 ; AND 24 CURSOR DOWNS
:LP LDA #$11
 JSR BSOUT
 DEX
 BNE :LP
 LDA #0 ; EXIT WITH NO KEYPRESS IN .A
 JMP RETURN

EXTRAKEY LDX #ESCLETT-ESCKEY-1 ; NUMBER OF ESCAPE KEYS
:LP CMP ESCKEY,X ; COMPARE TO CHOSEN KEYS ON
 BEQ :CT ; THE KEYPAD.
 DEX
 BPL :LP
 JMP KEYEVAL ; IF NO MATCH, MOVE ON.
:CT LDA ESCLETT,X ; OTHERWISE, GET THE ESC LETTER
 JSR JESCAPE ; AND GOTO ROM ESCAPE ROUTINE.

 LDA #0 ; CLEAR OUT KEYPRESS,
 TAX
 JMP RETURN ; AND EXIT WEDGE.

KEYEVAL LDX #OUTKEY-INKEY-1 ; NUMBER OF KEYPAD KEYS.
:LP CMP INKEY,X ; COMPARE WITH MATRIX CODES OF
 BEQ :CT ; KEYPAD KEYS.
 DEX
 BPL :LP
 JMP NOMATCH
```

I would venture to guess that most TC-128 readers do not concern themselves overly with other types of computers: they have found a computer that matches their abilities and needs with economy and style. The software that exists for this machine is head and shoulders above the competition (regardless of the number of bits they handle) in quality, price, and features. So why do software manufacturers continue to produce quality software that aids transfer of data between computers? Well, there are times when even us C-128 zealots have to admit that there are other computers out there and occasionally it is worthwhile to share data with the people using those "other" machines.

S.O.G.W.A.P. Software has re-released a program with improvements that allow C-128 owners with the 1571 drive to acquire data quickly and easily from other disk formats, specifically CP/M and PC-Dos formatted disks. This is a boon to those of us that have to transfer data daily between computers, and hate the inconvenience or lack the hardware of a modem transfer.

As most of you know already, Commodore 5.25" disks use a different recording technique than most other computers. This, in the past, has created problems for users with access to more than one computer and need of replicating data on more than one system. The standard method of transferring data in the past has been to have a modem on both machines (or a null-modem cable) and transfer data using a telecommunications program. This introduced users to more problems, since Commodore 8-bit computers use CBM ASCII, a different version of the code that most European and American computers use, ASCII (American Standard Code for Information Interchange). So when a file is written on, say an IBM PC, is transferred to a Commodore word processor, most of the characters are garbled and have to be converted to CBM ASCII (also referred to as PetASCII). Many of the best telecommunications packages knew of this trouble, and would do the conversion during the reception/transmission of the data, and all would be well, but this was a lesson that took a long time to learn.

Along come the C-128 and the 1571 drive, the flower of 8-bit technology. The computer handles all of the data in a straight-forward and quick way, and the drive is capable of receiving and making sense of dozens of disk formats, including CP/M and PC-DOS formats. It didn't take long for S.O.G.W.A.P. to take advantage of this quirk (and good fate) and design a program that allows C-128 users to read AND write data with all of these various disk formats. This was one and half years ago, and now they have a new version out, with features that makes it quite worth a look.

The Big Blue Reader 128 (BBR) will read a disk that was formatted as a PC-Dos 8- or 9-track disk, most of the CP/M formats that are still around, and the 1571's double sided disks. It will read data, transfer it to the 128's memory, and hold it there until told to write it to a new disk. In this process of writing, it writes to all of the named disks, and will do the conversion from ASCII to CBM ASCII, add or strip linefeeds (most pure-ASCII systems have this annoying habit of adding linefeeds to everything, which is not needed on the Commodore 8-bit computers), and even print the data stored to screen or printer (serial printers only, so you RS-232 types out there, like me, are out of luck). This last feature is such a plus that I can hardly hold my enthusiasm down: you cannot know how many times I have been sorting disks and had to load a word processor to see what is REALLY on a disk. This speeds things up immeasurably.

The features that set this apart from the previous versions of BBR are: RAM disk support (not the standard CBM-released RAM disk software, but another version, but allows you to transfer data files the size of 550K, almost the size of 2 1571 disks); 1581 support, which I will mention in a moment; and complete BURST mode speeds in reads and writes to all types of disks. As a check, I formatted a PC disk on a PC-XT clone computer (4.77 Mhz, the base unit), and it took an even 1 minute to format this 360K disk. To format a disk using BBR it took 50.4 seconds. The disks I formatted on the 1571 worked well when taken to the PC-XT, and likewise the other way. This may keep me at home formatting disks for friends, rather than the slow PC speeds!

In 128 mode, it is possible to defeat the built-in checksum correction when writing to a disk, and speed it up. This feature is worthwhile if doing many disks, but I have discovered that the amount of time saved is slight: only a .6 second improvement in formatting (43.3 vs. 42.7) and time saved while writing data to disk is likewise inconsequential. If only working on a couple of disks, it might be worthwhile to remain in the "slow" BURST mode of the 1571. No such verify-repressing mode is possible in CP/M or PC-DOS mode.

For 1581 owners, the (finally) popular 3.5" disks PC and CP/M 3.5" disks are readable and writeable. The RAM limits of even a 1750 (512K) REU prevent full-disk copying (or rather, since only file copying is possible, a file as large as an entire 720K disk) is not possible, but it will handle single files of any size up to the limit of available RAM. Unlike the 5.25" counterparts, it is impossible with this version of BBR to format 3.5" disks (you can use the Commodore shell to format CBM 3.5" disks within the program), which I find a drawback, but an acceptable nuisance. Except for those 'lucky' few that have spent the thouand on PS/2 systems, nearly all PC users have 5.25" disk drives, so you can save data on that type, and transfer files using DOS.

Big Blue Reader 128 is certainly a wonderful utility for folks that bring text and other 'sequential' files home to do work at home. This includes word processing and spreadsheet files, which can be troublesome, but are possible to translate between machines. I have discovered

For many congregations, MS-DOS seems like the only option available because of limited software for church use. Smaller 8-bit computers are often not considered because few programs exist to assist in membership and contribution management. Also many smaller congregations cannot yet justify the cost of software and the MS-DOS system and therefore do not computerize because of their size. But options are becoming available for the more inexpensive 8-bit systems!

Church Management Database is a five disk software application designed for the Commodore 128 system. Using Precision Software's Superbase 128 Database program, it very adequately helps meet the needs of the smaller membership congregation. With 1571 (340 Kb) drive, it is adequate for parishes of less than 450 members. And with the addition of a second and larger capacity disk drive (1581), it should adequately handle parishes of over 1000.

It is totally menu operated. The church secretary or pastor chooses a task to be performed from a displayed menu. Yet the beauty of the system is that persons with programming skills can add custom applications as need arises. A 110 page step by step instruction manual is included and there are over 35 help screens available on line. It has been designed for use by churches which normally distinguish between baptism and voting membership (such as Lutheran, Episcopal, Roman Catholic, Methodist, Presbyterian and Reformed churches).

The system consists of two modules. Module one (Member Management) has a 36 option menu - can automatically add, transfer, delete members and prospective members, update records, search via specific criteria, produce over 45 reports, labels and listings. Some reports provide staff viewing only to restrict general public view of confidential material.

The second module (Contributions Management) has a 22 option main menu. It can record weekly contributions (set up from the member module or from scratch), update entries, search and print contributions. Three major giving categories and 23 special categories make it adaptable to any parish contribution plan. Full analysis of contributions and of giving patterns are available. Reports on weekly totals, pledges and totals and much much more is included. Statements can be printed at any time of the giving year for one, a selected member, or all contributions.

In testing the system I found the data entry to be a very impressive - data is checked by the system and errors caught are explained to the operator. Coded lists are extensive and yet readily available at the touch of the help key. The available lists should satisfy any parish need but the Superbase system also enables the operator to generate any additionally needed lists. The variety of giving categories ought to fit any size parish need. There is a coded entry protection so that unwanted viewers cannot access the system without the proper code sequence. The statistical analysis is very impressive and meets the demands of most denominational report needs. And they system supports many different printers. The extra utility disk provides assistance in maintaining such things as scouting and other non-parish groups.

There are some limitations. There is no word processor with this system, although it has been tested with PaperClip and is able to produce information needed for mail merge needs (i.e., letter greetings inserted into a form letter). The family "card" permits only four children to a family, although the programmer has made provisions for getting around this. Communion entry might be made easier by a sequential run through the names. Contribution entry currently does not include entry by name. Most limitations are minor and only of personal preference. The system has had three years of parish testing and upgrades to the system are available as they are designed.

Cost - individually - Membership $150, Contributions $75 (package $179). Also required is the Superbase 128 database system ($55). There is a 30 day refund policy.

Big Blue Reader Review Continued...

hand. If they use Basic 8 a lot, or are programmers, they install the few ROM chips that are available for those applications. It might be worth a look by S.O.G.W.A.P. to make ROM chips with BBR available. It is certainly a fast-booting program (less than 30 seconds, and not copy protected), but if I could just type in a SYS command and have the BBR pop up, I might be more willing to bring work home. After all, after transferring the data, I still have to boot up my application to do the actual work in.

Other improvements I would suggest for the next version would include 1581 formatting of CP/M and PC disks, and even Macintosh, ST and Amiga disks. There are several utilities for transferring Commodore data to Amiga disks, but none (to my knowledge) for going the other way. But I don't want to leave you with the impression I am dissatisfied with this utility: it does all it says it will, and with a minimum of hassle and headache.

Big Blue Reader 128/64. Michael R. Miller. Published by S.O.G.W.A.P. Software; 115 Bellmont Road; Decatur, IN 46733 (219) 724-3900. $45 (upgrade $18 + original disk). Version 3.0d.

Many of us have grown along with Commodore computers, starting with the affordable VIC 20 and C-64 then graduating to the C-128. Along the way the BASIC language has grown. Now, with the C-128 and BASIC 7.0, we have one of the most powerful BASIC languages available to an eight bit computer. With more commands then ever before, many of which we may not use regularly, one can easily forget the proper syntax for some of these commands. C-128 HELPER is a program for the BASIC programmer that wants to get information about any of these BASIC keywords, right from the C-128, without having page through documentation for help. One of the first drawbacks of most helper programs of this genre is that it usually takes control of the computer. This means that you have to re-load the helper program and exit when you found your answer. When you're done, you have to restart the application or program you were working on. Not so with C-128 HELPER, to access the help menu you simply press the 'HELP' key, and get the desired information you want. When you are done, you are returned to the place you left, with the screen and all intact. No more, no less.

C-128 HELPER supports the C-128 or C-128D, a 1541 or 1571 drive and requires an 80 column screen. The program is non-copy protected so that back-up copying is no problem and even suggested.

When loaded into the C-128 the program is rather invisible, as it is waiting for you to access it by typing the HELP key. At this point the program acts much like the help menus in the popular Pocket Series, in that it is there but not in control until needed. After the help key is pressed, control is passed to the C-128 HELPER machine language program, your current screen is saved, and C-128 Helper is ready to give you a hand. The first thing you are greeted with is a menu of all the BASIC commands and a prompt to enter the keyword you want information about. Enter the keyword of your choice from the menu screen, and the program will retrieve the requested information from the program disk about that keyword. Nothing too impressive you may be thinking to yourself, but not only is the requested keyword and definition on the screen, but a brief program example of the proper use of the keyword and a cross reference to other associated basic keywords. Another nice feature is that the C-128 Helper lists the page number in the C-128 System Guide where the BASIC keyword is documented so you can easily find more information if needed. Besides the BASIC keywords being available, the complete set of screen poke codes, ASCII codes, BASIC token, BASIC abbreviations, commonly used memory locations, are also available via this handy on screen reference manual.

Another nice feature is that while in the program, any screen that can be seen can also be printed. But surprisingly, the C-128 HELPER can do more. The program comes with a 41 page manual which is put together well and offers much in the way of information. At no point did I get the feeling that it takes for granted that I know how to use the program, which is a common problem with many software manuals I've worked with.

Now let us say that you want the program to access Drive #9 instead of having to have the program disk in Drive #8, or you want to add some special notes to any of the existing screens. C-128 HELPER is totally modifiable and the instruction book includes 5 pages on customizing the program. As a matter of fact it even includes some suggestions about developing your own Menu and Data Disk, and interfacing it with your own programs, including the utilities to make new screens. Side two of the program disk includes "Menu Maker", a utility to edit and save a menu screen. Some suggested uses would be to make a Phone Directory, Appointment Calendar, or Notebook. Using the Menu Maker program to list the different records that you have on the disk, you then would have one full screen full of information available for each listing on the MENU screen.

Once you have constructed a menu screen you can then build the actual data screens, which are then saved to the disk. The program does its own compression or "packing" to save disk space, so the 80 column menu screen always takes up less than the expected 16 disk blocks.

Overall I would have to say that C-128 HELPER is more than I expected when I first heard about it, and there has evidently been a great deal of fore-thought in the development of the program. The manual has all of the needed information in a very readable form. Much more information is available in the manual than just your basic "How To" and it actually delves into the "What Makes This Thing Tick" aspect of C-128 Helper. The manual alone rates an A+ in my book. On the other hand, some things that could have been supported, but are not, would have been the 1351 Mouse as an input device, when requesting a topic, and support of the Ram Expansion would be a nice feature. Putting C-128 HELPER in your software library won't have the same effect as your favorite word processor or game, but anyone who does any programming will certainly find this program useful, and may even consider it inspirational, especially those who are just learning BASIC 7.0. I know I've cooked up a few uses for the customized applications that can be made.

Basic 7.0's rich command set combined with the C-128's fantastic screen editor make things that are difficult and tedious on other computers a snap. But every once in a while I run into a situation where I could really use some feature that is not implemented on the C-128 that I have encountered in other environments. Little things, such as the *Type* command found on the IBM and Amiga computers, or a *Merge* command, or the fantastic *Find* and *Change* commands found in COMAL and the old Vic-20 Programmer's Aid Cartridge. But alas, we all must learn to live without, right? Well, I used to think so, until Free Spirit gave me a call to tell me about their "Super Chips" for the C-128.

The Super Chips consist of three EPROMS which replace the three kernal ROMs in your C-128 (if you own the C-128D, Free Spirit also offers a two EPROM set to replace the C-128Ds two kernal ROMs) which increase the "vocabulary" of your C-128. "The Super Chips" should not be confused with the mediocre "Super Chip" marketed by Utilities Unlimited for the C-128's empty internal expansion socket. In the future I'll pan the singular, but for now I'd rather gloat about the plural! The Super Chips are easy to install. Simply remove the ROMs currently occupying sockets U33, U34, and U35, and replace them with the appropriately marked Super Chips. Of course, you have be wary of static electric charges and careful not to break the pins on the EPROMs upon insertion. (I know, I know, you just bought the upgrade kernal ROMs for your C-128, and now I am telling you about these beauties. If it is any consolation, the Super Chips do include the updated code so you will not lose the fixes that Fred Bowen worked so hard on and that you paid for.)

Once you have installed the Super Chips they are virtually transparent. The most visible evidence that your C-128 has been altered can be observed at power up. The Super Chips change the default colors from the traditional cyan on black on the 80 column screen and the 40 column default colors of green on grey to green on black on both screens. Why the Super Chip's programmer's chose to change the default colors is a mystery to me. They are neither more or less readable than the traditional C-128 colors. In addition a "Super Chip" copyright message is added to those already at the top of the screen. Another relatively minor, but more useful change is that the Super Chips automatically place the C-128 into 2.0 megahertz (fast) mode upon power up if the 40/80 DISPLAY key is in the down position. The last overt change to the C-128's operating system involves the function keys. The authors of the Super Chip have taken it upon themselves to redefine them for you. Although I really love the Super Chips, this really irks me. Not because their definitions are bad choices, in fact, some of them are quite thoughtful, but after 2.5 years of C-128 usage I am hardly enamored with the prospect of learning new function key definitions. To keep myself from using too words I don't want my 3 year old son (parrot) to repeat, I have created my own function key definitions and saved them to disk, but I fear it is going to be the source of many "naughty words" from many users.

The command I probably use the most often is the *Type* command. This command reads disk files byte by byte and displays them to the screen. I can't emphasize enough how useful this command is, because it allows me to quickly read or examine sequential text files without having to load a word processor or dump a program I am working on in memory. In addition to being useful for reading sequential text files, the *Type* command also is quite handy for examining other filetypes as well. For instance, I have recently found the *Type* command quite handy for examining the headers of BASIC 8 picture files. The *Type* command also recognizes BASIC program files and automatically de-tokenizes the file, displaying the BASIC listing perfectly to the screen. This option is very handy when you want to examine routines in one program while working on another.

Although I find myself using the *Type* command more frequently, the two commands that really make the Super Chips worth every penny are the: *Find* and *Change* commands. I can honestly say that after using these commands for just a day, you will begin to wonder how you lived without them. The *Find* command displays every occurrence of a specified string of characters, while the *Change* command allows you to replace a specified string of characters with another specific string of characters. Both of these commands work fast and flawlessly and are invaluable when you need to locate references to a certain item or make global changes for things like variable names or subroutine calls etc. Both commands allow you to specify a subsidiary range of line numbers on which the operations can be performed. The only glitch I discovered with the *Find* and *Change* command is that you can not use REM as a part of your specified text for searching as these commands are processed like BASIC commands and the REM statement renders them inactive.

I find the *Merge* and *Combine* commands extremely helpful, both of which allow you to append BASIC code from disk to a program in memory. These commands make it easy to build up a subroutine library to avoid "re-inventing the wheel".

Beyond the Super Chip commands I have already detailed, here is a brief summary the others: *Unnew* - Resurrects programs recently erased with the NEW command, *Start* - Displays the starting address for a program file on disk in both decimal and hexadecimal, *File* - Returns the start address, ending address, and length in bytes of a program file on disk, and the * command which sends MONITOR commands to a printer connected as device 4 via the serial bus. Also included is a text screen dump that works on both the 40 and 80 column screens and an 80 column screen editor which saves screens to disk or create print statements for BASIC programs.

The Super Chips are a wonderful addition for BASIC programmers. They are completely compatible with all commercial software (yes, really!) and work well with most BASIC 7.0 extensions such as BASIC 8. I highly recommend them. (The Super Chips, $49.95, from Free Spirit Software, 905 West Hillgrove #6, La Grange IL 60525, 312-352-7323).

```
:CT LDA OUTKEY,X ; IF FOUND, GET NEW KEYPRESS.
 LDX SHIFTKEY
 JMP RETURN

NOMATCH LDA CHARKEY
 LDX SHIFTKEY
 LDY KEYCODES

RETURN JMP (KEYRET)

*       VARIABLES

KEYRET DFB 0,0
RETIRQ DFB 0,0
SHIFTKEY DFB 0
KEYCODES DFB 0
CHARKEY DFB 0
TOGGLE DFB 1

INKEY = *
 HEX 46
 HEX 45,42,4D
 HEX 47,44,4F,4C
 HEX 51,52

OUTKEY = *
 HEX 13
 HEX 00,91,00
 HEX 9D,11,1D,0D
 HEX 00,00

ESCKEY = *  ; ESCAPES FOR 9,8,+,-
 HEX 4E,41,49,4A

ESCLETT = *
 TXT 'KJQI'

MARKER TSX  ; CAN WE ACCESS THE VDC CHIP?
 LDA $107,X
 CMP #$C2
 BNE EXIT
 LDA $106,X
 AND #$F0
 CMP #$60
 BNE EXIT

 LDA TOGGLE ; IF SO, CHECK TOGGLE
 BNE REGCUR ; IF NUMERIC MODE,
 LDA #3 ; MAKE HALF CURSOR.
 JSR VDCWRITE
 JMP (RETIRQ)

REGCUR LDA #0 ; IF CURSOR MODE,
 JSR VDCWRITE ; MAKE WHOLE CURSOR.
EXIT JMP (RETIRQ)

VDCWRITE LDX #$0A ; WRITE TO REGISTER 10 OF THE
 STX VDCADR ; VDC CHIP.
:LP LDX VDCADR
 BPL :LP
 STA VDCDAT
 RTS
```

Coming Soon!

TWIN CITIES 128

COMPENDIUM BOOK #1

The best of issues # 1 - # 18

Subscribers will be contacted by mail with ordering information!

Look for announcements on Q-link and GEnie!

The other day I was attempting to perform some tedious formatting task on Pocket Writer 2. About halfway through the third document, my husband (who knows everything) was looking over my shoulder, and said to me "Why aren't you doing this a better way?" and proceeded to show me a fast, easy solution 100 times superior to the method I was using. "Where did you learn that?" I asked, amazed. "Oh, I found it flipping through the manual one day," he smirked at me.

Now, I'm not the type of person who ignores her manual entirely. In fact, if I'm ever having a problem with some software, the manual is always the first place that I look. However, I don't usually read them for fun, like my husband does. Which means that once I've read enough to get me up and running, assuming I don't have any problems, I don't spend a lot of time perusing the manual. This also means that I don't always pick up the best ways to use the software to its fullest. Since I'm betting that there are a lot of folks like me, (and not like Loren), I thought I'd let you in on some tricks I learned about Pocket Writer 2 after really digging into the manual. Hopefully, this will save you from having to do the same. We'll call this: Mrs. Sparrow James' Slick Pocket Writer 2 Tricks.

Cursor Movement:

Now, I'm sure that most of you know that to get to the top of your document, you can just hit HOME twice, and to the bottom of your document, press F2. And of course, who could forget Control and G for "going to" a specific page. But did you know that if you want to use more detailed cursor movement (short of just moving your cursor) F3 and F4 will move you forward and backward one word, respectively. To move to the last character in a line, Control and Cursor Right will take you there. And guess what, Control and Cursor Left will take you to first character in a line.

Now you may be thinking, so what? Big deal. Moving the cursor isn't that exciting. But just think. If you used these often enough that they were second nature, think how easy it would be to move around your big documents. Moving forward by word could really help your proof reading by forcing you to look at each word individually. And how often have you been at the middle of a line and wanted to be at the end, so you just cursored over until you got there. Think how much more efficient it would be to just press Control and Cursor Right, instead of all of those Cursor Rights. For browsing through a document, Control and Cursor Up and Cursor Down will take you up and down by screen.

Editing features:

I suspect that most Pocket Writer 2 users are aware that to delete the line a cursor is on, just type Control and the Delete key. To insert a line, the command is Control and the Insert Key (Shift Delete). But something I wasn't aware of until yesterday is that Control and X will delete text from the cursor to the end of the line. To delete text from the beginning of the line to the cursor, type Control and Y.

When I want to delete a section of text, I used to paint the range using Control r, and then press Control and D. However, a more painless method to use is Control d before you paint the range. The program then prompts you "Letter, Word, Sentence or Paragraph?" Then just type the first letter of your choice (l, w, s, p). Although I see little use for this method when deleting a single letter, and its use may be limited even for a word or sentence, this may be a very efficient method of deleting a paragraph, because you don't have to do all of the cursor movement necessary to paint the range first. However, this method differs from range delete in that for this method, a paragraph is not saved in the range memory in case you change your mind, whereas if you use range delete, it will be saved in memory (until you do another range command). However, if you use the Control d method for a word or sentence, it does save those in memory (don't ask me, I don't understand the logic either. It seems to me you'd be able to remember a single word a lot better than an entire paragraph).

By the way, something I was reminded of the other day: when copying a range, remember that once you paint the range, it is in memory until you clear the range memory. The importance of this is apparent when you are making multiple copies of a range. I always forget this fact, and make one copy, then clear the range, paint it again, copy it again, etc. instead of simply painting the range once, and using Control c as many times as I need. (Pretty obvious, huh?) Yet, I know that other users aren't aware/forget this simple trick.

Something I find very useful when creating documents with Pocket Writer 2 is the ability to change the case of text, especially to all upper case. Truth be told, I've never gotten used to using the CAPS Lock key on the C-128, and half the time I change my mind of what should be capitalized for my titles and subtitles, etc., so this feature can really come in handy. Changing the case of some of the text in your document requires painting the range with the range commands, then Control, Shift and + for all upper case, and Control Shift - for all lower case.

Screen features:

These are the kind of settings that I don't pay a lot of attention to when I first start learning a new piece of software, and then I get to lazy to check out later on. After all, so long as I can see the menus, etc. on my monitor, who cares if the colors are awful, I hate my cursor type, and the obnoxious error bell is on about a million decibels too loud. However, for those of you who haven't yet taken the time to correct some of those annoyances and make permanent adjustments to your configure file, here are some of the more important changes you can make on the screen right now:

1. To turn off the awful error bell, type Control Shift B. It works as a toggle switch, so if you want to turn it back on again, simply repeat the command.

2. To change the cursor from a flashing block, type Control and *, which will make it a solid block. Repeat the command to make it a solid underline, and once more if you decide you liked the flashing block after all (you do, really? Yuck.).

3. To change you various colors of text, the command line, background, etc. type Control and one of the function keys (F1 through F8). Although I won't take up the space here to go through each individually, you should take a few minutes some day to experiment with several combinations until you find the one that works best for you. If you make an effort to find the "perfect" combination, make sure to adjust your configure file so that you don't have to do it again. In fact, the configure file is the best place to work with this anyway, since there are a few color settings which can be made only from the file.

Special Tricks:

If you ever need to see how many words your document contains, just type Control Shift w.

To see the amount of memory left in the computer, type Control Shift m. The manual recommends that you probably ought to save this file and open a new one if your bytes free is 1,000 or less.

To find all of the return marks in your document, press Commodore key, f, and then the back arrow (the key used for tabs located at the upper left of the keyboard) and return. If you are just looking for the return marks, and do not intend on replacing them, just press return again at the "Replace?" prompt. However, if you wish to delete the return marks, enter the same back arrow at the prompt and press enter. I find this feature particularly useful after I have converted a file from a text file to a sequential file in Pocket Writer 2 and back again to text. Converting the file to sequential adds return marks at the end of every line, and if I change it back to a text file, the return marks stay there. Although this process can be performed at a faster pace with some other software, some do not give you the option of skipping over return marks that you want, which means that you have to edit your text again later, adding the return marks. In addition, other programs often replace the return mark with an extra space, which can make your text look ugly.

Commodore, Shift and D will replace the text you are working on with a file that contains a copy of all of your current directories. At Twin Cities 128, we use this for printing directories of our disks (maybe moving file names around in our list so that they make more sense) and also for incorporating several file names into articles and letters/EMail to readers. I find this a very useful tool, but just remember, you will lose whatever you have in memory at the time, so save your document first.

Another little known fact is that Pocket Writer 2 will read a file created by GeoWrite. It must be called "text scrap" and be a "USR" file. To insert it in the current file, type Control, Shift and r.

If you are tired of the HELP menu, and you don't have you no longer use it at all, you can set up your configure file to automatically "dump" the Help area from Pocket Writer each time you boot up. Although this does not remove the help information from disk, it does dump it from memory, and gives you about 5K extra to work with for text memory. Simply go to the Configure file, and Type "On" where it says "Dump Help=". If you decide later that you want it back again, you must change it to "Off" in the Configure file. Or you can dump the help in the current file by typing Control Shift and H. If you want to turn off the help menu, but do not want to dump it from memory, type Control and h.

I hope that all of you loyal Pocket Writer fans know that to load a file from disk, simply get a directory, then place your cursor on the first letter of the file name and press Enter. The file will be loaded immediately.

There we are, just a few tricks to keep you busy. Keep in mind that the more you use the various features of any piece of software, the more likely you are to get the full potential from the program. Unfortunately, that may mean studying the manual a bit more, but the time saved in improved software usage will probably make it up several times over.

Free Spirit Software Inc.

UTILITIES

ULTRA DOS UTILITIES
Module I

High Speed Hard Drive or dual floppy drive backup utility for the Amiga 500, 1000 or 2000. 512K Amiga required. Compatible with any hard drive that follows conventional AmigaDOS protocol. Backup those valuable files on your Hard Disk the easy way for only **$59.95!**

OXFORD PASCAL 128

OXFORD PASCAL 128 is an implementation of standard Pascal designed specifically for the C128. It offers all the enhancements of this powerful language together with some useful enhancements for the C128. Only **$39.95!**

SUPER AIDE

All-purpose utility program for the C64 provides:

- Bi-directional scrolling
- Auto Line Deletion
- Trace function
- Disassembler
- Lo-Res Screen Dump
- Number conversion (10, hex, binary)
- Append files
- Format — short new/complete new
- Menu-driven
- Auto Line Numbering
- Renumber
- ML Monitor
- List all variables to screen
- Hi-Res Screen Dump
- Restore newed Basic program
- Change Device number
- Packed Line Editor
- Determine file load address
- Change THIS TO THAT — search for all instances of specified string and replace with second specified string
- *And much, much more!*

Super Aide, the complete programmer's tool kit. Only **$29.95!**

" . . . excellent, efficient program that can help you save both money and downtime."

Compute!'s Gazette
Dec., 1987

1541/1571 DRIVE ALIGNMENT

1541/1571 Drive Alignment reports the alignment condition of the disk drive as you perform adjustments. On screen help is available while the program is running. Includes features for speed adjustment. Complete instruction manual on aligning both 1541 and 1571 drives. Even includes instructions on how to load alignment program when nothing else will load! Works on the C64, SX64, C128 in either 64 or 128 mode, 1541, 1571 in either 1541 or 1571 mode! Autoboots to all modes. Second drive fully supported. Program disk, calibration disk and instruction manual only **$34.95!**

SUPER 81 UTILITIES

Super 81 Utilities is a complete utilities package for the 1581 disk drive and C128 computer. Among the many Super 81 Utilities features are:

- Copy whole disks from 1541 or 1571 format to 1581 partitions.
- Copy 1541 or 1571 files to 1581 disks
- Backup 1581 disks or files with 1 or 2 1581's
- Supplied on both 3½" and 5¼" diskettes so that it will load on either the 1571 or 1581 drive.
- Perform many CP/M and MS-DOS utility functions
- Perform numerous DOS functions such as rename a disk, rename a file, scratch or unscratch files, lock or unlock files, create auto-boot and much more!

Super 81 Utilities uses an option window to display all choices available at any given time. A full featured disk utilities system for the 1581 for only **$39.95!**

RAMDOS
Lightning Fast
RAM-DISK

RAMDOS is a complete RAM based "Disk" Operating System for the Commodore 1700 and 1750 RAM expansion modules which turns all or part of the expansion memory into a lightning fast RAM-DISK. RAMDOS behaves similar to a much faster 1541 or 1571 floppy disk except that the data is held in expansion RAM and not on disk. Under RAMDOS, a 50K program can be loaded in ½ second. Programs and files can be transferred to and from disk with a single command. RAMDOS is available for only **$39.95!**

GAMES

MONSTER POWER

BIG WHEEL MONSTER arcade action for the C64! One to four players can compete in Tractor Pulls, Mud Bogs and Monster Trucks.

Monster Power is only **$14.95!**

STRATEGIC PLAYGROUND FOOTBALL

Enjoy a nice game of football on the C64. Using playground rules, one or two players can compete. Get a treat from the icecream truck at half time.

Strategic Playground Football is only **$9.95!**

SUPER BIKE

Action-packed, fun-filled motor cycle arcade game for the C64. Race the clock in Motocross, Enduro, Supercross or Trials. Fly through the air on spectacular jumps. Bounce over woop-de-doos.

Avoid logs, trees, water holes, brick walls, other bikers, etc. as you vie for the gold cup.

Thrilling **Super Bike** action for only **$14.95!**

GALACTIC FRONTIER

Exciting space exploration game for the C64. Search for life forms among the 200 billion stars in our galaxy. Scientifically accurate. Awesome graphics! For the serious student of astronomy or the casual explorer who wants to boldly go where no man has gone before.

Only **$29.95!**

Order with check, money order, VISA, MasterCard, COD. Free shipping & handling on US, Canadian, APO,FPO orders. COD & Foreign orders add $4.00

MasterCard VISA

Order From: **Free Spirit Software, Inc.**
905 W. Hillgrove, Suite 6
LaGrange, IL 60525
(312) 352-7323
1-800-552-6777
For Technical Assistance call: (312)352-7335

SELECTED PRODUCTS AVAILABLE FROM FINANCIAL SYSTEMS SOFTWARE LTD IN ENGLAND 0905-611-463