PARSEC INC
POB 111
SALEM MA 01970-0111   USA

FORWARD & ADDRESS CORRECTION

BULK RATE
U.S. POSTAGE
PAID
SALEM, MA
PERMIT NO. 188

CMD Introduces ...

perfect print LQ *for* GEOS™

GEOS users: Now get near laser quality output from your dot matrix printer

Eliminates jagged output • Resolution up to 360 x 360 DPI (24 Pin), 240 x 216 DPI (9 Pin)
GEOCABLE compatible • Allows multiple copy printing • HQ drivers enhance graphic output

Perfect Print LQ is a complete print enhancement package for GEOS that delivers the highest quality dot matrix output possible. Includes a unique print utility and font set for enhancing GEOWRITE documents, utilities for creating fonts, and high quality drivers for other GEOS applications. Improves text and graphic output on virtually all 9 & 24 pin dot matrix printers and supports font attributes such as italics, outline, underline, bold, etc.

Main system (All drivers, utilities, and 7 fonts) **$34.95** • Font Package (45 LQ fonts) **$29.95**
Complete System (Main System & 45 fonts) **$49.95** • Shipping/Handling **$5.00** Canada add **$4.50**

RAMLink™

Power Backed REU Interface and Expandable RAM Disk

- **Non-Volatile Storage** - Operates on its own external power supply. Optional rechargeable battery back-up retains data even during power outages.
- **Compatible** - Use GEOS, CP/M, Q-Link, BBS programs, productivity software and more. RAM port for connection of REU or GEORAM of any capacity. Pass-thru port supports most cartridges.
- **User Expandable** - Internal RAMCard allows expansion up to 16 Mb by using standard SIMM's.
- **FAST** - Up to 400x faster than a 1541; 20x faster than RAMDOS; Built-in JiffyDOS plus parallel interface to speed access to CMD Hard Drives.
- **Easy to Use** - Plugs into the Cartridge Port. Operates like a standard disk drive. File and disk copiers included along with partitioning and GEOS support utilities.
- **RL-DOS** - Organize RAM into manageable partitions that emulate 1541, 1571, & 1581 drives or expand to the full RAM capacity with MS-DOS style subdirectories.

RAMLink (no RAMCard)	$179.95	RL (with RAMCard)	$219.95
Battery back-up unit	$24.95	1Mb SIMM	$50.00
HD Parallel Cable	$14.95	4Mb SIMM	$185.00

RAMDrive™

High-Speed Self-Contained Battery-Backed RAM Cartridge

- **Compact** - All the features you've ever wanted for RAM expansion in a small, transportable unit. Three capacities: 512K, 1 Mb and 2 Mb. Dimensions: 6"l x 3"w x 1"h.
- **Compatible** - Use RAMDrive with GEOS, CP/M, Q-Link, BBS programs, productivity software and more.
- **Non-Volatile** - External power supply eliminates drain on computer power supply and retains data indefinitely. Internal rechargeable batteries retain data up to 7 days.
- **FAST** - Up to 400x faster than a 1541; 20x faster than RAMDOS; Built-in JiffyDOS speeds access to CMD Hard Drives and JiffyDOS-equipped floppy drives.
- **Easy to Use** - Plugs into the cartridge port. Operates like a standard disk drive. File and disk copiers included along with partitioning and GEOS support utilities.

RD-DOS - Organize RAMDrive into as many as 30 manageable partitions that can emulate 1541, 1571, and 1581 disk drives or expand to the full RAM capacity with MS-DOS style subdirectories. Autoboot capability for both 64 and 128 mode programs. Configurable as any device number from 8 thru 30.

512K **$199.95** • 1 Megabyte **$249.95** • 2 Megabyte **$299.95**

HD Series Hard Drives

Special Limited-Time Price Savings

The Ultimate in Mass Storage for the 64/128

- **Capacity** - 20 Mb to 200 Mb capacities enable you to store the equivalent of up to 1250 1541 (170K) disks.
- **Speed** - The fastest Commodore compatible hard drives. Speeds up to 50x faster than a 1541.
- **Compatibility** - Ideal for use with GEOS, CP/M, Q'Link, BBS programs, productivity software and much more.
- **Compact Size** - 3 1/2" SCSI technology allows for a compact case about the same size as a 1581.
- **Expandability** - Chain up to six SCSI devices or connect to Macintosh, IBM-Compatible & Amiga computers.
- **Built-in Real Time Clock** - Automatically time and date stamps files and sets the GEOS clock.
- **HD-DOS** - Organize storage into as many as 254 partitions that emulate 1541, 1571, & 1581 drives or expand to 16Mb with MS-DOS style subdirectories.

HD-20	~~499.95~~	$399.95
HD-40	~~659.95~~	$549.95
HD-100	~~999.95~~	$849.95
HD-200	~~1299.95~~	$1099.95

JiffyDOS™ V 6.0

The Ultimate Disk Drive Speed Enhancement System

Hi-performance ROM upgrade • Performs all disk accesses up to 15 times faster
Guaranteed 100% compatible • Built-in DOS Wedge & file copier • Easy to install

JiffyDOS 64 or SX-64 **$59.95** • JiffyDOS 128 or 128D **$69.95** • Add'l Drive ROMs **$29.95**
System includes computer Kernal ROM(s) & one drive ROM. Please specify computer & drive model with serial#.

gateWay™

An Efficient, Powerful & Versatile New Desktop for GEOS 2.0

Streamlined File Management • Task Switching • Three drive support
Access full capacity of HD, RAMLink, RAMDrive, expanded REU's and GEORAM

gateWay 64 **$29.95** • gateWay 128 **$29.95** • Both **$44.95** • Shipping: **$5.00**
Note: The gateWay requires a Commodore 64 or 128, a mouse or joystick and GEOS 2.0 for operation.

*Swift*Link-232 Serial Cartridge

A real RS-232 port for Commodore 64 and 128 Computers

Provides an industry-standard IBM AT-style serial port that communicates at speeds from 300 to 38,400 baud and provides reliable 1200 and 2400 bps using Hayes-compatible modems. Includes 64 & 128 terminal programs as well as software for transferring files to other computers. (SWIFTLink cable required for modem connection)

*SWIFT*Link-232 Cartridge **$39.95** • *SWIFT*Link Cable **$9.95**
Shipping: US: **$5.00** (Cartridge), **$3.50** (Cable), **$7.00** (Both). Canada add **$4.50**.

SID SYMPHONY Stereo Cartridge

Bring the Excitement of Stereo Sound to your 64 or 128

SID Symphony adds a second complete SID chip to your 64 or 128 and can be attached to your home stereo or amplified speaker systems to povide a whole new dimension in sound. Enjoy hundreds of public domain stereo songs, import MIDI files, or use Compute's Music System book and SID editor to create original stereo music.

SID SYMPHONY Cartridge **$39.95** • Compute's Music System Book **$22.95**
Shipping: US: **$5.00** (Cartridge), **$3.50** (Book), **$7.00** (Both). Canada add **$4.50**

Ordering Information and Shipping Charges

HD and SD Hard Drives:	Continental US: $25.00 per drive (UPS ground), $35.00 (2nd-Day), $45.00 (Next-Day). Canada: $50.00 (Airmail). COD to U.S. only $4.00 add'l charge. Foreign prices: Add $100.00 to U.S. Retail Price
JiffyDOS:	Add $5.00 per order (UPS ground), $9.00 (2nd-Day Air), plus $4.50 for APO, FPO, AK, HI, and Canada, or $15.00 for overseas orders. No add'l shipping if ordered with any hard drive. COD's add'l $4.00
RAMDrive:	US: $7.50 (UPS), $14.00 (2nd day), COD add $4.00, Canada $15.00 Foreign CALL.
RAMLink:	US: $10.50 (UPS), $18.00 (2nd day), COD add $4.00. Canada: $20.00. Foreign: CALL.
Payment:	MA residents add 5% sales tax. We accept VISA, MasterCard, Money Orders, C.O.D., and personal checks (allow 3 weeks for personal checks to clear). Credit card orders provide the following: Card holders name, billing address, home/work phone, card number, expiration date and issuing bank.

Canadian Customers can now contact CMD's new Canadian distributor for faster service
HOLZ COMPUTER SUPPLY, 12,1620 11TH ST. SE, CALGARY, ALBERTA T2H 2L7 PHONE: 403-262-6896

• WE VERIFY ALL CREDIT CARD INFORMATION AND PROSECUTE INDIVIDUALS ATTEMPTING TO PERPETRATE FRAUD •

Note: Prices and specifications subject to change without notice. Normal delivery is 2 to 4 weeks, backordered items may take longer. GEOS, GEOS 2.0, GEOS desktop, GEOWRITE, GEOPAINT, GEOS CONFIGURE, and GEORAM are registered trademarks of Berkeley Softworks, Inc., CP/M is a trademark of Digital Research, Commodore 64, Commodore 128, Commodore SX-64, Commodore 128D, Commodore 1700, 1764, 1750 and Commodore RAMDOS are registered trademarks of Commodore International, Amiga is a trademark of Commodore-Amiga. Macintosh is a trademark of Apple Computer, IBM is a trademark of International Business Machines. Lt. Kernal® is a trademark of Fiscal Information, Inc. RAMLink, RAMDrive, CMD HD Series and gateWay are trademarks of Creative Micro Designs, Inc.

CMD **Creative Micro Designs, Inc.**
15 Benton Drive, P.O. Box 646
East Longmeadow, MA 01028

ORDERS ONLY: 1-800-638-3263
Questions/Support: 1-413-525-0023 • **FAX:** 1-413-525-0147 • **BBS:** 1-413-525-0148
Office Hours: 10 am - 5 pm Mon. thru Fri • 10 am - 2 pm Sat. (orders only) Nov. thru Feb.

TWIN CITIES 128 - ISSUE #31 - FEB/MAR 92

Twin Cities 128 C128 - The Commodore 128 Journal -
is only $3.95 US $4.95 Canada

Table Of Contents Page #

PARSEC's Telephone number
1-508-745-9125 9-5 EST Mon - Fri - Answering Machine/Voice
10-2 EST Tues + Thur - Voice support when available
Answering machine *may* be available at all hours depending on if the line is tied up or not from my online activities.

GEnie = C128.JBEE (on everyday) CIS = 70661,443 (once a month)

Twin Cities 128 is Published and Copyrighted 1992 by Parsec, Inc
POB 111 Salem, MA 01970-0111 USA All rights reserved.

This magazine is dedicated to my mother to whom I gratefully owe all my successes and failures.

TWIN CITIES 128 CHECKSUM PROGRAM *By Michael Gilsdorf*

If you decide to type in programs from Twin Cities 128 magazine, you should first type in and run TC128 Checksum. This program checks your typing by generating a two-letter checksum each time you enter a program line and press the RETURN key. The checksum is displayed in the upper left hand corner (home position) of the 40 or 80 column screen. To check for typing errors, compare the checksum on the screen with the one appearing in the magazine listing. If they're different, then you know you've made a typing error. The magazine listing will show the correct two letter checksum in front of each line number.

TC128 Checksum will detect most typing errors such as transposed characters and misspellings, but can on rare occasion be fooled. It uses the line number and value of each character as well as its position on the line to generate the checksum. TC128 Checksum will ignore spaces unless they appear inside quotes or within BASIC keywords. You can use BASIC keyword abbreviations such as ? for PRINT without affecting the result.

TC128 Checksum is also designed to make it easier for you to indent text or enter blank lines. To indent text, simply type the line number, space or tab over to where you wish the text to begin, and then begin typing. This feature will improve the readability of your listings by making portions of your program such as FOR-NEXT loops and DO loops stand out more easily. To enter a blank line, type a line number followed by at least two spaces (or tab) and a shifted character. When the program is listed, only the line number will appear.

```
10 REM READ DATA
20
30 FOR J=1 TO 80
40     READ A$
50     B$=B$+A$
60     IF A$="." THEN J=80
70 NEXT J
```

TC128 Checksum also has the ability to generate a checksum listing. This listing will show the checksum along side each line number as the program is listed. To begin the listing, type a # in direct mode (without a line number) as the first character on a line. Do not include any additional BASIC commands on the line; otherwise they will be ignored. Once the listing begins, you can use the NO SCROLL key or STOP key to pause or stop the listing as desired. You'll find the checksum listing especially useful if you need to redisplay the checksums and double check the lines you've already entered.

```
KM 10 REM READ DATA
BE 20
OB 30 FOR J=1 TO 80
JN 40     READ A$
HB 50     B$=B$+A$
JH 60     IF A$="." THEN J=80
FM 70 NEXT J
```

Also, should you decide to submit a program listing to Twin Cities 128 magazine for publication, you can use the # command to save a checksum listing to disk. To create an a SEQ file listing, type:

```
OPEN 2,8,2,"0:FILENAME,S,W": CMD 2
#
PRINT# 2: CLOSE 2
```

The same technique can be used to send the listing to a printer:

```
OPEN 2,4: CMD 2
#
PRINT# 2: CLOSE 2
```

The TC128 Checksum program is listed below. Be sure to save a copy to disk before running it. Once run, it will automatically activate itself.

```
1 PRINT CHR$(147);"TC128 CHECKSUM  V1.0"
2 PRINT "BY MIKE GILSDORF (C) OCT 91": PRINT
3 BANK 15: FOR A=3328 TO 3583: READ D: POKE A,D: T=T+D: NEXT
4 IF T<>29208 THEN PRINT "DATA ERROR": END
5 POKE 770,0: POKE 771,13
6 PRINT "TC128 CHECKSUM ACTIVATED"
7 PRINT "TO LIST, TYPE:  #": PRINT
8 PRINT "TO DEACTIVATE, TYPE:"
9 PRINT "POKE 770,198: POKE 771,77"
10 :
100 DATA 162, 255, 134,  60,  32, 147,  79, 134
105 DATA  61, 132,  62,  32, 128,   3, 170, 240
110 DATA  14, 144,  15, 201,  35, 208,   7, 166
115 DATA  45, 165,  46,  76, 223,  13,  56,  76
120 DATA 212,  77,  32, 160,  80, 169,  32, 198
125 DATA  61, 209,  61, 208,   8, 198,  61, 209
130 DATA  61, 240, 250, 230,  61, 230,  61,  32
135 DATA  10,  67, 132,  13, 160,   0,  32,  89
140 DATA  13,  56,  32, 240, 255,  32, 129, 146
145 DATA  19,  18,  32,  78,  75,  32, 146,  27
150 DATA  81,   0,  24,  32, 240, 255,  76, 234
155 DATA  77, 162,   0, 134, 251, 134, 254,  24
160 DATA 165,  22, 101,  23, 133, 253, 177,  61
165 DATA 240,  33, 170, 224,  34, 208,   2, 230
170 DATA 251, 165, 251,  74, 176,   4, 224,  32
175 DATA 240,  14, 166, 254, 177,  61,  24, 101
180 DATA 253, 133, 253, 202,  16, 246, 230, 254
185 DATA 200, 208, 219, 152, 208,   5, 169,  45
190 DATA 168, 208,  17, 165, 253,  74,  74,  74
195 DATA  74,  24, 105,  65, 168, 165, 253,  41
200 DATA  15,  24, 105,  65, 140,  75,  13, 140
205 DATA 205,  13, 141,  76,  13, 141, 206,  13
210 DATA  96, 200,  32, 236,  66, 153,  20,   0
215 DATA 192,   3, 208, 245, 200, 169,  63, 141
220 DATA   0, 255,  32,  89,  13, 169,   0, 141
225 DATA   0, 255,  32, 129, 146,  78,  75,  32
230 DATA   0, 166,  22, 165,  23,  32,  35,  81
235 DATA  32, 181,  75, 166,  65, 165,  66, 134
240 DATA  97, 134,  61, 133,  98, 133,  62,  32
245 DATA 152,  85, 160,   0,  32, 236,  66, 133
250 DATA  65, 200,  32, 236,  66, 133,  66, 208
255 DATA 184, 197,  65, 208, 180,  76,  55,  77
```

EXPANDING THE COMMODORE 128 *by Richard Curcio* "Part 2: 4 Mode 512K"

INTRODUCTION

Several bits in the many registers of the C128's Memory Management Unit (MMU) are unused. The MMU documentation describes some of these bits as "reserved for future expansion". Such is the case for bits 4 and 5 of the Ram Configuration Register (RCR) at $D506 in I/O space. The prevailing assumption has been that these two bits would select four "SuperBanks" of 256K each. In other words, 1 Megabyte of system RAM.

The more I pondered that possibility, the more convinced I became that providing that much memory might be impractical from both a software and hardware standpoint. But the success of my implementation of the formerly non-existent RAM 2 and 3 lead me to think that some form of SuperBanking might be possible.

This modification is an extension or upgrade of Part 1. By adding another 256K in the form of eight 64K x4 DRAMs, and 3 other ICs, a 256K-modified C128 is brought to a total of 512K with four modes of operation.

THESE PLANS ARE OF NO USE WITHOUT THE 256K CIRCUITRY. PLEASE SEE TWIN CITIES 128 ISSUE #30 FOR THE PLANS, DETAILS, PARTS LIST, ETC.

DISCLAIMER

This modification will render any warranties on your equipment null and void. The Author and Publisher do not assume any liability for Purchaser's implementation of these instructions. All information is believed to be accurate.

DIFFICULTIES

Attention "flat" C-128 owners, because of the low headroom inside a flat C-128 case this project is not for the fainthearted or inexperienced. Though it can be done, it is a VERY tight fit and requires a great amount of skill to make it fit!

The MMU has the ability to keep portions of RAM 0 "common" in all configurations. This so-called "common memory" is used to hold routines that do the actual bank switching and therefore must be accessible in all configurations. Without common memory, whenever the microprocessor attempted to change configurations, code would be switched out from under it. There are ways around this but common memory is still needed to put them into memory in the first place.

To switch "SuperBanks", then, would require "Super Common Memory" --- a portion of memory accessible in all SuperBanks. Just which of the remaining unused MMU bits Commodore intended to specify SCM is unknown. They can't be at $D506; bits 4 and 5 are the presumed SuperBank bits and the 256K modification uses the formerly unused bit 7 of the RCR to select RAM 2 or 3 for VIC/DMA access, as was intended. This implies that SuperBanking with SCM capability would have to be a two-register operation. Clearly, too much hardware and tricky software would be needed.

Instead of using the SuperBank bits to select four sets of 256K, this modification uses them to give 512K four modes.

OPERATION

The little bits/blocks table within the schematic summarizes the four modes. When bits 5/4 of $D506 are %00, the normal condition after power-on or reset, the C128 behaves as if it was "only" 256K modified. When bits 5/4 of $D506 are %01 (mode "B") RAM 4/5 can be accessed as if they were RAM 2/3 after the appropriate BANK statement. When bits 5/4 are %10 (mode "C") RAM 6/7 become the new RAM 2/3. In modes B and C, RAM 4-7 might be considered "extended memory". Note that "true" RAM 0 is always in place in these modes, so common memory remains so. When bits 5/4 = %11, RAM 4-7 becomes a second, "alternate" set of RAM 0-3, complete with its own common memory, zero page and stack. Machine language is needed to use this mode. When bit 7 of $D506 is %1, letting the VIC display RAM 2 (bit 6 = %0) or RAM 3 (bit 6 = %1) then bits 4/5 determine just which RAM 2/3 (true, extended or alternate) is displayed (see program 4, "Super 8"). This also applies to REU accesses of RAM 2/3 and to the use of the Memory Mover supplied with the 256K plans; a POKE to $D506 (decimal 54534 in BANK 15), will allow you to move data between RAM 0/1 and any of the three RAMs 2/3, but you can't move data between the three sets of RAM 2/3 -- at least, not in one operation. A "SuperMover" is needed.

Starting in normal mode, after BANK 15: POKE 54534, 20 accesses of BANK 2/3 will really access RAM 4/5. POKE 54534, 36 switches in RAM 6/7 as 2/3. To return to normal mode (1K of common memory at the low end, VIC displays RAM 0, "true" RAM 2/3), POKE 54534, 4. In some instances, PEEK, AND and OR should be used in the POKE statements so that changing bits 4/5 does not unintentionally affect the other bits of the RCR. Note that before RAM 4-7 can be safely accessed, page $FF of those blocks must be initialized. More on this under "SOFTWARE".

CIRCUIT THEORY

Integrated Circuits Z1, Z3 and Z4 of the 256K circuitry are unchanged. However, /CAS1 from MMU-1 (the original) no longer goes directly to the B input of Z2. The two /CAS0s from the two MMUs form a (fairly) straightforward 2-bit "CAS code". With a third bit for

Z2's C input, all eight Z2 outputs are utilized and we now have to detect the "no bank" condition when the MMU registers at $FF00-04 are addressed (neither of the original /CASs go low). NAND gate Z5-d takes care of that via Z2's active-high E3 Enable, which was connected to +5 volts in the 256K mod.

NAND gates Z5-a, -b, and -c combine FR/W with a signal from Z4 to provide a Clock Pulse for latch Z7. This captures the value written to bits 4/5 of the RCR. (A four-bit latch was used to allow for possible "future enhancements". Circuitry for reading Z7 is unnecessary since bits 4/5 of the RCR retain whatever is written to them and we can simply read the MMU). Diode d1 pulls Z7's Master Reset low during power-up and RESET. This clears the latch contents to zeros, establishing the default condition. Whenever the 128 goes into C64 mode, MS3 goes low. Capacitor C1 will then generate a brief low-going pulse which will also clear the latch, so we start in a known state in C64 mode.

Latch outputs Q0 and Q1 go to select bits S0 and S1 of Z6, a 74F153 dual 4-to-1 mux. (An 'LS153 apparently works -- even at 2 MHZ -- but I suspect that, with respect to timing, that might be cutting things close). The "mode" selected, routes the appropriate signals on Z6's Inputs to Z2's B and C inputs. The disable switch SW is optional, but recommended. (See footnote under "C64 MODE"). When the switch is open, the Ea and Eb enables of Z6 are held high and that chip is disabled; its Ya and Yb outputs are held low, and Z2 can therefore select only the original RAM 0 or 1 -- or neither, if Z5-d's output is low. If the switch is omitted, Ea and Eb MUST be connected to ground.

Using FR/W (called R/WA in the SAMs schematic,) to generate a WRite makes latch Z7 immune to alteration by an external DMA, such as from an REU. This duplicates the MMU's immunity to outside alterations. (FR/W also occurs a little sooner than R/W and has faster edges). The astute will recognize Z5-a, -b, -c as equivalent to an OR gate, and some might be tempted to use one of the now-unused gates from U9. I advise against that. So far, only two mother board connections have had to have been cut -- R29 and R30 -- and I'd like to keep it that way. Besides, a NAND is still needed to detect the no bank condition mentioned earlier.

MECHANICAL CONSIDERATIONS

As in the 256K modification, there should be ample room for the new circuitry inside a 128D. A small circuit board can be used to hold ICs Z5-7, or the double-stick foam upside-down wire-wrapping (DSFUDWW) method can be used. Piggy-back two more levels of four 64Kx4 DRAMs to existing memory, wiring the /CAS pins of each level the same way you did for the first level of piggy-backed DRAMs. For the low-profile 128, things are a bit more involved. Low-profile owners will almost certainly have to dispense with the R.F. shield, or cut an opening in it.

In my low-profile 128, I piggy-backed four more 64Kx4 DRAMs to U46-U49, alongside the first four atop U50-U53. The four remaining DRAMs were mounted on a small board positioned in the logo area of the mother board (which I insulated with a layer of vinyl tape), and held in place with small pads of double-stick foam. ICs Z1-Z4 of the 256K circuitry remained where they had been and Z5-Z7 were installed using the same DSFUD-WW technique. The illustration roughly indicates the layout. Latch Z7, not shown, was placed near U3 and ICs Z3 and Z4. (also see the "daughter" board suggested in the Supplement with the 256K plans). The disable SWitch was mounted on the left side of the bottom case, in the vicinity of the three-pin LED connector, CN13. I used a double-pole double-throw slide-switch (even though only one pole is needed), with a slider that didn't protrude very much, to make it difficult to actuate accidentally. The case plastic is soft, so cutting a rectangular opening was easy. A mini toggle switch has the advantage of using a round hole.

The way I installed the DRAMs probably was not the best course of action. For the low-profile, I recommend that the first four DRAMs be piggy-backed to the mother board DRAMs, as described in the 256K plans, and the remaining eight installed on a small board in the logo area. If starting from scratch, another strategy is to put all 12 DRAMs on the memory board. Without the metal RF shield, there is just enough headroom for three levels of 4 DRAMs. The piggy-backing and wiring of the memory board DRAM /CAS lines would be as described for the 128D in the 256K plans; that is, only pin 16 is lifted and each /CAS wired to that pin "leap frogs" a chip. With or without piggy-backing, if you choose to retain the R.F. shield you will have to cut an opening in it to clear the memory or daughter board components. The most ambitious solution to the dimensional constraints of the low-profile 128 would be to repackage it in a new case with a profile not quite so low. A metal case would eliminate the need for an R.F. shield.

SOFTWARE CONSIDERATIONS

Program 1 performs a VERY simple test of all eight RAM blocks. When RUN, the first two columns display the bank numbers and the ram blocks being accessed. The last column should display the values 0-7. Obviously, since only one memory location is tested in each block, this is not an exhaustive performance test. Note that system IRQs are disabled while program 1 executes. While the operating system "knows" about RAM 2/3, and initializes them upon start-up, it knows nothing about the new RAM 4-7.

A very important part of initialization is the copying of identical routines into the same locations ($FF05-$FF44) in all RAM blocks. These routines handle IRQs, NMIs, and RESET by saving the 8502 registers, the current configuration, switching in system ROM and JMPing to the appropriate handler. Code is not pulled out from under the processor when these routines change banks because execution continues in the ROM version of the routine. Fortunately, initializing RAM 4-7 is simple enough to even be accomplished in Basic (program 2). The ML initialization (program 3) gets around the problem of code being switched out from under the microprocessor by temporarily enlarging common memory to 16K. Since the ML is relocatable, it can be BSAVEd and BOOTed at any convenient address in BANK 15.

As mentioned earlier, in some cases it may be necessary to use AND and OR when altering bits in the RCR. The four-Doodle program supplied with the 256K plans does this when changing bits 6/7 to display bit maps in the four blocks. The RCR is first PEEKed, and the value is then ANDed with %00111111 (63/$3F). This clears bits 6/7 and preserves bits 0-5 and thus does not change the amount or location of common memory or the presumed SuperBank (now "mode") bits. The result is then ORed with 0, 64, 128 or 192 to allow the VIC to display RAM 0, 1, 2 or 3. This lets the program run on a 256K-modified 128 or in either 256K "group" of a 512K-modified 128. The "Globe" and "Pound" adaptations also run in either 256K group. (See "TRANSFER" program.)

Program 4, "Super 8", changes bits 4/5 of $D506 before each picture is BLOADed to the "extended" RAMs 2/3. The eight pictures are rapidly displayed by changing bits 4/5 as well as 6/7 to display the eight bit-maps. Because this program uses all eight blocks, it runs only in normal mode, so AND and OR are not used in the POKEs to 54534. Programs 2 or 3 MUST be run before running Super 8.

Like the four-Doodle program, Super 8 includes TRAP and a target line to restore the RCR to normal when the program ends or is halted by an error or by pressing the STOP key. (STOP-RESTORE does not return the RCR to its default contents!) During program development, the TRAP target routine should include PRINTing of the error variables EL and ERR$ to display any errors. Do this AFTER the RCR is returned to normal -- normal being 1K of common memory at the low end, VIC displaying RAM 0, and bits 4/5 whatever they were before the program was run. (%00??0100)

TRANSFER ROUTINE

In its current state, program 5 provides a crude form of "task-switching". This ML program could conceivably form the basis of a dual-tasking mode. It begins by testing $D506 (in Bank 15) for #$04, the RCR's normal contents and, therefore, normal mode. Satisfied that the computer is in normal mode, the routine switches to mode "B" and copies part of itself to RAM 4 impersonating RAM 2. It then performs a nearly complete reset, with RAM 4-7 acting as RAM 0-3. (The MMU initialization is skipped, since that would put the machine back where it started from, normal mode). If a bootable disk is in the drive, it will boot. However, GEOS128 crashes in the alternate environment, undoubtably because it alters $D506 without preserving the conditions of bits 4/5. The 8502/Z80 transfer code is missing from $FFD0 of the alternate RAM 0, but even when it's present, CP/M also crashes in the second 256K group, probably for the same reason GEOS does. (Both CP/M and GEOS still function in normal mode, however). Speedscript 128 functions in the second 256K, and so does Spectrum, which means that BASIC 8 does, too. Anything that doesn't mess with $D506 should work properly. Any program that alters the RCR without regard for bits 4/5, always making them %00, will work only in normal mode.

The Basic loader will install the reset-and-transfer routine at an address of your choice. When installed at address SA, BANK15: SYS SA resets to alternate mode. In both environments, BANK15: SYS SA+100 will allow you to go back and forth from one 256K group to the other. Each time the routine at SA+100 is called, the processor registers and the RCR contents are saved within the routine. (The value in the Configuration Register at $FF00 is NOT saved, so the routine should be called only in Bank 15). The RCR is then changed to switch in the "other" set of 256K and new values are put in the processor registers. This process is repeated each time the current 256K group is exited. If you lose track of which group you're in, X=PEEK(54534)AND48 (in Bank 15) will give 0 in normal and 48 in alternate.

To use the DOS Shell in the second 256K, insert the disk in the drive AFTER the reset to alternate mode is completed. Then BLOAD "DOS SHELL",B0. Enter the machine language monitor and change the LDA #$04 at $3227 to LDA #$34. Exit, type KEY1, "BANK12:SYS6656" +CHR$(13). Press <return>, then press <f1>. No doubt other programs can just as easily be made to work in alternate mode. Ideally, any program that stores values in $D506 should do so in a way that doesn't disturb bits that are not to be altered. Again, this can be accomplished through the use of AND and OR, in Basic or machine language.

There are many complications to implementing true dual-tasking. The transfer operation would have to occur on each interrupt. Since this would give each environment half as much processing time, 2 MHz (FAST) mode would have to be used to get an effective 1 MHz. This would confine dual-tasking to 80 columns, which is just as well, since there is only one Color Memory for the VIC to use. (It might be possible to use the "other" Color Memory block normally used for multi-color bit-maps. But the display would alternate between two text screens, probably causing both 40 column screens to be unreadable. Using non-overlapping windows in each 256K group would work in 80 columns, but not in 40). A means must be found to keep disk and other I/O operations from "colliding". Which program receives keyboard input? Wnen STOP-RESTORE is pressed, which program stops? How are "wedges" and even normal interrupt activities affected by the IRQ-driven transfer? These are the sort of puzzles certain masochistic types take great delight in trying to solve.

C64 MODE

In C64 mode this modification can provide as much as 194,560 bytes of RAM but, for that much RAM to be available, 64 mode must be entered in a non-standard way.

Although the MMU documentation states that "there are no preconditions ... to force a particular memory alignment in C64 mode", I have not found that to be entirely true. It appears that some portion of RAM 0 must be present. That is, you can not GO64 and have ALL of any RAM block other than RAM 0. And, apparently, that required portion of RAM 0 must be at the bottom of memory. (I could be mistaken, though).

In 512K normal mode, use the machine language monitor to enter this code at $8000 in TRUE RAM 2 (omit comments):

```
28000 sei
28001 lda #$8e   ; RAM 2, I/O and kernal
28003 sta $ff00
28006 lda #$07   ; display RAM 0, 16K of
28008 sta $d506  ; com. mem. at low end
2800b jmp $ff4d  ; 64 mode
```

Exit to Basic, then load and run the 512K TEST program to store the values 0-7 at location 49152 in all eight RAM blocks. Then type BANK2:SYS32768 <return>. When the 40 column C64 screen appears, ?PEEK(49152) will display "2". RAM 0 is present from address 0 to 16383. Above that, RAM 2, 4 or 6 can be switched in by POKEing 0, 16, or 32 to 54534. Although the MMUs disappear in 64 mode, latch Z7 remains, and we can still alter bits 4/5. The fourth value, making both bits %1 by POKEing 48 in 54534, again gives us RAM 4, but ALL of it. The screen turns into garbage because the lowest 16K of RAM 0 was switched out. (This would be RAM 4 as "alternate" RAM 0 in 128 mode). Without preparation, i.e. ML routines, the only way to recover from this is by a hardware RESET.

If we had stored #$84 in $D506 while in 128 mode, in C64 mode VIC would display RAM 2 and RAM 0 would exist from $0000 to $03ff (1K of common memory). Switching Banks would switch in a new screen as well as new RAM from $0400 to $FFFF. This could only be done from an ML routine in the cassette buffer or some other location below $0400. Switching Banks would also switch out any ML in the $C000-$CFFF range. A further complication is that, because the MMUs disappear in 64 mode, the contents of latch Z7 cannot be read. (In 128 mode, bits 4/5 of the RCR retain whatever is written to them, and we can read the RCR). One solution to this difficulty is to always return to some known "base" configuration.

Of course, ML routines that use techniques like the TRANSFER routine (which uses techniques borrowed from the routines in page $FF,) could be stored in RAM while the computer is in 128 mode. These could give a means around the "vanishing RAM" problem. But since using the added RAM in 64 mode is going to be VERY tricky, we might just ignore the added RAM, always GO64 the normal ways, or disable the modification in 64 mode. If you've installed the disable SWitch, which is recommended, just flip it to the "un-mod" position.* The disabling could be automated by replacing capacitor C1 with a diode to MS3, in the same direction as d1. This would keep latch Z7 cleared and unalterable in 64 mode. With either C1 or a second diode, GO64 DOES NOT WORK INALTERNATE Reset while holding down the C= key ALWAYS works.

* So far, I've had to do this with only one program, an old version of Renegade, a disk backup utility, which is now called Maverick.

Editor's note: Richard can upgrade your "flat" C-128 or C-128D for you. The best way to contact him is to write to him and include a SELF ADDRESSED STAMPED ENVELOPE along with your night telephone number. The cost for the 256K upgrade is $80 plus parts and shipping both ways. The cost for the 512K upgrade is $95 plus parts and shipping both ways.

Write to Richard before sending him your computer so he can schedule your upgrade and give you the latest pricing information for the chips and hardware needed to do the upgrades. I am very happy with my new C-512D!

Richard Curcio
22 Seventh Ave
Brooklyn, NY 11217

512K PROGRAMS

PROGRAM 1 : 512K.PROG1

```
lk 100 rem *** test ram 0-7 ***
ki 110 :
en 120 bank15:poke53274,0:   rem disable vic irqs
mo 130 x=0
id 140 fori=0to7:ifi>3thenx=2
mp 150 ifi=4ori=5thenbank15:poke54534,20:rem banks 4/5 as 2/3
bb 160 ifi=6ori=7thenbank15:poke54534,36:rem banks 6/7 as 2/3
eb 170 bankiand3orx:poke49152,i
dg 180 next
ci 190 bank15:poke54534,4:   rem normal
be 200 x=0
mj 210 fori=0to7:ifi>3thenx=2
pn 220 ifi=4ori=5thenbank15:poke54534,20
lp 230 ifi=6ori=7thenbank15:poke54534,36
oj 240 print"bank"iand3orx;
ni 250 bankiand3orx:print"ram"i;peek(49152)
ih 260 next
pc 270 bank15:poke54534,4:  rem back to normal
ko 280 poke53274,241      :  rem enable irqs
kd 290 end
```

PROGRAM 2 : 512K.PROG2

```
ik 100 rem *** initialize ram 4-7 ***
ki 110 :
be 120 bank15:ifpeek(54534)<>4thenprint"wrong mode!":end
jo 130 poke53274,0            :rem disable vic irqs
dg 140 fori=65285 to 65348 :rem $ff05-ff44
ao 150 gosub230:next
dm 160 fori=65530 to 65535 :rem $fffa-ffff
cc 170 gosub230:next
oo 180 :
ce 190 bank15:poke53274,241:rem enable irqs
ji 200 pokedec("d506"),4   :rem normal
fc 210 end
bg 220 :
ee 230 bank15:x=peek(i)    :rem get data from rom
jl 240 pokedec("d506"),20  :rem ram 4/5 = 2/3
ea 250 gosub310
go 260 bank15
ge 270 pokedec("d506"),36  :rem ram 6/7 = 2/3
fp 280 gosub310
lb 290 return
gh 300 :
fh 310 bank3:pokei,x
fo 320 bank2:pokei,x
nj 330 return
```

PROGRAM 3 : 512K.PROG3

```
ic 100 rem *** initialize ram 4-7 in m.l. ***
ki 110 :
lo 120 sys4000
ln 130 ;
kd 140 ;buddy 128
nb 150 ;
gl 160 .mem
of 170 ;
gd 180 *= $1300
pj 190 ;
ad 200 ; call in bank 15
an 210 ;
mo 220 :        php
ck 230 :        sei          ; no interruptions
lh 240 :        lda $d506
ag 250 :        cmp #$04     ; normal"?
ej 260 :        bne msg      ; no...
mf 270 :        pha
jg 280 :        lda #$27     ; 16k of common mem.
hf 290 model    sta $d506    ; mode 'c' (6/7=2/3)
op 300 :        lda #$80     ; ram 2, rom & i/o
km 310 cnfl     sta $ff00
fl 320 :        ldx #$3f
kc 330 loopl    lda $ff05,x  ; irq, nmi & reset
lp 340 :        sta $ff05,x  ; routines
fa 350 :        dex
aj 360 :        bpl loopl
pa 370 :        ldx #$05
ij 380 loop2    lda $fffa,x  ; "hard" vectors
kc 390 :        sta $fffa,x
ic 400 :        dex
eb 410 :        bpl loop2
mb 420 :        lda #$c0     ; ram 3, rom & i/o
ji 430 :        cmp $ff00    ; is it"?
oa 440 :        bne cnfl     ; no -- make it so
jg 450 :        lda #$17     ; mode 'b'
nj 460 :        cmp $d506
lb 470 :        bne model    ; repeat for ram 4/5
bm 480 ;
oe 490 :        lda #$00
pk 500 :        sta $ff00    ; bank 15
mb 510 :        pla
do 520 :        sta $d506    ; 'normal' mode
fg 530 :        plp          ; status
ee 540 :        rts
gd 550 ;
jg 560 msg      plp
kn 570 :        jsr $ff7d    ; primm
ph 580 .byte    $0d          ; "return"
jo 590 .asc     "wrong mode!"
he 600 .byte    $0d,0
ik 610 :        rts
jm 620 .end
```

512K PROGRAMS

PROGRAM 3 : 512K.INT.LDR

```
kk 100 rem *** init 4-7 in ml ***
gf 110 bank15:sa=4864:rem relocating
jg 120 fori=0to85:readd:pokesa+i,d:next
dl 130 rem use bank15:sys sa
oj 140 data   8,120,173,  6,213,201,  4,208
mc 150 data  58, 72,169, 39,141,  6,213,169
ki 160 data 128,141,  0,255,162, 63,189,  5
ek 170 data 255,157,  5,255,202, 16,247,162
pc 180 data   5,189,250,255,157,250,255,202
gl 190 data  16,247,169,192,205,  0,255,208
ii 200 data 224,169, 23,205,  6,213,208,212
bg 210 data 169,  0,141,  0,255,104,141,  6
mo 220 data 213, 40, 96, 40, 32,125,255, 13
pm 230 data  87, 82, 79, 78, 71, 32, 77, 79
pp 240 data  68, 69, 33, 13,  0, 96
```

PROGRAM 4 : 512K.SUPER 8

```
ic 100 rem *********    super 8    *********
bg 110 rem *    press any key to stop      *
fb 120 rem *    re-start with "run 190"    *
if 130 rem *******************************
mg 140 :
ke 150 trap270:gosub280:bank15:graphic1
hc 160 fori=0to7:poke54534,z(i)
gd 170 bload(n$(i)),b(b(i)),p7168
bo 180 next:poke54534,4
dm 190 trap270:gosub280:rem re-read arrays for re-start
mk 200 bank15:graphic1:x=0
om 210 t=199:rem display duration
lh 220 ifx>7thenx=0
cn 230 poke54534,z(x)
ih 240 fori=0tot:next:rem time delay
hf 250 geta$:ifa$<>""then270
ld 260 x=x+1:goto220
ak 270 bank15:poke54534,4:graphic0:end
kf 280 restore
od 290 fori=0to7:readn$(i),b(i),z(i):next
ll 300 return
hb 310 :
ce 320 rem  name, bank, rcr value
mk 330 data "ddmiddle earth",0,4,"ddopbox1",1,68
da 340 data "ddfront pic",2,132,"ddbackcover",3,196
mn 350 data "ddgraphic blocks",2,148,"ddplan",3,212
aa 360 data "ddspiral",2,164,"ddletters",3,228
kn 370 :
fb 380 rem  some of these pix are art studio
nn 390 rem  converted to doodle
ll 400 rem  you can use any doodle file
md 410 rem  in place of the above doodles
```

PROGRAM 5 : 512K.TRSF.BAS

```
bi 100 rem *** transfer loader ***
mm 110 bank15:sa=5120:rem relocating
lg 120 fori=0to149:readd:pokesa+i,d:next
ln 130 ad=sa+96:gosub420
ia 140 pokesa+32,l:pokesa+34,h
ck 150 fori=0to5:readaa,dd:ad=sa+dd:gosub420
lg 160 pokesa+aa,l:pokesa+aa+1,h:next
fe 170 print"reset    = bank15:sys"sa
di 180 print"transfer = bank15:sys"sa+100
do 190 end
pm 200 data 173,  6,213,201,  4,240, 18, 32
ld 210 data 125,255, 13, 87, 82, 79, 78, 71
jo 220 data  32, 77, 79, 68, 69, 33, 13,  0
ak 230 data  96,120,  9, 20,141,  6,213,169
em 240 data  96,160, 20,133,206,132,207,169
je 250 data 206,141,185,  2,160, 53,177,206
eb 260 data 162,  2, 32,119,255,136, 16,246
ak 270 data 160, 30,169, 41,162,  2, 32,119
dj 280 data 255,200,169,207,162,  2, 32,119
nj 290 data 255,200,200,200,200,200,169, 52
pn 300 data 162,  2, 32,119,255,216,162,255
lj 310 data 154,169,  0,160, 52,140,  6,213
co 320 data  76, 21,224,234,234,  8,120,141
ap 330 data 147, 20,142,145, 20,140,143, 20
kg 340 data 104,141,140, 20,186,142,137, 20
ee 350 data 173,  6,213,141,132, 20,  9, 48
bc 360 data 141,  6,213,169,  4,141,  6,213
ei 370 data 162,246,154,169, 48, 72,160,  0
ab 380 data 162,  0,169,  0, 40, 96
ga 390 rem ** adjustments **
ik 400 data 104,147,107,145,110,143,114,140
cf 410 data 118,137,124,132
hj 420 h=ad/256:l=ad-int(ad/256)*256:return
```

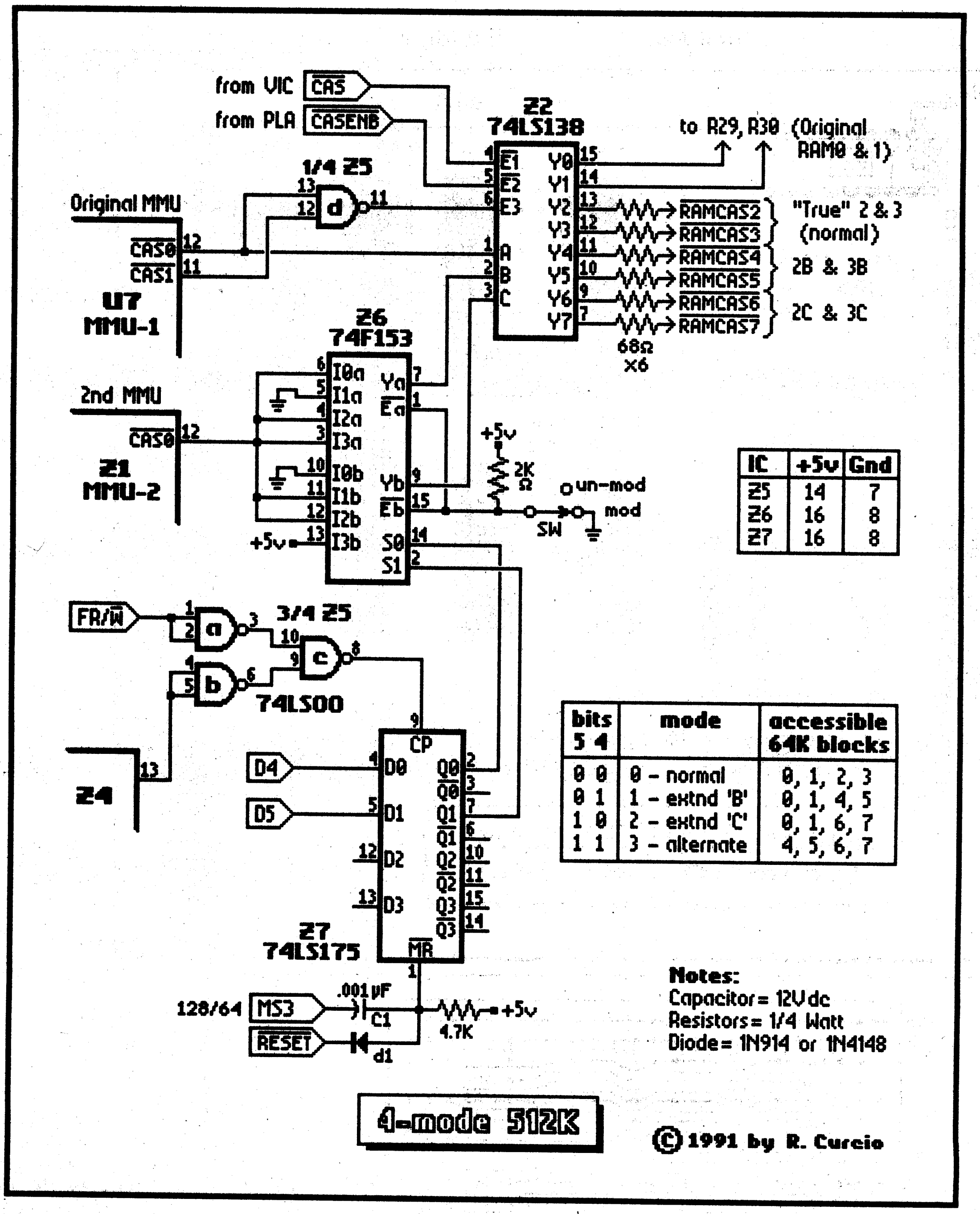
PROGRAM 5 : 512K.TRSF.SRC

```
jf 1000 rem ***  transfer control from   ***
ah 1010 rem     normal to alternate mode
dj 1020 :
fa 1030 sys4000
ep 1040 ;
df 1050 ;buddy 128
gd 1060 ;
pn 1070 .mem
hh 1080 ;
pk 1090 *= $1400
il 1100 ;
jf 1110 ;
lg 1120 ; reset to alternate
kj 1130 ;
ef 1140 areset  lda $d506    ; check mode
oa 1150 :       cmp #$04
cb 1160 :       beq copy
ah 1170 :       jsr $ff7d    ;primm
```

PROGRAM 5 : 512K.TRSF.SRC - continued

```
eo 1180 .byte   $0d
pi 1190 .asc    "wrong mode!"
mo 1200 .byte   $0d,0
oe 1210 :       rts
ad 1220 ;
na 1230 copy    sei          ; no interruptions
li 1240 :       ora #$14     ; mode 'b' makes
il 1250 :       sta $d506    ; ram 4 = ram 2
ik 1260 :       lda #<switch
oi 1270 :       ldy #>switch
ig 1280 :       sta $ce      ; set-up pointer
oc 1290 :       sty $cf      ; for indsta
dp 1300 :       lda #$ce
eg 1310 :       sta $02b9
gi 1320 ;
jc 1330 ; copy instructions beginning at "switch" to
ne 1340 ; same locations in ram 4 impersonating ram 2
ig 1350 ;
jm 1360 :       ldy #end-switch
mo 1370 cloop   lda ($ce),y
dl 1380 :       ldx #$02     ; bank 2
ho 1390 :       jsr $ff77    ; indsta
hc 1400 :       dey
dm 1410 :       bpl cloop
mm 1420 ;
if 1430 ; make indicated changes
oa 1440 ;
jh 1450 :       ldy #srcr-switch
dc 1460 :       lda #$29     ; "and"
ci 1470 :       ldx #$02
mf 1480 :       jsr $ff77
pb 1490 :       iny
ap 1500 :       lda #$cf
fa 1510 :       ldx #$02
on 1520 :       jsr $ff77
bj 1530 :       iny
ce 1540 :       iny
co 1550 :       iny
di 1560 :       iny
ec 1570 :       iny
ga 1580 :       lda #$34
kb 1590 :       ldx #$02
do 1600 :       jsr $ff77
il 1610 ;
gn 1620 :       cld          ; begin reset sequence
ad 1630 :       ldx #$ff
dc 1640 :       txs          ; stack pntr
hb 1650 :       lda #$00
id 1660 :       ldy #$34     ; altn mode
pb 1670 :       sty $d506
nb 1680 ;
ba 1690 ; the next instruction is never
ei 1700 ; executed in "true" ram 0.
```

```
op 1710 ;
jo 1720 switch  jmp $e015    ; later in reset
ml 1730 :       nop
nf 1740 :       nop
bh 1750 ;
an 1760 ; this routine lives in true ram 0 and true
gf 1770 ; ram 4 (pseudo 2 in mode 'b' or second ram 0
gn 1780 ; in alternate mode) with changes as noted.
hl 1790 ; permits jumping from normal to alternate.
ek 1800 ;
cg 1810 xfer    php          ; start here
gh 1820 :       sei          ; no interruptions
ce 1830 :       sta atemp+1
pn 1840 :       stx xtemp+1
ba 1850 :       sty ytemp+1
bm 1860 :       pla          ; retrieve status
nl 1870 :       sta srtemp+1 ; store it
kc 1880 :       tsx          ; stack pointer
om 1890 :       stx pntemp+1
cf 1900 :       lda $d506    ; get ram config reg.
ab 1910 :       sta crtemp+1 ; save for return
ci 1920 srcr    ora #$30     ; change bits 4/5
ed 1930 ;                      (use 'and #$cf'
op 1940 ;                      at same point
ph 1950 :       sta $d506    ; in ram 4)
ok 1960 ;
ii 1970 ; at this point current 256k group is switched
nm 1980 ; out. execution continues at same point in
np 1990 ; "other" 256k group
bc 2000 ;
bb 2010 crtemp  lda #$04     ; first time value
og 2020 :       sta $d506    ; (use #$34 in ram 4)
da 2030 ;
fo 2040 pntemp  ldx #$f6     ; first time
ea 2050 :       txs
jj 2060 srtemp  lda #$30     ; ditto
ne 2070 :       pha
bf 2080 ytemp   ldy #$ff
bg 2090 xtemp   ldx #$ff
fb 2100 atemp   lda #$ff
ee 2110 :       plp
lm 2120 end     rts
ii 2130 .end
```



IC	+5v	Gnd
Z5	14	7
Z6	16	8
Z7	16	8

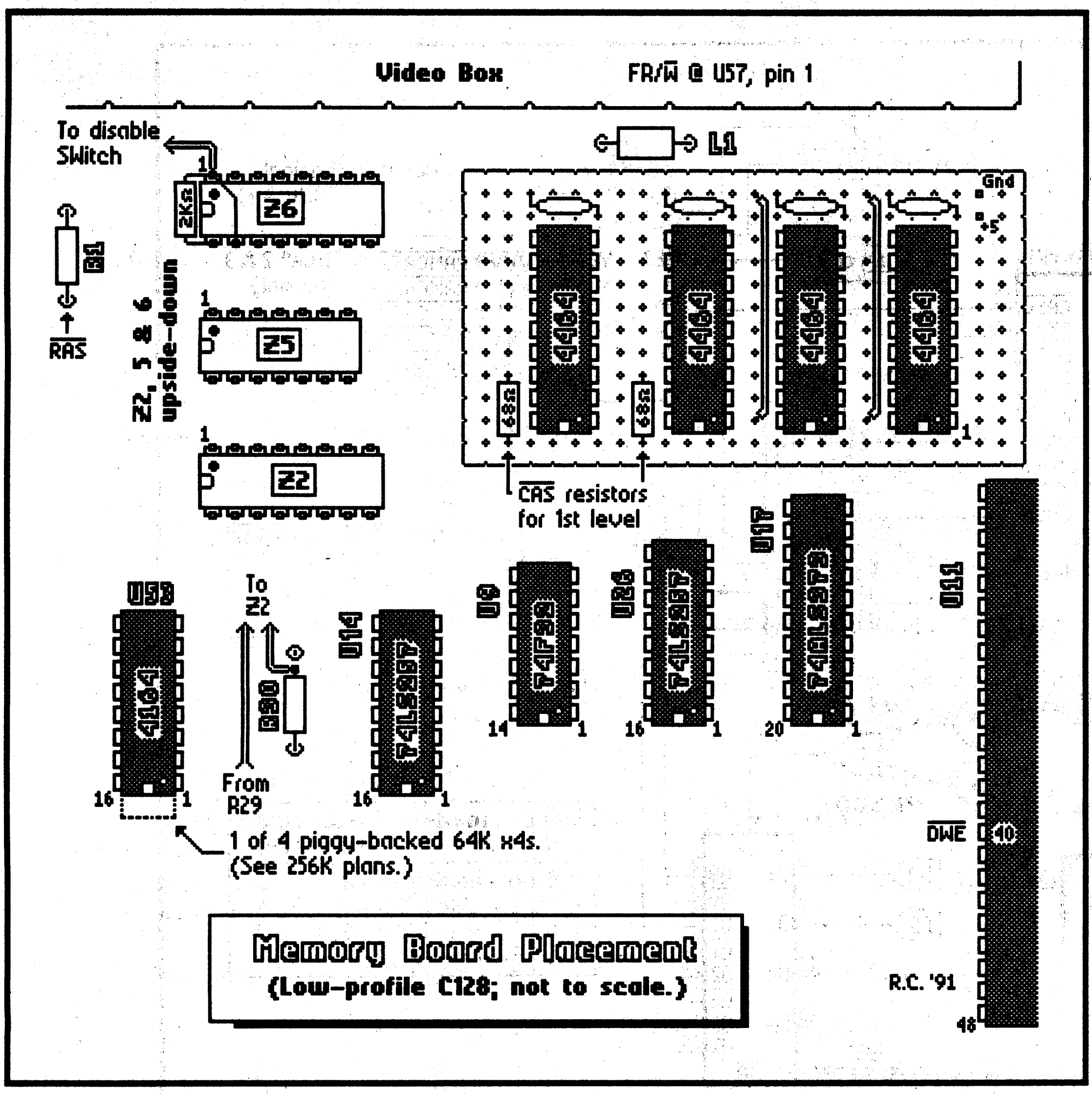
bits 5 4	mode	accessible 64K blocks
0 0	0 - normal	0, 1, 2, 3
0 1	1 - extnd 'B'	0, 1, 4, 5
1 0	2 - extnd 'C'	0, 1, 6, 7
1 1	3 - alternate	4, 5, 6, 7

Notes:
Capacitor = 12Vdc
Resistors = 1/4 Watt
Diode = 1N914 or 1N4148

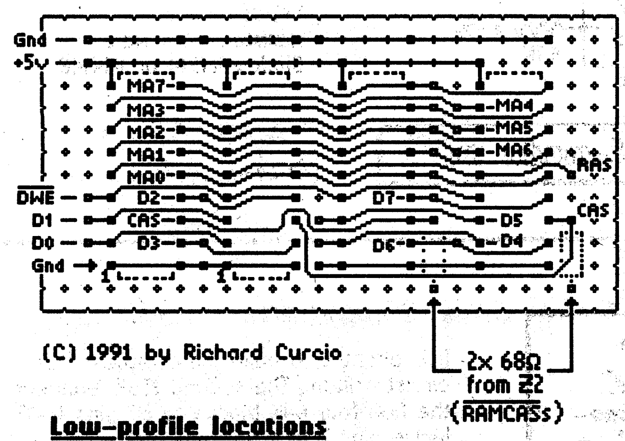
Signal	Location * "Flat"	"D"
MS3	U11, pin 15	-same-
RESET	U6, 40	"
FR/W	U57, 1	U61, 3
D4	U13, 12	-same-
D5	U13, 14	U13, 3

[*] See 256K plans for other signals.

It is possible to build the complete 512K circuit omitting the second 256K. Connect the first four 64K blocks to Z2 pins 12-15, leave pins 7 and 9-11 un-connected, and jumper pin 1 of Z7 to ground. The switch will still enable/disable the mod. When you add the second 256K, remove the ground jumper from pin 1 of Z7.



Memory Board Placement
(Low-profile C128; not to scale.)



Signals	Low-profile locations
MA0-7	U45, pins 5-7, 9-13
D0-7	U18, pins 9-11, 13-17
DWE	U11, 40
RAS	R1
RAMCASs	from Z2

Memory Board Wiring

Bottom view with pin 1 toward front of mother board. Note +5 and Gnd reverse of logic chips. ◘ = top-side cabling

This picture to the left is for the 512K upgrade and for C-64 mode !

Dialogue 128

(v2.2)

"... a leading contender for the 'finest piece of code ever written for the Commodore 128.'"

Frank Hudson - Dialogue v2.0 review
TC128 issue #27

Superior Telecommunications Software for your C128 or C128D Computer

- Regular and SwiftLink (CMD UART cartridge) versions supplied
- 300-4800 baud (bps) operation (300-38400 with SwiftLink)
- Dialogue is designed for all types of users. Program can be controlled by mouse, joystick, or keyboard. Pull-down menus AND direct key sequences are available
- Punter, Xmodem (Chk, CRC & 1K), Ymodem batch, and CompuServe Quick-B upload/download protocols are supported
- Integrated full featured text (buffer) editor
- Support for all major modem types: 1650, 1660, 1670, Volks 6480, Volksmodem 12, Hayes compatibles, and MNP/V.42 modems
- Fully compatible with: RAMDOS, RAMLink, RAMCard, CMD Hard Drives, Lt. Kernel, and IEEE drives. Partition support for 1581. Burst support on 1571/1581
- 64000 byte (800 line) capture buffer is configurable from 1-8 seperate buffers. 1750 REU support allows up to eight 64000 byte buffers, or a huge 512000 byte buffer
- VT100, VT52, IBM ANSI (color), CBM graphics, and ASCII terminal emulations
- 40/80 column and 25/50 line selectable display (on 80 column screen)
- Powerful auto-execute script language allows programming of automatic sessions
- Dialogue 128 is supplied with a well laid out, 61 page manual
- Online support for Dialogue available on:
  CompuServe: 76703,3050
  GEnie: G.FARMANER
  Q-Link: Dialogue
  PunterNet: C128 conference and 138/Gary Farmaner
  Support BBS: (416) 630-4367
- Dialogue disk is NOT copy protected
- Dialogue requires: C128/C128D, 1541/1571 drive, 80 column display and modem
- Demo version available on major online services or by mail (see below)

Make checks or money orders payable to:

Triple Point Software
#369 - 253 College St.
Toronto, Ont, Canada
M5T 1R5

Check with your favorite dealer, or order direct:
Price: $29.95 (US funds)
$34.95 (in Canada)
Please add $4 for shipping and handling

The following are available only from Triple Point Software:
Upgrades: $14.95 (with manual addendum)
$19.95 (with new manual)
Demo disk: $2.95
Upgrades and demo disk include shipping
Canadian residents add GST. Ontario residents add PST

A Review of KeyDOS *by Michael Gilsdorf* "A Function ROM for the C128"

What is KeyDOS?

KeyDOS is a EPROM chip containing a number of utilities designed to enhance the C128. As opposed to ROM cartridges which plug into the cartridge port, KeyDOS is installed either in the empty socket inside the C128 (called the Function ROM socket), or inside any 17xx Ram Expansion Unit (REU). Of the two options, the Function ROM installation is the easiest. For the C128D, simply open the case and plug the chip into the socket. If you own a flat C128 you'll need to unsolder and remove the metal shield before you can reach the socket. On the other hand, if you wish to install the chip inside the REU, you'll need to cut a circuit trace(s) and solder in a 32 pin socket. The socket is not included, but is easily obtained from Radio Shack for about a dollar.

Along with the chip, KeyDOS is packaged with a 5.25 inch disk and a 50+ page user's manual. The disk is not crucial to the operation of KeyDOS, but it does contain a number of useful programs and examples - more about that later. I found the manual easy to use, but the small print coupled with a poor xeroxed copy made some pages hard to read. KeyDOS is written by Randy Winchester, and is sold through Antigrav Toolkits, PO Box 1074, Cambridge, Massachusetts. It carries a money back guarantee as well as a lifetime warranty!

Operation

After KeyDOS is installed, the first thing you notice after you power-up is the KeyDOS message. It informs you KeyDOS can be activated by holding down the ALT key when either resetting the computer or SYSing 65366. (The KeyDOS disk also contains a couple of programs for activating KeyDOS.) I was happy to see the SYS number displayed on the screen. I can't recall how many times in the past I've forgotten a number and had to dig through documentation to find it. The manual makes no mention how to deactivate KeyDOS, but other than resetting the computer, pressing the stop and restore keys seems to do the job. A more graceful means to deactivate KeyDOS and restore the original function key assignments while preserving BASIC would be more desirable. KeyDOS is compatible with JiffyDOS, GEOS, Quick Brown Box, CS-DOS, and Commodore RAMDOS.
However, you may need to disable it for some software. I found PaperClip would not load with KeyDOS active.

A help menu listing all the KeyDOS commands and utilities can be displayed on the screen by pressing the ESC and back-arrow keys. There are 21 other ESC key combinations along with the commands assigned to the function keys.

Function Keys

F1	Load program
F2	Run program in 128 mode
F3	Display disk directory
F4	Run program in 64 mode
F5	Display SEQ file
F6	Scratch file
F7	Change drive with POKE 186
F8	Scratch & save file
RUN	Boot file from disk
HELP	Print "ON U(PEEK(186))"

If you're not happy with the above default key assignments, they can be reprogrammed. A one-time change can always be done using the Basic "key" command - provided you know how to use it. If not, or if you wish to reprogram the run and help keys, then you're directed to use the KeyDOS Compiler utility (ESC 2). The name compiler is somewhat of a misnomer since it is not a language compiler. Rather, it is a program which creates a boot sector and program on disk. When a reset occurs or the BOOT command is invoked, the computer loads the program and redefines the function keys. The KeyDOS Compiler allows you to choose from a list of 20 predefined definitions for programming a key. Noticeably absent from the menu, though, were the LIST command, and a command to display and list a BASIC program file to screen from disk.

To minimize key strokes and make it easier to access the drive, many of the function key commands have "ON U(PEEK(186))" appended to the tail end. Since location 186 holds the current device number, its value can tell you what drive was used last. Although this is an improvement over the default CBM assignments, I was a little disappointed to see PEEK(186) used with the function keys. It's generally safe to PEEK(186) in the beginning of a program to determine what drive was last used (a few KeyDOS utilities do this), but it can lead to problems if used in direct mode and the value is not checked. Location 186 can hold any legal device number including zero, so a syntax error will result if a disk command is issued when the location contains any number less than 8. In all fairness a warning is given in the user's manual concerning this quirk, but nevertheless a different method could have been employed. Perhaps a better way would have been to use an alternate location (e.g., $BE), and update it only if a drive is used. Another nice improvement would be to display the current drive as a command prompt while in direct mode, similar to CP/M. That way, a user would have no doubt about what drive is current before issuing a drive command such as NEW or SCRATCH.

KeyDos Utilities

The heart of KeyDOS is its utility programs. Since they're in ROM there is no need to search for a disk or wait for them to load. With a simple ESC key combination, a program is available at your finger tips. These utilities are a mix of BASIC and machine language (ML) programs collected from various sources including the Transactor. Those which rely on BASIC, though, will occupy and overwrite any resident BASIC program you happen to have in memory. KeyDOS provides no warning of this, nor is any given in the user's manual, so be sure to save the program to disk before using a BASIC utility. Below is a list of the utilities along with a brief description.

ESC (left-arrow) Display KeyDOS help menu.

ESC 1 Executes KeyDOS Utility program. This program is a file management utility written partly in BASIC. You can easily send commands to the drive, as well as select and mark files in the directory listing for renaming, scratching, copying, displaying, and printing files. CBM ASCII and standard ASCII conversion as well as multi-drive support are also provided. Files can be copied only from one drive to another, and REL files are not supported. (For copying files using a single drive, see ESC = .) Unlike using the function keys when in direct mode, the current drive is displayed as a command prompt.

ESC 2 Executes KeyDOS Compiler. This BASIC program allows you to customize the function keys and save the configuration to disk as a boot sector. You can program a function key from a list of 20 predefined key assignments. No means is provided to create your own definitions so you don't have the same flexibility as the KEY command, but programming a key is much easier.

ESC 3 Creates 1581 partitions and subdirectories. One feature which sets this program apart from other similar programs is the displaying of the BAM (track and sector allocation map). This is a real "must" when creating partitions on a disk which already contains files.

ESC 4 Reprograms the F8 function key to make it easier to access 1581 partitions and subdirectories.

ESC 5 Reprograms the F8 function key to execute a batch program. A batch program is a SEQ file on disk which contains a sequence of BASIC commands which are executed as if you typed them while in direct mode. The KeyDOS disk contains a few examples of batch programs. Sadly, KeyDOS doesn't appear to offer much improvement in the creation or execution of batch files above what is described in Miklos Garamszeghy's article published some time ago in the Transactor. Not only is creating/editing batch files still cumbersome, but disk access is not allowed. In addition, the READY prompts and carriage returns are not suppressed, so it's difficult to maintain control of a screen display. A small SEQ text editor is needed for editing/debugging a batch program, but none is supplied. These shortcomings all tend to limit the usefulness of a feature with a powerful potential and commonly found on other operating systems.

ESC 6 Install and activate CBM RAMDOS 128. Allows the REU to be used as a RAMdisk - a utility very handy to have in ROM! Andrew Mileski's patch provides support for REUs with 2 Meg upgrades.

ESC 7 Reboot GEOS from the REU.

ESC 8 Execute Diskmon - a popular disk monitor by Anton Treuenfels. This monitor supports drives with different device numbers, but unfortunately, should you exit the monitor and call this utility again, a system crash occurs.

ESC 9 Disassemble memory and send output to disk or printer.

ESC 0 Execute Hexpert debugger. This debugger, written by Eric Trepanier (c) Advanced Software Systems, wedges itself into the C128 monitor and allows you to set break points and execute a machine language program one step at a time while observing memory and register contents - a very useful utility indeed!

ESC + Change the device number of a drive without powering it off.

ESC - Reset all drives with device numbers from 8 to 11.

ESC \ Execute a COLLECT command while preserving the boot sector. Careful with this one... it preserves only the first sector of track one. Boot programs which reside on multiple sectors are not preserved.

ESC * Unnew a program.

ESC (up-arrow) Toggle between two 80 column screens. Operation is similar to the Transactor program by D.J. Morriss.

ESC : Search/replace and scroll utility. This utility is the ROM version of Joseph P. Caffrey's Programmer's Aid for the C128 which was published in Transactor. It includes two handy features: a search and replace function within a range of line numbers, and a forward and backward scroll/list feature controlled by the cursor keys. Although this utility is very useful, the cursor has a tendency to lock-up when scrolling upwards on a blank screen. Also, once enacted, the utility can not be disabled without a reset.

ESC ; Execute a screen dump to printer or drive.

ESC = BASIC program to copy one or more files using one drive.

ESC , Activate scratchpad. This key combination allows you to use the screen as a scratchpad for temporary notes and memos.

ESC . Executes Clock Manager. Allows you to set and display an on-screen alarm clock.

ESC / Video Manager. Allows you to set screen colors, size, and various display attributes.

Final Observations

The manner in which KeyDOS is packaged is perhaps one of its strongest appeals. An EPROM chip that can be installed inside either the computer or REU provides a nice degree of flexibility seldom seen for C128 products. Furthermore, since the utility programs are in ROM, they are quickly and easily made available by simply pressing no more than a couple of keys.

However, these strengths don't overcome KeyDOS's weaknesses. Aside from the problems previously mentioned, a few KeyDOS utilities could use some additional polishing to make them operate more smoothly and correct other minor quirks which occasionally crop up. For example, at times when a drive error light flashed, the utility would not display the error message, nor allow it to be read. At other times an error would cause the utility to suddenly abort with no explanation as to why. In addition, exiting a utility would not always restore things as they were originally (e.g., re-enable stop/restore keys, restore window parameters, etc.).

But among KeyDOS's shortcomings, the most frustrating one was the reprogramming of the F8 function key when using ESC 4 and ESC 5. With the F8 key having any one of three possible definitions at any given time, it is easy to forget just what the current F8 definition is. Worse, the KeyDOS help menu remains unchanged and continues to display the default key assignments. On one occasion I ended up scratching and saving a file instead of executing it as a batch program. When this occurred I found it impossible to unscratch and recover the original file since its sectors had been already overwritten. From then on, I used the KEY command to check the key's definition, but this may not help users who have little or no knowledge of BASIC. A better solution would be to use a different function key - one that doesn't have the same potential for disaster.

KeyDOS utilities could also be better integrated and more consistent. For example, while some utilities support a wide range of device numbers, the drive reset utility supports only 8 through 11. Also, it's hard to understand why the single and multi-drive file copy utilities weren't combined into a single general purpose file copier program, and why REL file support wasn't included too, when today's better commercial file copiers support it.

Equally puzzling was why some utilities were chosen to be installed on the KeyDOS chip. A few of these routines can be easily duplicated with a minimum amount of BASIC programming, or may be seldom used, or used only once to initially configure the computer. The KeyDOS Compiler utility is a good example of the latter case. I would have liked to see it replaced with a more useful feature such as having the help key display the last-command used (similar to the F3 key in MS-DOS), or a drive number prompt while in direct mode, or a small text editor. True, the KeyDOS Compiler makes programming the keys easy, but since users typically will not be creating boot programs on a regular basis for the sole purpose of reprogramming the function keys, why have it in ROM? Not only does the C128 operating system already provide support for this feature (i.e., KEY command), but similar programs can be found in the public domain which perform the same function and allow you to create a bootable program. Stranger still, the KeyDOS disk already contains a program (Keyboot by Geoff Sullivan) which also reprograms the function keys and creates a boot program. By adding a SYS call to the kernel to allow the run and help keys to be redefined, this program could provide the same functions as the KeyDOS

Compiler. Here again, integrating all the key reprogramming functions and options into one single disk-based program would have made more sense.

On a more philosophical note, one also has to wonder why BASIC rather than ML was used for some KeyDOS routines. Considering some utilities were designed to aid BASIC programming (e.g., ESC :), it's odd that KeyDOS would have other routines whose use could easily destroy your program without so much as a warning. Once you know which routines are dependent upon BASIC, you can save your program to disk before calling them and afterwards retrieve it; however, this work-around is not very convenient or foolproof.

Lastly, although KeyDOS is compatible with JiffyDOS, it does not directly support it. I found this disheartening in light of the fact JiffyDOS enjoys wide spread support from a large user base, and is quickly becoming (or is?) a C128/64 defacto standard. No JiffyDOS, RAMLink, or hard drive commands are listed among the 20 predefined key definitions, nor does KeyDOS allow you to create any. Furthermore, JiffyDOS owners will find some of KeyDOS's default function key assignments and utilities redundant. Most annoying, though, was the need to restore the function keys back to their original JiffyDOS settings. Since KeyDOS does not allow you to do this, you'll have to find and use yet another program to redefine the function keys and create a boot sector. Unless you own a hard drive, you may find it a hassle to retrieve and insert a disk for booting everytime a reset occurs just to restore the key assignments. Sadly, this could have been avoided by giving users the option to press a key other than ALT, so the function key re-assignment routines could be by-passed when KeyDOS is activated.

In conclusion, I have mixed feelings about KeyDOS. While I can't recommend it for all C128 users, some may find some of the utilities handy to have in ROM. But with a few improvements and more unique features, KeyDOS has the potential of a becoming a real "power" chip appealing to a wide spectrum of users and providing them features not found anywhere else.

OVERALL RATING: C-

$32.95
Antigrav Toolkit
PO Box 1074
Cambridge, MA 02142

Twin Cities 128 Subscription Information

Magazine only:
6 issues a year - $20 for the US
6 issues a year - $26 all others

Magazine with companion disks:
6 issues a year - $36.50 for the US
6 issues a year - $46.50 for all others

User Group Rates (minimum of two subscriptions)
Magazine only:
6 issues a year - $17 for the US
Magazine with companion disks:
6 issues a year - $26.00 for the US

For the US:
The cost to add the companion disk to TC128 from a current subscription starting with issue #32 is $4.00 for each remaining issue.

For all others:
The cost to add the companion disk to TC128 from a current subscription starting with issue #32 is $6.00 for each remaining issue.

Effective 920201 we will acknowledge all (re)subscriptions, change of addresses, and subscription inquires by either a letter or postcard.

How to read your address label:
1st line: ## #

The first number is the issue on which your subscription ends and the second number following it will have a "1" if you get the companion disk.

2nd line:Attention line (optional-usually not used)

3rd line:Your Name
4th line:Your Address
5th line:City, State, Postal Code
6th line:Country (not written on US labels)

Make checks payable in US funds to:

PARSEC INC
PO BOX 111
SALEM MA 01970-0111
USA

EXPANDING THE C128 EXPANSION PORT

A Hardware Review by Noel Nyman

Reviewed:

SS100 Plus - 80 line digital I/O board
$139

Dual 6522 VIA - four "user port" I/O board
$169

16 Channel ADC module - converts analog signals to digital values(also requires Dual VIA board)
$69

All from
Schnedler Systems
PO Box 5964
Asheville NC 28813
1-704-274-4646

My friend Mark is a C128 owner who runs a plastic injection molding business. He has designed a circuit board that monitors the molding machines. It is connected to counters and function switches, and measures the speed of each machine.

Mark's board plugs into his C128's expansion (cartridge) port. Custom programs log all molding machine activity and alert him to low production rates, downtime, and emergencies.

With the decoding and buffering required, the board became large and elaborate. Mark spent several months designing, building, and debugging it. If the board fails, he has no backup, and the system is shut down until he can repair it.

The I/O (Input/Output) boards manufactured by Schnedler Systems may be cost effective alternatives for projects like Mark's. They work with either a C64 or a C128 (in all three modes), and have these advantages:

* fully assembled and tested
* industrial grade components
* all chips socketed for easy replacement
* work with most other peripherals
* fully documented, with sample programs and data sheets on major integrated circuits
* sample software disk for C64 and C128 modes in BASIC and machine language
* replacements, backups and repairs readily available

80 LINES, NO WAITING ---

Schnedler's least sophisticated board contains forty input lines and forty output lines addressed as five 8-bit ports in the $DEF8-$DEFC range (57080-57084). Most software and peripherals ignore this area completely. For products such as SwiftLink-232 that use $DExx addresses, the board can be changed to the $DFxx page by a jumper.

You read an input port with a PEEK in BASIC or a "load" instruction in machine language. The data is buffered, but is not inverted. The inputs are TTL (Transistor-Transistor Logic) level compatible. Most applications will use switches instead of voltages. Schnedler's excellent documentation provides several sample switch circuits.

A machine language "store" instruction or a BASIC POKE to a port address changes the outputs. Output ports latch "set" or "clear" ("on" or "off") until you change them again. The C128's reset line clears all outputs automatically when the computer is turned on or reset.

The outputs are solid state current sinking switches. Each output line can switch up to 50 volts and each port can sink a total of 500ma (milli-amps). Many small relays and lamps can be switched directly by the output lines, saving additional hardware. The output buffers contain internal back biased diodes for use with inductive loads.

A fuse protects the computer against electrical problems. You can use a separate power supply for the I/O board by connecting it to pads Schnedler provides.

Common IDC connectors are used for the inputs and outputs. Schnedler lists mail order sources, including 800 numbers, for connectors and cables. The board includes an expansion bus socket for any other device that must also use the C128's cartridge port.

INTERRUPTS AND TIMERS ADD VERSATILITY ---

For monitoring emergency signals, you might prefer Schnedler's Dual 6522 VIA board. It contains two fully decoded VIAs (Versatile Interface Adapters) which are earlier versions of the 6526 used for the C128's user (modem) port. The VIAs provide a total of 32 I/O lines. Each line can be software programmed as an input or an output. There are several additional "handshaking" lines available on each VIA.

The VIAs can be programmed to activate an interrupt to the MPU (MicroProcessor Unit) when input signals arrive. The C128's user port has this capability, too. But, like the RESTORE key, it's connected to the NMI (Non-Maskable Interrupt) line in the computer. The

NMI is "edge" sensitive. It will only trigger an interrupt if a sharp voltage increase occurs. A gradual increase, produced by gently pressing on the RESTORE key for example, may not generate an interrupt. Most signals require conditioning for reliable NMI operation.

Schnedler's board uses the IRQ (Interrupt ReQuest) line instead. IRQ is "level" rather than edge sensitive. Even a gradual rise in voltage will trigger the interrupt. This board is a good choice for monitoring events that do not occur often, but MUST be serviced immediately. When the event occurs, you can respond to it immediately. Your program will not have to waste time polling the input lines.

Each VIA has two very accurate clock/timers available. They can time events, generate pulses, or create timed IRQ interrupts. The VIA registers are fully decoded in the $DExx or $DFxx page. Several VIA boards can be used together on the same computer. Switches change the absolute addresses of each board within the address page.

Schnedler's manual provides sample programs in BASIC and machine language that utilize most VIA features. An accompanying disk saves you the trouble of typing in the code. The manual contains manufacturer's data sheets on the 6522 which fill in any details for obscure applications.

I'M A DIGITAL PERSON IN AN ANALOG WORLD ---

The most sophisticated circuit Mark designed measured the molding machines' speeds. He used ADC's (Analog to Digital Converters) to change the analog voltage representing motor speed to a digital value with which the C128 can work.

Schnedler's ADC0817 chip provides sixteen "multiplexed" analog inputs. The same ADC circuit looks at each input in sequence. With a 100 microsecond conversion time for each input, multiplexing doesn't slow things down in a C128 system. The ADC can be programmed to ignore inputs that are not used.

Using the same measuring circuitry for all inputs is convenient if you're comparing the several signals. No balancing of individual ADCs are needed. A disadvantage is false values if an input line becomes disconnected. If the cable for input #2 of a three line system breaks, the ADC output will "float" to some number between the values for input #1 and input #3. You can design the input circuits to let you detect this problem.

The C128 comes with four ADC inputs built in ... the "paddle ports". If you've tried to use them, you know that they aren't very reliable. BASIC programs will not read the paddle ports. Machine language software averaging of numerous samples is required to get valid data. Even then, the accuracy is limited. In contrast, Schnedler's ADC board gave me rock solid, consistent results using fixed inputs over a forty-eight hour test period.

The ADC board plugs into Schnedler's Dual VIA board for address decoding and can't be used by itself. ADC addresses are separate from those used by the VIA's and all the VIA lines and capabilities are available.

The ADC0817 chip provides 8-bit resolution. This works well with the C128's data bus, but may be inadequate for demanding applications. The documentation provides information and schematics for a 12-bit resolution ADC. You have to build your own circuit board to use a different ADC chip.

COST vs. CONVENIENCE AND RELIABILITY ---

Most of Schnedler's customers are schools, labs, and manufacturers. They find the C128 (or C64) an easily programmed, inexpensive way to monitor or control experiments and production systems. These institutions have the expertise to build dedicated I/O boards themselves. But, their research and design costs would be much higher than Schnedler's prices.

For the home user, cost is a more important factor. Schnedler's least expensive board costs more than a new C64. Are they worth the price for your project?

If you have electronics expertise, you can probably build your own dedicated I/O board. You may even have the parts in your "junk" box. But, parts cost is only part of the story. It will take you time to design and build your board. You'll have to deal with bus loading and timing factors. Subtle errors may give you unreliable data under conditions that are difficult to predict. You may have problems adding inputs or outputs later.

In contrast, Schnedler's boards are assembled and tested. With the software provided, they take very little time to adapt to your system. If your goal is a working project rather than the fun (or frustration) of creating your own system, Schnedler's products are a good investment.

RIO COMPUTERS **800-782-9110** ORDERS ONLY
MON-SAT 8AM-6PM PACIFIC TIME

CUST SERVICE/TECH SUPPORT
AUTOMATIC VOICE/FAX SWITCH
702-454-7700
TUE-SAT 10AM-6PM

VIDEOFOX

The Tool For Creative Video Buffs

Generate video titles, opening credits, window advertising, animation or other small trick movies

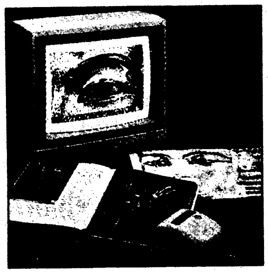
- ■ All of these exciting effects are are easy and fun you to do with our new Videofox software
- ■ Over a dozen special effects such as scrolling, combing, winshield wiper and spiral mixing
- ■ Mix text, graphics and effecs to produce hundreds of combinations
- ■ Independent adjustment of foreground and background colors
- ■ Page flipping in real time for perfect animation sequences

ONLY $59.95

VIDEO DIGITIZER

- ■ Digitize black and white or color pictures
- ■ Digitize any video source including VCR
- ■ Digitize either 4, 7 or 13 level grey levels
- ■ Menu controled picture brightnes
- ■ Includes three independent software programs for total control and editing of digitized images: DIGISON ~ VIDEOFOX ~ DIGIMULTI
- ■ Free color filters included for digitizing color images from black and white cameras
- ■ seperate adjustment of brightnes levels for each of the red - green - blue primary colors
- ■ Easy transfer of pictures into Pagefox

ONLY $249.95



HANDYSCANNER 64

The Worlds First Handscanner for the 64!

- ■ Professional quality super high 400 dots per inch resolution ~ Reads the graphics from any printed document
- ■ Converts any material to digitized graphics in seconds ~ B/W setting for crisp reprodution of high contrast line art
- ■ Elaborate grey-tone scale digitizes color or black & white photos using 3 built in dithering settings
- ■ Enlarge or reduce 300% to 33% ~ Graphic memory of 640 X 400 standard (640 X 800 with Pagefox module)
- ■ Included software has all the standard functions of a good drawing program

ONLY $299.95

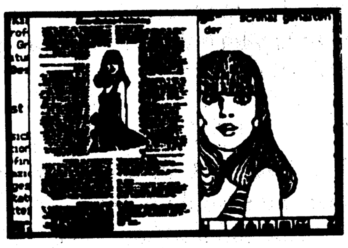
PAGEFOX

3 Easy To Use Editors For Perfect Home Desktop Publishing

GRAPHIC EDITOR ~ TEXT EDITOR ~ LAYOUT EDITOR

- ■ Completely menu driven
- ■ 100Kb storage enlargement module keeps entire page in memory
- ■ Uses proportional mouse or joystick for total control over text, graphics or picture

ONLY $139.95



ACTION REPLAY V 5.0

THE ULTIMATE UTILITY/BACKUP CARTRIDGE FOR THE C64/128

Allows You To Freeze The Action Of Any Memory Resident Program And Make A Complete Backup To Disk

WARP 25 - The worlds fastest disk serial Turbo
- ■ Typical backup will reload in under 5 seconds
- ■ No special formats-save directly into Warp mode
- ■ Warp Save/Load available straight from BASIC

RAMLOADER - Loads most commercial originals 25 times faster than normal!

UNIQUE CODE CRACKER MONITOR -
- ■ Full monitor features
- ■ See the code in its Frozen state not Reset state

MORE UNIQUE FEATURES - Menu driven operation
- ■ Simple operation: Just press a button at any point
- ■ All backups reload WITHOUT cartridge at Warp speed
- ■ Sprite killer: Make yourself invincible-disable collisions
- ■ Freeze HiRez screen & save in Koala & Paddles format
- ■ Print out any screen in 16 grey scales
- ■ 100% compatible with ALL drives and computers
- ■ Disk utilities: fast format, directory, list and many other commands operated directly from function keys

MIDI 64 -Only $49.99

- ■ Full specification MIDI at a realistic price
- ■ MIDI In - MIDI Out - MIDI Thru
- ■ Works with Sampler and Adv. Music System

MIDI CABLES (4 ft. prof. quality) -Only $*8.99
FREE cables when you buy MIDI & ADV. MUSIC at same time

DIGITAL SOUND SAMPLER

Only - $89.99

THE ADVANCED OCP ART STUDIO

COMPREHENSIVE, USER FRIENDLY ART AND DESIGN SOFTWARE

Only - $29.99

ADV. MUSIC SYSTEM

Powerful modular program for creating, editing, playing and printing out music

- ■ Playback thru internal sound or external MIDI keyboard/synthesiser
- ■ Print music in proper musical notation together with lyrics using **PRINTER** module
- ■ Enter music a note at a time in written music format using the **EDITOR** or via on screen piano **KEYBOARD** emulator or via an externally connected **MIDI** keyboard
- ■ Generate almost unlimited sounds with the flexible **SYNTHESIZER** module
- ■ Linker joins files to form large compositions

Only - $29.99

MAKE THE MOST OF YOUR ACTION REPLAY

GRAPHICS SUPPORT DISK

- ■ View screens in a slide show sequence
- ■ Add scrolling messages to your saved screens
- ■ Contains full sprite editor
- ■ Explodes sections of saved screens to full size

Onl.y - $19.99

SUPERCRUNCHER - ONLY $9.99

Turn your Action Replay into a super powerful program compactor. Reduce programs by up to 50%! Further compact progrms already crunched by Action Replays compactor

RIO COMPUTERS
3430 E. TROPICANA AVE. #65
LAS VEGAS, NV 89121

*Add $5.00 shipping/handling in the continental U.S.: $8.00 - PR, AK, HI, FPO, APO: $11.00-Canada/Mex: C.O.D. orders add $4.00 to above charges: SPECIFIY COMPUTER MODEL WITH ORDER: VISA/MC/Checks/Money Orders/C.O.D. Accepted: Please call for return authorization number - returns may be subject to a 20% restocking fee: We reserve the right to substitute products of different appearance but equal quality and function for items pictured above: Prices subject to change without notice.

800-782-9110
702-454-7700
IN NEVADA

THE GATEWAY *by Dick Estel* "Some Nice Features, but It's Not for Everyone"

When Creative Micro Designs (CMD) developed the RAMLink hardware for the Commodore 64 and 128, they realized it would have to be compatible with GEOS. Failure to make it so would have eliminated a large part of the market for this package of hardware that controls and provides a usable DOS interface to expansion RAM.

The result is Gateway, an alternative GEOS operating system that replaces the normal Desktop. In addition to providing the GEOS link for the RAMLink and related hardware, Gateway can be used on any Commodore 8-bit system instead of the normal system supplied by GeoWorks (formerly Berkeley Softworks). The Gateway works with all BSW GEOS software and most third party programs.

Gateway provides a screen display that edges away from the GRAPHIC orientation that is the "G" in GEOS, and towards a text display. In my opinion this is a weakness shared by virtually all the "alternate" desktop programs that have come out in the last few years. More about that later.

The main part of the Gateway screen shows a list of the programs on the disk, with a small icon to the left of each file name, leaving at least some visual representation of what the file is about. The file name display can be narrowed down to show names only, or widened to include the file type and the size.

At the left of the screen is a "fuel gauge" which represents the amount of disk space used. Clicking on this gauge brings up an info box type display for the disk, giving the amount of kilobytes used and the amount remaining, as well as what drive it is on and the last time the disk was modified. On the right side is a vertical bar with arrows at top and bottom, and a movable button. Clicking on the arrows scrolls up or down one filename, while clicking on the bar scrolls a full screen. In addition, the button can be moved by holding down the fire button, bringing up another section of the directory. This function is not very precise, but if you are familiar with the disk you are working on, you will be able to get pretty close to the desired area with it.

Across the bottom of the screen are icons for each available disk drive, and the familiar trash can. The icon for the selected printer does not exist in Gateway. Clicking on a disk icon changes to that drive as expected, but clicking on the trash can brings up a directory display showing all files that have been discarded and may be recovered. Since the trash can is emptied automatically when you change disks or open a file, this system does not really protect scratched files any better than the original Desktop system.

The top menu bar is similar to the Desktop, and clicking brings down a sub-menu, but there are several key differences. The GEOS menu includes the Control Panel, which is one of the major differences in Gateway. The key item here is the Chooser, which is supposed to allow changing printer or input drivers. Unfortunately, on my copy it does not, and a number of other people have reported the same problem. The Control Panel also permits changing screen colors, setting up F Key definitions, and resetting the clock.

The ACTION menu is similar to FILE in the Desktop, with the ability to act on the selected file(s). VIEW offers a choice of file types, so that you could display only application data files, for example. SELECT allows the highlighting of all files or all files on a page, just as with Desktop.

SPECIAL includes one of the really neat features of Gateway, the Set Desk Icon which lets you create custom icons for each disk. You must first find or design the icon and copy it the desired disk. Then you click on the icon, click on SPECIAL, and click on SET DISK ICON. The chosen icon becomes the icon shown for that disk at the bottom of the screen.

Also under SPECIAL is a feature unique to Gateway, BROWSE. This is actually a utility to allow you to search for specific file names. The search is case-specific, but wild cards may be used. This allows you quickly to find a specific file on a large, full disk. When the file name is found, the directory display switches to that part of the list and the name flashes briefly.

I thought Gateway had several strengths and a lot of weaknesses. One of the later is such a major problem that I put my copy of the program away and went back to the regular Desktop, but I will emphasize the positive first. My favorite feature is the ability to create customized disk icons. I got carried away and created icons for about 15 different types of disks, and have made these available for Gateway users. I liked the easier and quicker way to get to the info box, which is accomplished by clicking on the little icon to the left of the file name.

When you dump files into the trash, the screen is not redrawn as with Desktop...the scratched file names simply disappear and the rest of the names move up to fill the space. This is a big time saver since Desktop redraws the screen after each file is deleted. When doing multi-file copying the display does not switch to the part of the directory listing where the file is shown like Desktop does; however, Gateway redraws after all the files are copied.

The ultimate plus of Gateway is the Switcher. If you have ever used geoWizard, which allows you to switch over to virtually any other GEOS program, and return instantly to where you left off when you're done, you have an idea of what Switcher does. However, the

Gateway tool works a bit differently. When you want to switch to another program, you activate the Switcher, then quit the file you are in, return to Gateway, and open the other file. From this point on you can hit the Switcher hot key and instantly change from one to the other with no Open/Close/Quit choice necessary. This is a huge convenience for many projects. It does have some limitations--unlike geoWizard, you can't go just anywhere. For example you can not open the same Desk Accessory in both files. You can only switch back and forth between the two locations you set up, until you QUIT one of them. Sadly, Switcher and geoWizard are 100% incompatible; use of either will destroy the other until you re-boot.

Overall, I was not that delighted with Gateway. Why not? It is my opinion that each step away from the graphic representation of files established by the original GEOS desktop is a step away from the reason GEOS exists. This is a personal opinion, perhaps not widely shared, but this approach, common to all the alternate Desktop programs I have seen, detracts from their value to me. I am used to finding the desired file by looking for its icon; it is more trouble to find it in a text listing.

I was never able to configure Gateway to boot up with the drives set up the way I wanted. Gateway uses device drivers rather than the Configure file to control your system setup. Despite placing the REU driver in the appropriate position, Gateway always booted up with my 1581 as drive A, 1571 as B, and no REU. I could then set up the REU, which made it drive C. I work with the 1581 as drive A and the REU as drive B, using a 1571 for occasional tasks. Running a program from the REU caused it to become drive A, switching the 1581 into the C position. I had to do several additional steps to get the 1581 and REU set the way I wanted them (as A and B respectively). This was a big negative, but there were several minor annoyances.

I did not like having to click three times before I could reset the clock. I missed having a printer icon and the ability to drag a file to it (instead I have to highlight the file, click on ACTION, and click on PRINT). I prefer the numeric kilobyte information that is always visible on the Desktop. Moving files around is less flexible than with Desk Top...you hold the Commodore key, click on the icon to the left of the filename, and move a ghost icon to the desired location. I thought that the Chooser ought to be in working condition before the program was released. The only way I could change printer drivers was with a public domain desk accessory program, Change Printer, AND moving the desired driver to be the first one on the disk.

Finally, the ultimate negative - I could find no way to recover from a lockup, meaning that any work in RAM was lost. I have identified a couple of situations that tend to cause a lockup, and there are many unexpected ones. I can recover 95% of the time by pressing the reset button. There is a program called Rescue that will handle about half the lockups that can't be recovered by resetting. But none of these methods worked with Gateway. This in spite of the fact that the documentation encourages you to copy RBoot to the Gateway boot disk. After losing a significant amount of work a few times, I abandoned Gateway. Overall I would be inclined to use Gateway with some GEOS projects if this problem and the Switcher failure were fixed. Any project where I expect to switch back and forth between two applications would be greatly simplified by using Gateway's Switcher. If I needed to do a lot of switching with various program, I think I would stick to my regular boot disk with geoWizard.

It has been acknowledged that Gateway has little to offer the GEOS user with a one-drive system, and it is not really much help to anyone without a RAM expander (the switcher only works in the REU). Should you use Gateway? Consider what is involved in your most frequent GEOS projects, read as many reviews as you can, and think about how the writer's comments apply to your own situation. This will allow you to make an informed decision based on your needs and interests.

A Post Script: When I booted up Gateway to do some final checking for this article, I soon experienced a crash in which my RAM drive became inaccessible. I got an "Illegal Track" error. I have no idea why, but everything in RAM was lost, even though I was still in Gateway.

(Dick Estel is the former editor of The Interface, newsletter of the Fresno (California) Commodore Users Group, and a devoted GEOS enthusiast)

Rating: C

Editor's note: just as this magazine was going to the printers CMD released a new set of patches for Geos (not Gateway) by Jim Collette that will allow you to use Geos and the Desktop with your Ramlink.
I believe now the "offical" policy is that the new patches ship with Ramlink instead of Gateway.
The cost of the disk with the patches is $10 and available from CMD or you can download them from CMD's private library on GEnie.
More news on this subject in the following issue.

SYSTEM 6 *by Howard Herman*

Bremer's System 6 (v2.0) does most of what it says it will do. No more, somewhat less. That is okay. Why it does, what it does, the way it does, is puzzling.

System 6 is described as a complete personal finance management and control system. It is intended for individual use, primarily for forecast budgeting and analysis of personal expenses, using double entry accounting.

It does this nicely. However, a lot of data has to be entered. This is no fault of System 6. As with any good accounting program, if sparse data is entered, there will not be anything meaningful to report.

As an example, instead of entering one amount monthly for a Visa charge card, each charge made using the card is double entered, and another double entry for the monthly check paying Visa. This is the sort of detail required.

For this effort at data entry, the program will give reports that let you know if you are meeting your intended budget and an over-view of finances. System 6 was my first hands-on experience with personal budgeting. I was surprised to learn how much was being spent on a couple of unnecessary discretionary items. In reporting this type of information, System 6 performs very well.

The 30 page manual does a good job of explaining how a double entry accounting system works, and offers plenty of sample entries for use during its tutorial. No prior accounting knowledge is needed to run System 6.

The program comes on two 5 1/4" copy protected disks (an original and backup). Appendix C in the manual explains how to easily copy the program on to a 1581 disk.

Printed reports are pre-formatted. There is no option for custom reporting. A feature that is missing is a comparison of expense items as a percentage of total expenses and income. There is no way to compare budget expenses with the usually recommended guidelines.

Because System 6 is based on monthly comparisons of budgeted and actual income and expense, it is difficult to enter, and to get useful reports for items that do not occur every month. A stock paying a quarterly dividend will show in System 6 reports as budgeted income that either falls short, or exceeds expectations. Expense items present the same dilemma. Quarterly and annual payments for insurance, taxes, club memberships, magazine subscriptions, etc., distort the monthly budget reports.

Most good productive software will allow for custom configuration. There are many examples: GEOS, Digital's Pocket programs, BobsTermPro128, Dialogue128, Paperclip iii, Devpak 128, there are many more.

Unfortunately System 6 does not fall into this class. It presumes a lot, and offers few options. Example: Each time the program runs from a two drive system, the user must: first turn off drive #9, boot from drive #8, then turn on drive #9. Before ending a session: turn off drive #9, and once exited from System 6, turn on drive #9.

The user is presented with the welcome option of not using drive #9. If decided that this may be the better way, and a system has a drive #9, it is necessary to turn drive #9 off before running the program, and then on again. Only two visits to the on/off switch.

System 6 will not boot if drive #9 is actively on line.

Those having multiple drives may be happy to learn that drives higher than #9 can be left on. This is not documented in the manual.

The on/off switches for my several drives are not easily accessible. Even if they were, I would not want to go through these gymnastics each and every time System 6 is run.

I anticipated several sessions with System 6. I also wanted to avoid having to visit a chiropractor. I decided to slip a disk (sic) into drive #9 (a 1581) with a copyright cbm 86' USR type file on it. Doing an initialize to change #9 to a more acceptable #13. This meant, however, that to System 6, my system now had one drive, a #8. This required another disk swap. One disk in drive #8 to boot the program. After booting, a swap to put the data disk into drive #8. The program disk requires its file to be listed first in the directory. System 6 requires a separate disk for data. One new data disk each month.

When entering data into System 6, if you make a mistake, you cannot go back to correct it. Often, you must complete the entire entry. Only then are you given the option of not saving it. Allowing the user to return to an apparent mistake, correct it, and make further entries would be easier and quicker.

The manual describes how an incorrect entry can be changed or deleted from the General Ledger. However, every time I tried this, the program crashed. This is a bug that needs to be fixed. There is also a minor bug (buglette?) when printing reports. However, this may be due to my printer configuration.

A few other things that are puzzling:

* The opening screen requires selection of using either one or two drives. An optional screen color selector is here too. This screen appears first, every time System 6 is run. These sort of options are selected once, rarely a second time, and are best hidden away in a sub-menu, letting the program boot direct to its useful functions.

* Only petascii output to the printer.

* The user is returned to a menu selection after each data entry. Since data is normally entered in batches, it would be easier having the entry screen appear again, ready for more data.

* The program writes to disk after each entry, making data entry slow with a 1581 drive.

* Because a new data disk is required each month, the program controls the new disk formatting. This makes it difficult or risky to include other program or data material on the disks used by System 6. If System 6 is run from a hard drive or ram drive the user will have to be extremely careful to protect other files.

System 6 will allow entry of up to 200 monthly records, in 120 accounts, with a maximum of 30 accounts in any one accounting group. If the 200 monthly maximum is not sufficient, a month can be closed early, and a new disk used for additional entries.

For personal finance budgeting, to see where your money goes, System 6 does its job. It might have been nicer if it did a bit more. And especially if it did what it does differently, leaving to the user to decide the best configuration for use on his or her system.

Rating: C-

$44.95 (includes S&H)

Bremer Systems, Inc.
4242 Collie St
Lilburn, GA 30247

Twin Cities 128 Information * Explosion *

This is a subscription special only available for a limited time and only available to User Groups!

***Order as many back issue copies (limit 20 of each item) of TC128 #20, #24, #28, #29 as your Group needs for a low flat fee of $4.00 to cover the UPS shipping (offer valid only in the US). Overseas and Canadian UGs contact us with amounts etc etc to get the price on surface parcel post shipping.**

***The first fifty User Groups to order will receive as a bonus a set of FREE Transactor microfiche. Each set of microfiche ranges from Vol 4 to Vol 8. The number of issues in each set of microfiche range from 19 to 4 issues. Order early and get a bigger package! Since most of you meet in schools, colleges, and libraries were they have microfiche readers I figure this is a nice little bonus for your members!**

Requirements:
Sign up two or more members with the disk subscription at the discounted User Group rates (2 x $26 = $52) between Feb 14, 1992 and April 1,1992.

Send us a copy of your latest User Group newsletter.

Provide us with a street address for UPS shipping.

***while supplies last**

Make checks payable in US funds to:

PARSEC INC
PO BOX 111
SALEM MA 01970-0111

DRIVING THE SERIAL BUS *by Michael Gilsdorf*

A Brief Look at Commodore's Serial Bus Command Structure

One of the things which sets the C128 apart from its other 8-bit cousins is its serial bus. The bus is unique since it has two separate and distinct modes of operation - burst and standard serial. The standard serial mode has been with us for some time now, having its beginnings back when the VIC 20 and C64 were introduced. However, the serial bus was not the first bus that Commodore used. Those of us who can recall the PET and 128B computers will remember that they used a parallel bus structure which was modeled after the IEEE 488 standard. But, even though all these buses differ in the way they send and receive data, they all share a common bus command language for managing and controlling the activity on the bus.

Bus commands should not be confused with the DOS commands such as Scratch, Rename, Copy, New, etc. Bus commands are special command bytes used to control the devices on the bus such as disk drives and printers. These commands instruct a device to send data (Talk), receive data (Listen), or stop sending or receiving (Untalk or Unlisten). The serial bus can accommodate multiple listeners, but only one talker is allowed on the bus at a time. If more than one device was to talk, the data would become garbled, handshaking would get confused, and the bus would lock up.

Bus command bytes appear like any other byte on the bus except for one important difference. Normally, the attention (ATN) line (pin 3 on the serial I/O connector) is high. But when a bus command byte such as Talk or Listen is about to be sent, the computer pulls the line low and generates an interrupt signal for each device on the bus. A device acknowledges the interrupt by signaling it is busy or ready for data. If it's busy, the device will finish its task, such as writing data to disk, then notify the computer it is ready to read the command byte. The ATN line will continue to stay low until all devices are ready and the last bus command byte is sent. However, once the ATN line returns high, any subsequent bytes sent over the bus will be interpreted as information bytes (e.g., data bytes, DOS commands, etc.). After the last information byte is sent, the computer holds the ATN low once more while it sends an Untalk or Unlisten command. This informs all the devices to stop listening or talking, and allows them to return to some other task.

Controlling the ATN line is usually assigned to only one device, the computer. It is designated the bus controller, and its job is to ensure no more than one device is talking on the bus at any given time. But even though the bus controller can send Talk and Listen commands to any device, not all devices may be designed to be both a talker and a listener. For example, a printer would have no need to send data to the computer, hence it would not respond to a Talk command. On the other hand, a disk drive must be capable of both talking and listening since it must send and receive data.

(Note: Except for the CMD HD series hard drives, no other peripheral device has the hardware necessary to change the state of the ATN line. This capability gives the HD a considerable amount of independence from the computer allowing it to send data directly to other devices, swap device numbers, etc.)

Primary Address Command Group.

Bus command bytes can be classified into one of three groups based upon their functions. The first group is called the Primary Address Command Group. Bytes in this group are all constructed in a similar manner. They're composed of a command portion (bits 7-5), and a device number (bits 4-0). Before this byte is sent, the ATN line goes low which signals all devices that a bus command is being sent. Then, all devices read the command byte to determine whether it is being addressed. The following table list the primary address bytes along with their meanings:

PRIMARY COMMAND	BYTE	DESCRIPTION
Listen	001 XXXXX	Commands device X to read the information bytes which follow. (legal values: $20-$3E)
Talk	010 XXXXX	Commands device X to send information bytes. (legal values: $40 - $5E)

The value X is the same number as the device number used in BASIC statements such as Open 3,X,3 or Load "FILENAME",X,1. Although the bit structure allows X to range from 0 to 31, only values from 4 to 30 can be assigned to devices on the serial bus. Device 31 can not be assigned since it is reserved for a special purpose which is described below. The remaining numbers (0-3) are assigned for other uses (i.e., keyboard, cassette, RS-232/modem, and screen).

Unaddress Command Group.

The second group of bus command bytes is called the Unaddress Command Group. This group contains only two command bytes, and could really be considered an extension of the Primary Address Command Group described above. Not only are the bit structures for both groups identical, but the ATN line also goes low prior to a command being transmitted. However, there are two important differences which set this group apart. First, as opposed to the Primary Address Command Group, the ATN line returns high immediately after the command is sent. The second difference concerns itself with bits 4-0. See the table below. Notice that device number 31 is now defined and takes on a special meaning for ALL the devices on the bus. That is, each device on the bus behaves as if it has two device numbers; 31 and its own unique number between 4 and 30.

UNADDRESS COMMAND	BYTE	DESCRIPTION
Unlisten	001 11111	Commands all devices on the bus to stop listening to information bytes. (value: $3F)
Untalk	010 11111	Commands all devices on the bus to stop sending information bytes. (value: $5F)

Secondary Address Command Group.

The third and last group is called the Secondary Address Command Group. The bytes in this group always follow a Talk or a Listen command byte, and are composed of two parts, a command nibble (bits 7-4), and a secondary address nibble (bits 3-0). After this byte is sent, the ATN line goes high. The table below is a list of the secondary address bytes and their meanings:

SECONDARY COMMAND	BYTE	DESCRIPTION
Open	1111 YYYY	Commands the device specified by the primary address to open a channel and assign it to secondary address Y. This bus command should not be confused with the OPEN command used in BASIC. (legal values: $F0-$FF)
Close	1110 YYYY	Commands the device specified by the primary address to close the channel assigned to secondary address Y. This bus command should not be confused with the CLOSE command used in BASIC. (legal values: $E0-$EF)
Secondary Address Channel	011Y YYYY	Commands the device specified by the primary address to use the channel assigned to secondary address Y. Bit 4 can be used to flag a device to operate in some special mode; however, some devices such as disk drives will ignore it. (legal values: $60-$7F)

The value Y uses the least significant bits of the secondary address found in BASIC statements such as OPEN 3,5,Y or LOAD "FILENAME",8,Y. However, a disk drive will only recognize the low nibble (bits 3-0) and secondary addresses from 0 to 15. For example, OPEN 2,8,2 and OPEN 3,8,18 would both have the same secondary address and drive channel associated with it. You can demonstrate this for yourself. First, create a SEQ file on disk.

```
10 open 2,8,2,"0:test,s,w"
20 print#2,"data"
30 close 2
```

Next, read the file using the same low nibble as the secondary address for both OPEN statements. Then, read each character while alternating between logical file numbers. You'll find that both logical file numbers are using the same channel (or buffer) in the drive. The program below demonstrates this technique.

```
10 open 2,8,dec("02"),"0:test,s,r"
20 open 3,8,dec("12"),"0:test,s,r"
30 j=2
40 do while st=0
50   get#j,a$: print a$;
60   j=5-j
70 loop
```

Now, compare this result to what you get when using OPEN 3,8,3. Replace the DEC("12") with DEC("03") in line 20, and read the file again. Notice in this case two channels are assigned - one per OPEN statement. In most cases, this is desirable, so it's best you assign secondary addresses below 16. But be aware that some secondary addresses are reserved by some devices and have special meanings. For example, disk drives reserve 0 for loads, 1 for saves, and 15 for the error/command channel.

The bus command "Open" can have slightly different meanings for different devices. For example, in the case of disk drives, DOS doesn't open and assign a channel to a secondary address until it receives the information string containing the file name and channel direction (i.e., Read or Write). The Open bus command simply tells the drive to place the information string which follows into the command buffer. DOS doesn't really need to open a channel to the command buffer since it remained open and assigned to secondary address 15 when the drive was initialized. Even though the computer may close its command channel and direct the drive to do likewise, DOS continues to keep it open.

If you examine the above tables you will find that are a number of bytes which are not defined. These are $00 to $1F and $80 to $DF. A few of these are defined in the IEEE 488 standard, but are not recognized by Commodore's serial bus.

While we're on the subject of bus commands, it might be worthwhile to briefly mention the "end of information" (EOI) marker which is sometimes referred to as "end of file" (EOF). For Commodore computers, this marker is not a byte, but rather is a special handshake that is performed when the last information byte is transmitted. When a computer receives this handshake, it sets the STatus variable accordingly.

To better understand how the bus commands work, lets look at some examples. The first line shows the commands as they appear in BASIC. The listing that follows shows the bytes and the order in which they are transmitted on the serial bus. Study these examples and you can see how the primary and secondary address bytes are created. In another article, we look at a technique which describes how to have one device send data directly to another device (spooling).

If you have any questions or comments, you can reach me on Q-Link under the name "MIKE ALL" or on GEnie under "M.GILSDORF1". Until then, easy DOS it!

(see the program examples on the next page, Page 29)

EDITORIAL THOUGHTS and UPDATES:
By John Brown

The Most Effective Way To Cope With Change Is To Help Create It!

Many people, like me, have keep their 128 computers all these years because we realize one very basic thing. The C128 is a great machine and we like using it! When our (only?) friends, the engineers at Commodore, put together this "hedge bet" machine to keep Commodore afloat they worked wonders. From being rom based with the built in ml monitor, a z80 under the hood along with a 65xx, an easy to use and well rounded Basic language, to a neat 80 column screen, this machine has it all. How many computers come with the built in ability to output two different pictures to two different screens at the same time? Not to mention a TV too! The best feature of all was it was affordable. From the initial purchase price, the software, to the add-on items, it was a true home computer and it still is! Which brings us to the next paragraph.

Many readers have written to us asking why we have been devoting so many pages to the memory upgrades for the C-128. For those of you that had early issues of any Commodore magazine, including this magazine, may remember the "C-256" that was just right around the corner, well now it is just around the page! There are two simple reasons for running these articles:

1) People have wanted this upgrade since 1986 and never received it!
2) If you do not get this information here where are you going to get it???

Upgrades paths with expanded memory and speed have been made widely available to other computer owners from both the OEM computer maker and aftermarket developers. Examples are:
MAC users (Mac +, SE, etc)
PC users (the original"64k" PC, XT, AT, 386, etc)
APPLE owners (C,+,E, GSii etc)

But C128 and C64 owners have not had much of an upgrade path for speed or memory. Maybe it was not "really" needed. But this is the 90's and these items are both wanted, needed, and expected. People want to do more with their machines. Call it a challenge, call it tinkering, or just call it fun, but people want it! We as users, Commodore businesses, and programmers have to satisfy these needs to keep the 128 and 64 up to par and vital into the 90s. The problem with the C-128
(continued on page 33)

DRIVING THE SERIAL BUS

Example #1

OPEN 1,8,15,"S0:NAME": CLOSE 1

ATN	TYPE	BYTES	MEANING
Hi			
Lo	Pri	$28	Listen
Lo	Sec	$FF	Open (use command channel)
Hi	Info	$53 $30 $3A $4E $41 $4D $45 $0D	S0:NAME
Lo	Unaddr	$3F	Unlisten
Hi			
Lo	Pri	$28	Listen
Lo	Sec	$EF	Close
Hi			
Lo	Unaddr	$3F	Unlisten
Hi			

Example #2

OPEN 2,9,19,"0:NAME": GET#2,A$: CLOSE 2

ATN	GROUP	BYTES	MEANING
Hi			
Lo	Pri	$29	Listen
Lo	Sec	$F3	Open (use command channel)
Hi	Info	$30 $3A $4E $41 $4D $45 $0D	0:NAME (open data channel)
Lo	Unaddr	$3F	Unlisten
Hi			
Lo	Pri	$49	Talk
Lo	Sec	$73	Sec Addr (use data channel)
Hi	Info	$XX	Data byte sent by drive
Lo	Unaddr	$5F	Untalk
Hi			
Lo	Pri	$29	Listen
Lo	Sec	$E3	Close (close data channel)
Hi			
Lo	Unaddr	$3F	Unlisten
Hi			

Example #3

```
OPEN 2,8,2,"0:NAME1,P,R": OPEN 3,9,3,"0:NAME2,S,W":
GET#2,A$: PRINT#3,"DATA": INPUT#2,B$: PRINT#3,"INFO";: CLOSE 3: CLOSE 2
```

ATN	GROUP	BYTES	MEANING
Hi			
Lo	Pri	$28	Listen
Lo	Sec	$F2	Open (use command channel)
Hi	Info	$30 $3A $4E $41 $4D $45	0:NAME
		$31 $2C $50 $2C $52 $0D	1,P,R (open 1st data channel)
Lo	Unaddr	$3F	Unlisten
Hi			
Lo	Pri	$29	Listen
Lo	Sec	$F3	Open (use command channel)
Hi	Info	$30 $3A $4E $41 $4D $45	0:NAME
		$32 $2C $53 $2C $57 $0D	2,S,W (open 2nd data channel)
Lo	Unaddr	$3F	Unlisten
Hi			
Lo	Pri	$48	Talk
Lo	Sec	$62	Sec Addr (use 1st data channel)
Hi	Info	$XX	Data byte sent by drive
Lo	Unaddr	$5F	Untalk
Hi			
Lo	Pri	$29	Listen
Lo	Sec	$63	Sec Addr (use 2nd data channel)
Hi	Info	$44 $41 $54 $41 $0D	DATA
Lo	Unaddr	$5F	Untalk
Hi			
Lo	Pri	$48	Talk
Lo	Sec	$62	Sec Addr (use 1st data channel)
Hi	Info	$XX ... $XX $0D	Data bytes sent by drive
Lo	Unaddr	$5F	Untalk
Hi			
Lo	Pri	$29	Listen
Lo	Sec	$63	Sec Addr (use 2nd data channel)
Hi	Info	$49 $4E $46 $4F	INFO
Lo	Unaddr	$3F	Unlisten
Hi			
Lo	Pri	$29	Listen
Lo	Sec	$E3	Close (close 2nd data channel)
Hi			
Lo	Unaddr	$3F	Unlisten
Hi			
Lo	Pri	$28	Listen
Lo	Sec	$E2	Close (close 1st data channel)
Hi			
Lo	Unaddr	$3F	Unlisten
Hi			

SPOOLING with COMMODORE DRIVES *by Michael Gilsdorf*

"Command Your Drive to Send Files Directly to a Printer or Another Drive !"

What's Spooling and Why Use It?

The term spooling stands for "Simultaneous Peripheral Operations On Line". It is the process where a disk drive sends a file stored on disk directly to another device such as a printer or another drive. The source drive is responsible for transfering the file over the bus to the device - not the computer. Since the computer is not needed for loading, printing, or copying files, it is free to continue with some other task - provided of course, it doesn't need immediate use of the bus. You may be wondering why Commodore "smart" drives and printers can't do this. Well...they can!

Before we discuss how to do spooling, let's first look at why you would want to use it. When you want to copy a file from one drive to another, the normal technique is to load the file from the source drive into a buffer set aside in the computer. When the buffer fills up, you then write its contents back out to the destination drive. This process is repeated until the last byte of the file has been copied. As you see, each byte is actually transmitted over the serial bus twice; once to the computer and once more to the destination device. The method is simple, but it wouldn't receive a high mark in speed. Although some buses are faster than others, they still tend to be the "slow link" in the file transfer process. If you can reduce your dependency on the serial bus, you should be able to transfer files faster. Another factor which can influence the speed of file transfers is the size of the buffer. Generally speaking, large buffers can provide faster file transfers. On the other hand, large buffers also mean less free RAM for program use.

Spooling can overcome these problems. Since the computer doesn't need to play middleman, no buffers are needed in the computer. Also, since the file bytes are sent directly from one device to another, each byte is sent over the serial bus just once. The computer is still needed to OPEN and CLOSE the devices on the bus, but it is free to perform other tasks while the file is being copied or printed. Suppose you're editing a letter using your word processor, and you realize you need to print out a file that you have stored on disk. Wouldn't it be nice to send a command to the drive, and have it send the file to the printer while you continue editing your letter? Most word processors for the C128/C64 can't handle this. They usually require you store your letter to disk, load the desired file, wait while the file is being printed, then retrieve your letter. Some word processors may allow you to keep your letter in memory, but the computer is still needed for transfering the file from disk to the printer. On the other hand, spooling will allow you to print your file and edit your letter at the same time with no noticeable loss in speed or performance. When you finish editing your letter (and the bus is free), you can then save your letter to disk and make the drive responsible for sending the letter to the printer. In this case the drive could be viewed as a large smart printer buffer. With the proper code it may even be possible to spool a file to a printer and one or more drives at the same time.

How to Spool

Now that you see the advantages of spooling, let's examine how it is done. You begin by opening the source drive and destination device on the bus. For drives, you'll need to include the file name; for printers you may want to include some addition control codes (e.g., set margins, upper case/graphics, etc.). When opening the devices, it is important you use the same secondary address for both. Be careful when assigning the secondary address number. Some numbers have internal meanings to the device. For example, drives reserve 0 and 1 for loads and saves, and use 15 for the command/error channel. Printers (or printer interfaces) usually equate a secondary address to a special font or print style. It's usually best to use secondary addresses between 2 and 14.

After opening both devices, you are now ready for the next step - commanding one device to talk and the other to listen. Unfortunately, it is not as simple as sending a talk and a listen. If you attempt this, you'll discover the DOS routines refuse to cooperate and the devices do not link up. To overcome this problem we need a way to trick the source drive into believing that a listen to the destination device is also a command telling it to talk. Two memory locations in the drive, LSNADR and TLKADR are used by DOS to determine if a talk or listen is meant for it. LSNADR holds the value \$20 + the device number of the drive. Should a listen command appear on the bus, DOS compares it with the contents of LSNADR to determine if it is being addressed. Similarly, TLKADR holds \$40 + the device number. Both LSNADR and TLKADR are initialized when the drive is powered up or reset; however, these locations can be changed by using the "M-W" command. This provides a key to solving our problem. By changing the TLKADR on the source drive to match the LSNADR on the destination device, the source drive will begin talking when it reads a listen command directed at the destination device. In other words, the source drive begins talking when the destination device begins listening!

After sending the listen command, you next send the secondary address. This is used by both devices for channel assignments. Now, all that remains is to have the computer release the bus, and turn it over to the source drive to start spooling. Once spooling is completed, send an unlisten and an untalk, restore the original talk address, and then follow-up by closing the channels. "Spool.printr.bas" is a basic program which demonstrates the basic steps involved for spooling a file from disk to a printer.

Before you run "Spool.printr.bas" there are a few things of which you should be aware. First, the program will not operate with JiffyDOS activated. If you have JiffyDOS, you must switch it off before running the program. Second, the program is drive dependent. You'll discover unless the drive is a 1541, spooling will most likely fail. The reason for this can be found if you remember that both the 1571 and 1581 drives can communicate on the serial bus in either of two modes; standard and burst. Since a printer can only respond with standard communication, any attempt by the drive to use its burst routines to communicate with the printer will cause the bus to lock up. Fortunately, Commodore provided a means to allow us to specify the mode of operation for these drives. Unfortunately, however, the commands used to specify the mode of operation are different for each drive. It is not clear why Commodore did this, but we can program around it, nevertheless. The Mode of Bus commands are "U0>M0" and "U0>M1" for the 1571, and "U0>B0" and "U0>B1" for the 1581. The M0 and B0 specify standard mode while M1 and B1 specify burst. Whenever the source drive is a 1571 or 1581, we must tell it to "slow down" when communicating to a printer (or 1541 drive). After the process is completed, we should tell the drive to return to burst mode. Now, armed with this knowledge we can correct our spooling program so it will work with either a 1571 or 1581. Add two of the following lines to the program "Spool.printr.bas":

```
27 print#1,"u0>m0" : rem use if 1571 is source drive
27 print#1,"u0>b0" : rem use if 1581 is source drive

93 print#1,"u0>m1" : rem use if 1571 is source drive
93 print#1,"u0>b1" : rem use if 1581 is source drive
```

Another, more academic problem with the above program, is it does not automatically terminate the transfer process. By that I mean, it does not include any code for checking to see if the drive has completed transfering the file to the printer. Instead, it relies on the user to terminate the spooling process by pressing the <return> key. This may be ok for printing since we can easily see when the printing is complete, but it would not be satisfactory for drive-to-drive transfers. In the latter case, we really do not have a reliable indicator showing the file transfer is complete. If you accidently terminate the process too soon, you will end up with only a partial copy of the file - not very desirable. Obviously, if you were doing spooling within a program, you would want to have the process end automatically.

Two solutions to this problem come to mind. Perhaps you can think of others. The first involves having the computer read the bytes at the same time the destination device is reading them. When the computer reads the byte with the end-of-information (EOI) marker, the computer knows the last byte was sent, and can close the channels. Although this method does not free the computer for other tasks, it still does not require buffers or the need to send each byte over the bus twice. You might employ this method for an application whose only purpose is to copy or print files.

A second method might be to have the computer periodically check the bus for activity while it is performing some other task. After the EOI byte is sent the bus will remain inactive and you can read for this condition to determine when to terminate. Careful thought should be devoted to the bus checking routine to ensure that it correctly identifies an EOI condition (inactive bus). Be aware that the bus can sometimes go inactive for a short period of time while one device is printing or reading/writing to disk. Also, you might consider patching the bus checking routine into the computer's IRQ interrupt sequence so that it runs in the background and appears unnoticed to the user. If coded carefully, it should not interfere with the computer's ability to perform its primary task.

The second program "spool.drive.bas" is written in BASIC and allows the use of multiple configurations of drives and printers. Although a little longer than the one above, this program includes the following features:

1. Runs in either the C128 native mode or C64 mode (Be sure to type it in while in C128 mode).

2. Performs spooling between any two Commodore drives(1581/71/41), or between any Commodore drive and a printer.

3. Checks for errors.

4. Terminates automatically by checking for an inactive bus.

Try out spooling for yourself. Maybe you can apply it to enhance some of your own programs. If you have any questions or comments, I can be reached on Q-Link under the name "MIKE ALL" or under GEnie under "M.GILSDORF1". Until then, easy DOS it!

Note 1: For the program below, the drive-to-drive transfer "should" work with JiffyDOS enabled - provided both drives are equipped with JiffyDOS. If this is not the case, or if the transfer is between a drive and a printer, switch Jiffydos off before running the program. Also, be sure to disable any other fast loaders before running the program.

Note 2: Be sure to set the beginning lines of the program to correspond to your system and the type of spooling you wish to do (drive-to-drive or drive-to-printer). For 1581 drives you can specify a partition if you wish. If you are using an interface with a non-Commodore printer you may be able to send commands for specifing fonts, margins, etc. If not, set them equal to a null string (=""). Also be sure to use a secondary address which provides the desired output. Check your manual(s) if in doubt.

Spool.printr.bas

```
jk 5 rem c128 spool drive-to-printer
fh 10 a=11      : rem device no. source drive
ad 15 b=4       : rem device no. printer
gk 20 sa=0      : rem secondary address
fa 25 open 1,a,15 : rem open command channel on source
al 30 open 2,a,sa,"0:readme.ltr,s,r" : rem open source
    file for reading
dd 35 open 3,b,sa  : rem open printer
hk 40 tk=2*16 or b : rem listen address of destination
ll 45 print#1,"m-w"chr$(120)chr$(0)chr$(1)chr$(tk): rem
    change address
of 50 s=6*16 or sa : rem secondary address command byte
pl 55 sys 58174,b  : rem send listen
ae 60 sys 58578,s  : rem send secondary address
    command byte
cj 65 sys 58687    : rem release bus (data & clock
    lines high)
gg 70 input "press <return> when done printing";a$
hn 75 sys 58662    : rem send unlisten
dc 80 sys 58645    : rem send untalk
ad 85 tk=4*16 or a : rem original talk address of source
oe 90 print#1,"m-w"chr$(120)chr$(0)chr$(1)chr$(tk) :
    rem restore address
ip 95 close 2:close 3: close1: rem close channels
```

(See page 34 for Spool.drive.bas)

(Editorial - continued from page 28)
and C-64 is not that they are bad machines or they can no longer function or perform the tasks we required of them in 1985. It is just that some people require more of them in the 90's. It is for these people (and myself ;) that we are producing the ZIP board and running these articles.

Okay, so what is on the platter for the upcoming years besides these items? Starting with issue #32 we will be carrying a regular CP/M, Geos, and Basic 8 column. Why? Because we couldn't fit them into this issue and to help you wring every bit of productivity and enjoyment out of your present software and hardware was possible.

Timely News and Reviews

We have hit you with the MPS-1270 and Unix reviews before anyone else. Now see KeyDos and System Six! Even more new stuff in issue #32. Our goal isn't to wait until information comes to us but to search it out and publish it! There is a ton of software out there for C-128 users, it just is not clearly visible to most people. So we seek, we find, we publish! If you are a software author or software company and you can't be seen, send us your material to review, to publish, to use. You will not get ignored here. We will change the way you think about Commodore software and magazines by encouraging change and advancement. It will not be easy but it will be enjoyable! These will be the best years yet for C-128 software and hardware and I am looking forward to it!

Utilizing What You Have

One thing people have wanted for GEOS is a program like geoPager. This article actually took the place of the scheduled Geos column by Robert Knop because I was crammed for room. I know there are a lot of Deskjet and Laserjet owners, newsletter editors, and small publishers dying for this type of program. Here it is.

Questions and Comments

Q: TC128 should be on better paper?

A: It "should" be, but it will not be. The simple facts are that this type of newsprint costs exactly 1/2 as much as 40lb wrap (printer's term for copy paper). It also weighs less then 1/2 that of 40lb wrap. This means much lower printing costs, shipping costs, and a lot less storage space used. Though there are lower grades of newsprint I feel that is looks pretty good and suffices for our needs, which is getting the news out.
(continued on page #39)

Spool.drive.bas

```
kb 100 rem c128/c64 spool drive-to-drive/printer
fh 110 rem by michael gilsdorf (c) july 1990
oe 120 f=abs(peek(65533)=255): rem f=0 if c64; f=1 if c128
jg 130 if f=1 then cl=peek(215): slow: if cl=128 then fast :
       rem 80 col chk
jk 140 rem          valid device names
km 150 data 4, "1581", "1571", "1541", "printer"
ng 160 f$="0:filename"        : rem source filename
fj 170 t$="seq"               : rem source file type
mo 180 a$="1571"              : rem source name
fk 190 a=8                    : rem source device no.
di 200 c$="i0"                : rem source drive command
am 210 :
bj 220 b$="1581"              : rem destination name
ln 230 b=9                    : rem destination device no.
de 240 d$="/partition"        : rem destination drive command
de 250 :
lk 260 pi$="u"                : rem printer interface command
di 270 p$=chr$(27)+chr$(64)   : rem printer escape codes
il 280 sa=7                   : rem secondary address
pd 290 rem check for errors and display settings
lm 300 r$=chr$(18): print chr$(147);r$;"c128/c64 spool drive-to-
       drive/printer"
fd 310 print "by michael gilsdorf     (c) july 1990": print
kj 320 read k: k=k-1
md 330 for j=1 to k: read n$: if a$<>n$ then next: print "error
       : source name": end
ad 340 j=k: next: restore: read k
ol 350 for j=1 to k: read n$: if b$<>n$ then next: print "error
       : destin name": end
gk 360 j=k: next
im 370 if a=b or a<8 or a>15 or b<4 or b>15 then print "error
       : device no.": end
ld 380 if b$<>"printer" and b<8 then print "error
       : device name or no.": end
ba 390 if b$="printer" and b>7 then print "error
       : device name or no.": end
pk 400 if sa<2 or sa>14 then print "error: secondary address": end
kh 410 open 1,a,15: close 1: if st<0 then print "device";a;
       "not present": end
nb 420 open 1,b,15: close 1: if st<0 then print "device";b;
       "not present":end
hj 430 if c$="" then c$="i0"
in 440 if d$="" then d$="i0"
ee 450 fs$=f$+","+t$+",r": fd$=f$+","+t$+",w": if b<8 then d$=pi$
       : fd$=p$
mc 460 cc$="i0": cd$="": if b>7 then cd$=cc$
nn 470 if a$="1571" then cc$="u0>m0"
bd 480 if a$="1581" then cc$="u0>b0"
aa 490 if b$="1571" then cd$="u0>m0"
dg 500 if b$="1581" then cd$="u0>b0"
lp 510 if a$="1571" and b$=a$ then cc$="u0>m1": cd$=cc$
le 520 if a$="1581" and b$=a$ then cc$="u0>b1": cd$=cc$
lo 530 if a$="1581" and b$="1571" then cc$="u0>b1": cd$="u0>m1"
```

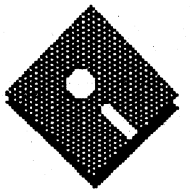
Spool.drive.bas

```
eo 540 if a$="1571" and b$="1581" then cc$="u0>m1": cd$="u0>b1"
ha 550 print r$;"check program settings"
el 560 print: print "source drive:"
he 570 print "device:   ";r$;a$,a
if 580 print "command:  ";r$;c$
ln 590 print "filename: ";r$;f$,t$
mj 600 print: print "destination device:"
md 610 print "device:   ";r$;b$,b
lp 620 print "command:  ";r$;d$
ga 630 if b>7 then print "filename: ";r$;f$,t$
hj 640 if b<8 and p$<>"" then print "code(s):  ";: l=len(p$)
do 650 if b<8 and l>0 then for j=1 to l: print r$;
   asc(mid$(p$,j,1));: next: print
cf 660 print: print r$;"disable jiffydos if both devices do not"
cf 670 print r$;"have jiffydos installed (e.g., printer)"
jg 680 print: print "press <stop> or <return>"
gm 690 get x$: if x$="" then 690
if 700 print chr$(147);"opening devices..."
bk 710 n=1: open n,a,15,cc$: gosub 1020: print#n,c$: gosub 1020
eb 720 open 2,a,sa,fs$: gosub 1020
ge 730 n=3: open n,b,15,cd$: gosub 1020: print#n,d$: gosub 1020
mh 740 open 4,b,sa,fd$: gosub 1020
ck 750 :
aj 760 rem change talk address; send listen,
       secondary address & release bus
ip 770 tk=2*16 or b: m=120
ma 780 print#1,"m-w"chr$(m)chr$(0)chr$(1)chr$(tk)
am 790 s=6*16 or sa
bg 800 if f=1 then sys 58174,b: sys 58578,s: sys 58687
gf 810 if f=0 then poke 780,b: sys 60684: poke 780,s: sys 60857
       : sys 60941
hb 820 :
oh 830 print chr$(147);"wait...transfer in progress": fa=0
oc 840 print ".";: io=peek(56576) and 192: if io<>192 then ct=0
al 850 ct=ct+1: if io<>lt then fa=0: lt=io
gb 860 fa=fa+1: if ct<15 and fa<30 then 840
ho 870 if fa>74 then print: print
       "*** transfer may have failed ***"
jl 880 if fa>74 then print
       "***     disable jiffydos     ***"
lh 890 :
fa 900 rem send unlisten, untalk, close, &  restore talk address
eo 910 if f=1 then sys 58662: sys 58645
if 920 if f=0 then sys 60926: sys 60911
ff 930 tk=4*16 or a: print#1,"m-w"chr$(m)chr$(0)chr$(1)chr$(tk)
gi 940 for j=0 to 1: n=j*2+1: gosub 1020: close n+1: next j
ik 950 n=1: if cc$="u0>m0" then print#n,"u0>m1": gosub 1020
eg 960 if cc$="u0>b0" then print#n,"u0>b1": gosub 1020
kl 970 n=3: if cd$="u0>m0" then print#n,"u0>m1": gosub 1020
fn 980 if cd$="u0>b0" then print#n,"u0>b1": gosub 1020
og 990 close 3: close 1: end
cf 1000 :
hg 1010 rem read error channel
```

(continued on page 36)

```
Spool.drive.bas
ap 1020 if n=3 and b<8 then return
ik 1030 input#n,e,e$: if e<20 then return
kd 1040 print e$
hc 1050 if n=1 then print r$;a$,a: print r$;c$
ko 1060 if n=3 then print r$;b$,b: print r$;d$
bf 1070 print r$;f$,t$
bj 1080 for j=1 to 4: close j: next j
   (end of spool.drive.bas  program)
```

Top-Quality Computer Supplies at Rock-Bottom Prices!



5-1/4" Double sided, double density disks by 3M. Tested and certified on both sides. Perfect for 1571 and 1541 drives. Big box of 25 - **$15.95** ($1)

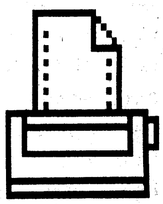
5-1/4" Double sided, high density disks by Maxell. For use with SFD and drives that use high density disks only. Box of 15 - **$16.95** ($1)

3-1/2" Double sided, double density disks by 3M. Great for 1581 and other 3-1/2" disk drives. Big box of 30 - **$28.95** ($1)

3-1/2" Double sided, high density disks by Maxell. Use with high-end Macintosh and some MS-DOS systems. Not compatible with 1581. Free plastic carrying case included (while supplies last). Box of 15 - **$26.95** ($1)

JD's Computer Supply

P.O. Box 873, Pearl City, Hawaii 96782-0873
Our shipping system runs on a Commodore 128d.



Mailing labels by Avery. Our continuous tractor-feed are great for mailing labels, but have many other uses as well. Smudge-resistant. Box of 5000 - **$15.95** ($4)

Laser Printer Labels by Avery. 100 8-1/2 by 11" sheets with 33 labels per sheet. Great for use with copiers also! Label size 1" by 2-3/4". Box of 3300 - **$19.95** ($2)

Ask for Our Free Catalog!

Ordering Information

Thank you for your order! Your shipment should arrive 10-15 working days after we receive your order. In the event an item is out of stock temporarily, we will place the item on backorder and notify you immediately. Please allow up to 30 days for shipments of backordered products. We immediately ship orders paid with money orders, cashier's checks, and CODs. Please an additional 10 working days for personal and company checks to clear. Sorry, we do not accept credit cards. Repeat customers - no check holding if order is under $75. Shipping charges for each item are in parenthesis next to the price. Please make sure you include shipping charges or we cannot process your order. We ship using USPS Priority Mail or UPS. Never a surcharge for shipments to Alaska, Hawaii, Guam, Puerto Rico, or APO/FPO! We can ship 2nd Day Air or next-day for additional cost. Inquire for shipping charges using these services.

Place your COD orders by mail (send to address on left) or use E-Mail for faster service. Send to "JD.COMP" on GEnie and "JD Supply" on Q-Link. Everything we sell is covered by our 100% Total Satisfaction Guarantee. Just return your item within 30 days and we'll be happy to exchange it or refund your money - your choice. Full details of our guarantee are included with each shipment. We cannot be responsible for typographical errors.

geoPager *by Robert A. Knop, Jr.* "The geoPaint Batch File Printer"

I. Introduction

Printing with GEOS is a slow process. This stems from the fact that any printout (save draft or NLQ printouts from geoWrite) is a graphics dump. Your 128 must convert the graphic bitmap of geoWrite or geoPaint into the format readable by your printer, and then send the data byte by byte, eight pixels at a time, to the printer.

With geoWrite, if you have a many page document, you can start a print job, and then leave your computer alone for the time it takes it to complete the job. However, with geoPaint, since each document is one page long, you have to baby sit the computer if you want to print many pages; each new page must be started manually. geoPager is designed as a means of avoiding the ensuing tedium. It allows you to create a list of geoPaint files, and then print out all of those files in one operation- giving you the opportunity to go get a cup of coffee, go for a jog, take a vacation overseas, etc., while you are waiting for the printout to complete.

II.System Requirements

geoPager is designed to run under GEOS 2.0 on a Commodore 128 in either 40 or 80 columns. (There is a separate version for GEOS64.) It will run with the same base system requirements as GEOS128: a 128 or 128D, an input device, a monitor, and one disk drive. However, as with anything in GEOS, operation is much faster if you have a RAM device of some sort; also, the purpose of geoPager is somewhat defeated if you don't have a printer. geoPager can read geoPaint documents from any disk drive in your system (drives A through D, i.e. units 8 through 11). It should be able to correctly print geoPaint documents with any GEOS compatible printer driver.

III.Use of geoPager - Overview

With geoPager, you first create what is known as a "Paint Batch File" (PBF, or geoPager document). This file is simply a list of geoPaint filenames. geoPager saves this list, so that if you later want to print files from the same list, or a slightly modified version of the same list, you don't have to rebuild the list from scratch. The list is in memory so you can then print out either all of the files in the list or a subrange of the files. Additionally, if you like, you can make multiple copies of each file, or you can cycle through the list more than one time, all in one operation.

IV.Operation of geoPager

Figure 1 shows the geoPager screen. This section will take you through the elements of the screen, as a way of explaining how get geoPager to work. Note that unless you directly double-clicked on a geoPager document, before you see the screen depicted in Figure 1, geoPager will present you with the familiar "Create/ Open/ Quit" dialog box, asking you to select the geoPager document with which you wish to work.

(1) <u>geoPager document name</u>: This box displays the name of the PBF (geoPager document) currently in memory.

(2) <u>geoPaint file list</u>: This box contains the names of the geoPaint files in your current Paint Batch File. The box only displays twelve geoPaint filenames at a time; geoPager allows your list to contain up to 255 geoPaint filenames. The files are numbered, so you can tell how far you are from the top of the list. (Additionally, these numbers are used as a reference when selecting which files to print.)

(3) <u>Scrolling icons</u>: These icons allow you to scroll through your list, to look at different sections of it. From left to right, these icons are: up one line; down one line; up one page; down one page; top of list; bottom of list.

(4) <u>Control Icons</u>: These icons allow you to issue commands to geoPager. Their functions will be described in order:

<u>Move</u>: This allows you to move a geoPaint filename to a different position in the list. Select the filename to move by clicking on it, then press Move. The name of the file being moved will be displayed in the Status Window (5). Use the Scrolling icons until you see the position to which you want to move this file. Click on the filename at this position. The moving file will be inserted before the filename on which you click.

<u>Insert</u>: This allows you to add a new geoPaint file to your list. The file will be added before the currently selected (highlighted) filename. When you click on Insert, you will see a dialog box displaying the names of all of the geoPaint files on the current disk (see Figure 2). This dialog box can display the names of up to 255 geoPaint file on one disk. The Open, Disk, Drive, and Cancel icons function just as they do in other GEOS applications like geoPaint or geoWrite. The scrolling icons act on the list of disk files just as the corresponding icons in Figure 1 do on your geoPaint filename list.

Although reading files from a RAM drive is fast, you will notice that it takes GEOS quite a while to find all of the geoPaint files on a full disk. For this reason, the directory is buffered between insertions. What this means is, if you want to insert several files from the same drive, GEOS only has to search through the disk for geoPaint files the first time you click on Insert. If you immediately click on Insert thereafter, the directory of geoPaint files comes up quite quickly.

A note about inserting files from multiple disks: Although geoPager allows you to change disks and insert geoPaint filenames from many different disks, it is not generally a good idea to do so. When you do finally print out all of the geoPaint files on your list, unless all of disks containing these files can be in drives on your system at once, you will have to sit at your computer and swap disks when necessary. In other words, if you have only one 5.25" disk drive, it is best not to put geoPaint filenames from multiple 5.25" disks in the same Paint Batch File.

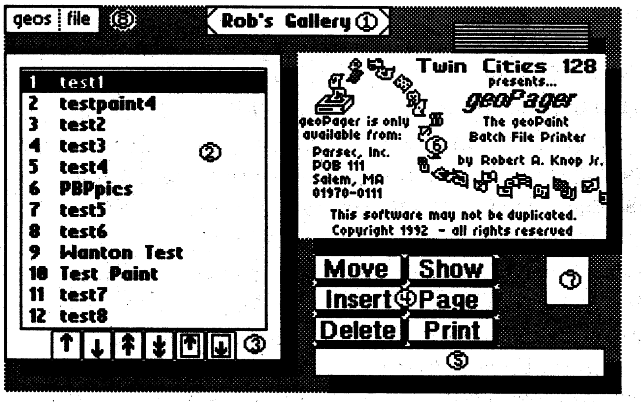
<u>Delete</u>: This icon deletes the currently selected file from your file list. Following file names are moved up to fill the gap.

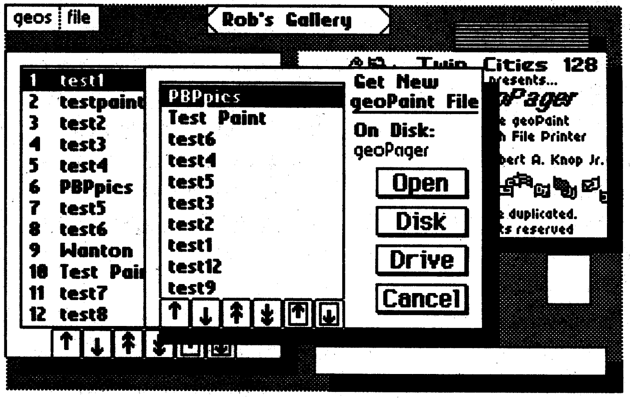
<u>Show</u>: This allows you to take a look at the currently selected geoPaint file in the Show Window (6). With the Show Position icon (7) you can look at different parts of the page. The Show Position icon functions much as the similar icon in geoPaint. The Show Window (6) contains the geoPager title screen when no geoPaint file is being shown.

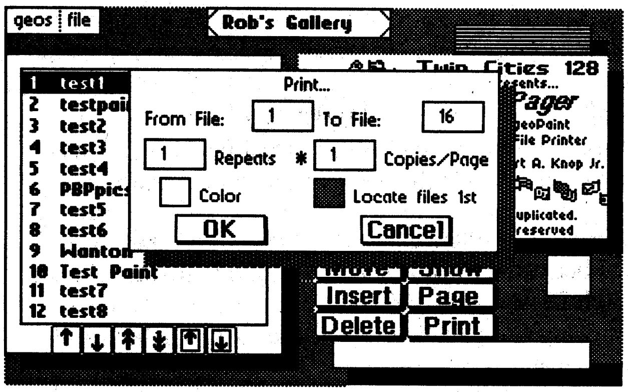
<u>Page</u>: This icon prints out one copy of the currently selected geoPaint file.

<u>Print</u>: This icon brings up the Print Options dialogue box (see Figure 3) which allows you to print out some or all of the files on your geoPaint file list. On the Print Options box, "From File" and "To File" select the numbers of the first and last geoPaint files from your list to be printed. They default respectively to the first and last files in your list. "Repeats" is the number of times for geoPager to cycle through your list when printing. "Copies/Page" is the number of times to print each geoPaint file each time through the list. So, if you selected 2 repeats and 2 copies/page, geoPager would print each document four times, twice each time through the list. "Use Color" selects whether geoPager should attempt to print the files in color, or black and white.

<u>Locate Files 1st</u> is a safety measure. If you don't select locate files 1st, and during the printing geoPager can not find one of the geoPaint files in your list, it will stop and ask you what to do. If you do select "locate files 1st", geoPager will look for all the files and ask you what to do with ones it can not find *before* the long printout process begins. In either case, when geoPager can not find a file, it gives you three options: Retry, Skip, or Cancel. Retry is only useful if you do NOT locate files first. It allows you to put a different disk in one of your drives, and let geoPager look again for the file. Skip skips the current file, and continues with printing on the next file. If you select "Skip" during a "locate files 1st," that file will be ignored during the







printing process. Finally, Cancel cancels the print job.

V.Menu Options

There are two menus at the top of the geoPager screen (8): geos and file. Under geos, options are "geoPager info" and any desk accessories you have on your disk. These are self explanatory. Under file are the traditional four options, Close, Update, Recover, and Quit. Briefly, Close saves your current PBF and allows you to work on another. Update saves the current PBF, but does not close it. Recover reloads the current PBF from disk, restoring it to its state when you first loaded it (or the last time you updated it). A dialog box makes sure you really want to recover before you do so. Finally, Quit saves the current PBF and exits geoPager.

geoPager is one of the programs available on the companion disk to Twin Cities 128 issue #31.
If you are not subscribing to TC128 with the companion disk, this disk is available seperately through our Parsec catalog for only $9.95 plus S&H.
Disk code TC128 #31

(EDITORIAL - continued from page 33)

In case you have not noticed the usual TC128 was 20-24 pages and all of us gladly shelled out the dollars for it. With the money I am saving on reduced costs I will try to keep it at 32 pages. I think most people would rather have timely news and more of it then better paper. As a matter of fact I think this is the 1st TC128 to ever reach 40 pages!

Q: What is a "Parsec"?
A: I bet you never played "PSI 5" and that you do not watch Dr.Who or Star Trek (just goshing). Besides being my company it is also equal to 3.26 light years. If you want to get real tongue tied look up the full meaning in the dictionary.

Q: What is "lweir" and "liteweir"?
A: Actually they were names I tossed around for the company back in 85 and I use them for trademarks on my software products. Lite(Parsec)weir, (softweir - hardweir), Irish(weir). Now you know.

Q: What is the ZIP board?
A: The ZIP board is an accelerator board that will be designed by a company that makes accelerator boards for Apple II products. The basic specs on the board so far are:

The board will speed up how fast the C-128 runs up to 4 times!

The board will provide a method to slow it down to 2 or 1 Mhz when needed.

The board will run at 8Mhz in C128 mode.

The board will run at a fast speed in C64 mode too.

It will be installed internally on the flat C128 and C128D

It will work with CMD products

The memory will be user expandable on the board.

The board will support up to 512k or 1meg of ram using the banking methods published in TC128 so it will be compatible with Basic 7.

It will sell for under $200.

It will be ready in 8 months once we start.

I expect to start by the end of Feburary 1992. People that have sent in the survey will be sent a form to send in along with their $50 deposit. The supplies are going to be limited on this board (about 1500) so if you want to be guaranteed a board you better fill in your survey and send it in now.

Q: Who publishes Dialouge 128 now?
A: Triple Point Software, see their ad in this issue.

Q: Will the Kraft Triple Track trackball work on my C-128?
A: Funny you should ask since I got a review unit a couple of months ago. The answer is *maybe* it will. It will only work on the C-128 in joystick mode. Although it appears to be nicely built and a real gee-whiz kind of product I could not get it to function properly in joystick mode on either a C-128D or flat C-128. I would not risk $99 on something that might not work properly and even then only works in joystick mode. If you want a trackball to use with the older paint programs try an Atari trackball. I had a black one with the yellow ceramic ball and it worked fine on my C-128. Call the Triple Track review the review that wasn't.

Comment: on pg 40 is a TC128 endorsement of geoMatrix. "Do I make GEOS users aware of a excellent, although competitive, product?". You buy this magazine to be informed and I believe there is always enough room in our market for another excellent product.

GEOS SUPPORT GROUP by Grady Brown

I'd like to thank Twin Cities 128: The COMMODORE 128 Journal for this opportunity to tell you of a support group for GEOS users. geoMETRIX GEOS Users Group is a group made up of GEOS users & programmers as well as those that would like to learn to use GEOS.

GEOS is an operating system that stands for Graphic Environment Operating System and is much like the Macintosh. Disk commands are given by pointing and clicking your mouse or joystick on graphic icons instead of typing out filenames.

A few programming packages have been written for those that prefer to write their own programs. GeoBASIC and GeoProgrammer are two commercial packages. StudentForth and Brian are two Public Domain languages.

With all these programs being written for GEOS and all those that are using it, a need for an informational exchange appeared. That is when geoMETRIX GEOS Users Group and the geoJOURNAL came on the scene. geoMETRIX, going on two years old is a user group consisting of occasional users and die hard users from beginners to experts as well as programmers.

geoMETRIX GEOS Users Group is a membership group with members all over the world. Only a small handful are local, and can attend the monthly meetings. For this reason the geoJOURNAL is a mainstay for most members. You can also subscribe to the geoJOURNAL, which is a bi-monthly publication that goes out to subscribers and geoMETRIX members all over the world.

Membership in geoMETRIX is $10.00 per year for those residing in the United States. $12.00 per year in Canada and $20.00 per year for those in other countries. Members receive a One Year subscription to the geoJOURNAL (six issues), access to our Support BBS, special 'Members-Only' offers and access to the geoMETRIX Public Domain Library Disk Collection.

Subscriptions to the geoJOURNAL are for those that don't want all the extra frills that membership includes. The geoJOURNAL is packed with informative Feature Articles and regular columns for beginners to experts on many different topics within GEOS. Subscriptions are One Year (six issues) and run $5.00 in the United States, $6.00 in Canada, and $12.00 for other countries. A single issue may be purchased for $1.00.

If you are interested in learning more about the GEOS system or new tips and short-cuts to do everyday things with GEOS then you should join as a member of geoMETRIX GEOS Users Group or subscribe to the geoJOURNAL. If you are one who prefers to lead or teach others, then please write for more information and geoMETRIX will find a place for you in the geoJOURNAL to help others. What ever your reason, join today, and get more enjoyment out of GEOS.

Grady Brown, author of the this article, is the editor of the geoJOURNAL and lives in Camas, WA. He is also an active local member of geoMETRIX.

geoMETRIX GEOS Users Group
20224 S.E. Spague Road
Oregon City, Oregon
97045-9641

```
**************************************************
*                   CLASS(Y) ADS                 *
**************************************************
```

These ads are free for TC128 subscribers and advertisers. Commodore BBS and UG listings are free to all.

All people with POBs have to submit a street address to Parsec with a matching night time telephone number (we will not release the street address to anyone UNLESS there is an unresolved problem with your ad). For sale and wanted ads must include either an address (street or POB) OR a telephone with the time to call. An example is, (1-508-745-9125 EST 9-5 answering machine) so people can easily contact you.

All ads will run until you ask for them to be removed or until they are "bumped" off the listing by a newer ad. The date the ad was 1st run will be expressed as 920105 (year 92, 1st month, fifth day).

ALL ADS *MUST* be submitted on either 1541 or 1581 disks as either PetAscii or straight Ascii sequential text disk files, no exceptions! If you can send matching hardcopy it would be appreciated. We will take ads from subscribers through e-mail. We are no responsible for anything including typos. BUYER BEWARE!

The guidelines to buying through the mail are unless you know the person well:

1) Buyers and sellers should insist on COD, ship by UPS (if possible), cash or money order!
2) Get a telephone number!
3) Try to have some fun horse trading! :)

FOR SALE HARDWARE

*920201 - Clay MacDonald 303-927-4498
C128D, JD V6.0, fan, all docs, like new $300.00, 1581, JD, like new $100 - both units have device switches. QBB 64 - $50, 1764(512k) - $100, Many other items

*920201 Alex Dundek 612-645-6636
1571 - new in box - best offer over $150

FOR SALE SOFTWARE

WANTED - SI/FI - MISC

BBS LISTINGS

USER GROUP LISTINGS

*920201 - Basic Bits Commodore Group, PO Box 447
PO Box 447, North Ridgeville OH 44039

Zip Accelerator Board release form - fill out and mail

Name:
Address:
City:
State:
Zipcode:
Phone number:
Network:
Online address:

I (the above named person) agree to give Parsec, Inc P.O.Box 111 Salem,MA
-------------------------------------- 01970-0111 a $50 deposit towards the purchase price of an accelerator speed up board and/or chip for my C128/128D. This will be a 8Mhz speed up board. Parsec, Inc. can keep this deposit for a period up to 8 months from the time of deposit while developing the board. Parsec, Inc agrees to sell this board to you for $200 or less and that it will be software compatible with the majority of software and hardware including CMD products.

DO NOT SEND ANY MONEY NOW!!!!!

If you agree to the above terms just sign in the above space and return this form to our U.S. mail address. You will be contacted later by US mail or E-mail, I prefer a GEnie mail address!!!

Please print your name in the above spaces, use a typewriter, or use printer labels.

IF YOU HAVE ALREADY SENT IN YOUR SURVEY DO NOT SEND IN THIS SURVEY, YOU ARE ALREADY REGISTERED

Due to the amount of mail we will be recieving please do not expect a personal reply at this time! Fill this form out if you are willing to give me (Parsec, Inc.) a $50 deposit at a later date.

Zip Technology (see a review of their products in the June 1991 issue of inCider - the Apple II magazine) has agreed to develop the board if certain conditions are meet. Parsec, Inc will meet those conditions and market the board when it is finished.

For those of you who don't know me or have not dealt with Parsec,Inc. we are a mail order company and have a strong online and market presence. We sell both hardware, software, and public domain software. We have a good record with the MA BBB (no complaints at all) and have helped promote the C64/128 in many ways (Example: 8BUSA and the WoC 1990). Until Oct 1990 we also advertised in RUN magazine for about two years.

Cut here and save for your records

Parsec, Inc.
P.O.Box 111
Salem,MA 01970-0111
U.S.A.

Genie = C128.JBEE (on everyday)
CIS = 70661,443 (on once in a while)