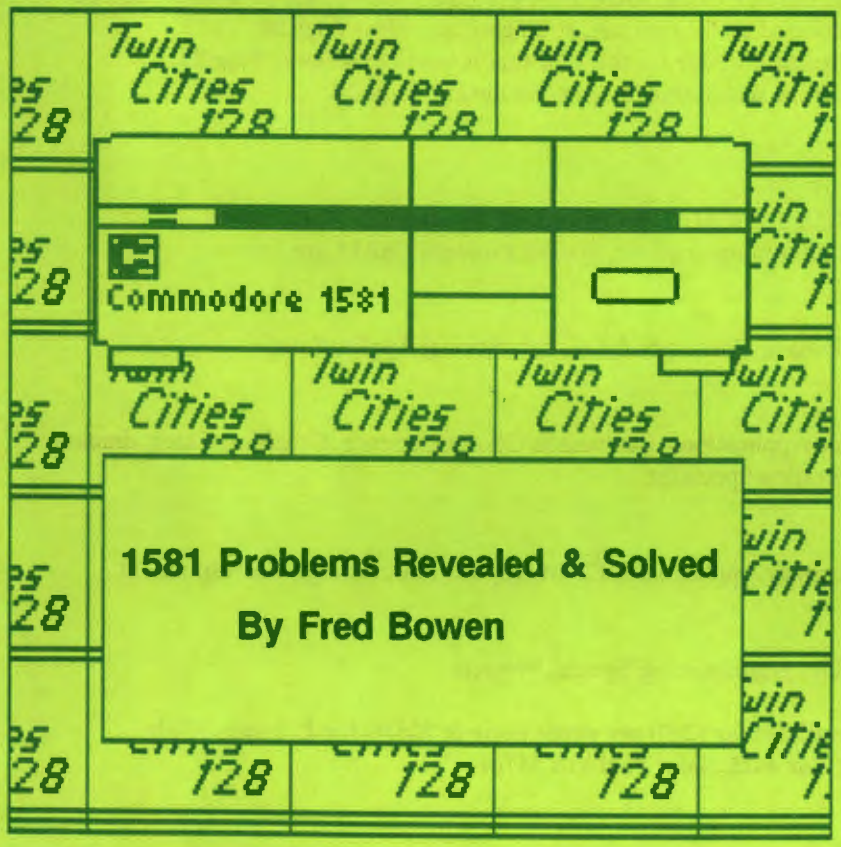
TWIN CITIES 128

THE COMMODORE 128 JOURNAL

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



North America's only C-128 specific publication

Issue #19 $2.50

Rumor / Opinion / Mayhem

C-128 Price & Progress Report

Evaluations of:
- **PaperClip III**
- **Colorez 128**
- **RUN Productivity Pack III**
- **CP/M COMAL**
- **1541/1571 Drive Alignment**

Adding a Fan to the C-128D

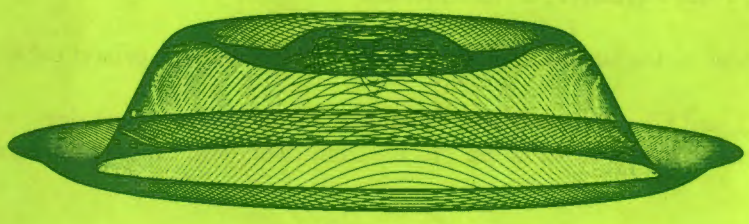
BASIC 8 Backgrounds

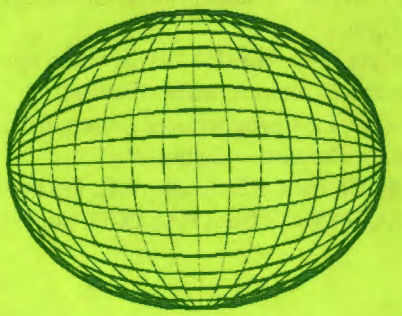
C-128 Public Domain Software

Sparrow's Slick Tips

The Assembly Line

The Back Page





TWIN CITIES 128: THE COMMODORE 128 JOURNAL

North America's only C-128 specific publication, Issue #19, Copyright 1988

TABLE OF CONTENTS

STAFF

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

Avonelle Lovhaug, Associate Editor
Productivity Applications, Proofreading and Layout, Business & Correspondence, Educational Applications

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

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

John Kress, Staff Writer
BASIC 7.0 and Assembly Language Programming, Graphics programming, Special Projects

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

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

I can't tell you how many times I have heard the assertion, **"Today, you do not need to be a programmer in order to use a personal computer."** Whether this statement emanates from your local computer guru, a computer salesperson, or in an article from a popular magazine, the intent for making such a declaration seems to be aimed at calming the fears of those who feel they cannot utilize the wonders of personal computing without a degree in computer science. But is it true that you don't need to know how to program to effectively use a personal computer?

While I agree that the advent of affordable microcomputers and sophisticated commercial software during the 1980s has elevated computers from the realm of the mathematically gifted and the halls of the scientific elite, I take great deference to the premise that today's microcomputers, no matter how sophisticated, have lifted us beyond the "user-programmed" stage.

Before you self-proclaimed "non-programmers" start forming a lynch mob, let me borrow some more of your time to formulate my arguments. One of the definitions given Webster's New World dictionary for the term "program" is: "A logical sequence of instructions performed by a digital computer in solving a problem or processing data." A "programmer" is of course, "One who programs." While this definition indeed may sound a tad intimidating, in reality no matter what activity we perform with our Commodore 128's or any microcomputer for that matter, we are in effect, by definition, "programmers".

What I am suggesting is that every interactive activity, from using the simplest of computer games to the most sophisticated productivity applications, is indeed "programming" the computer. Think about it. No matter how "user-friendly" the documentation or packaging for a program claims to be, if it interacts with the user in any way, it most certainly requires guidance through a logical sequence of instructions in order to operate. This theorem holds true for every type of computer program, except those where the program's sole purpose is to turn the so-called "user" into nothing more than a spectator. *(On a side note, every time I am assaulted by some well-meaning soul who wishes to "convert" me to so-called advanced userdom by trying to convince me that I need an Amiga, I am always ordered to gaze at a screen where I am supposed to bow down before some rotating Greek-god image or robotic specter flinging mirrorred spheroid objects into the air. I am told by my friends who are involved in this cult that there are thousands of such "demonstrations" in Amiga public domain libraries that many Amy users spent a great deal of time admiring and worshipping. My question is: What motivates these people to spend thousands of dollars and hours of time to become passive mental slaves to these machines? How can we help them? Perhaps we should ask the experts in mind-numbing by machine, the television networks.)*

After all, what separates choosing items from a menu or pointing and dragging icons across a screen with a mouse from laying out ordered sequences of instructions in BASIC or Pascal is nothing more than the metaphor. Is the creation of formulas in a spreadsheet or the formatting of a document in a word processing program really any different than creating code in a "traditional" programming environment? I assert that if we use our computers in any interactive manner we are indeed programming them. The only difference between the computer programmer creating Pascal code and the computer programmer creating word processing "code" is the language and confines with which the programmer is working. And in the case of many sophisticated applications, the language is indeed as rigorous and complex as any traditional programming environment.

If the concept submitted in the previous paragraph seems profound, I suggest that computer industry and computer professionals are partially to blame for our lack of vision and understanding of what we are really doing when we use our machines. The title "programmer" has, since this industry's inception, been revered; reserved for only those who excel in the "traditional" discipline of computer programming. In paying homage to the abilities of these traditional programmers the industry likes to set them on a pedestal. Sometimes this attitude is quite overt; for instance, computer magazine articles which portray the creators of commercial software as heros and legends for us to look up to and admire. At other times this notion manifests itself quite subtly, for instance advertisements from software companies that portray their products as "easy to learn and use, perfect for non-programmers". However, in the context of this discussion that means that these products are programs which don't really do much, which by the way I have found in most cases to be more or less true.

Let's be honest with ourselves and others when we describe what it takes to really be a personal computer "user" rather than a personal computer "watcher". Even in the best case scenario it takes hours of practice and study, and yes, in one form or another, you will be required to learn some kind of computer programming language; whether that language is a traditional programming language like BASIC or the commands and syntax of ABC Writer. Also, let's not strive to glorify the knowledge of programming for programming's sake. When it comes right down to it, the knowledge of how to write assembly language programs or how to use that powerful word processor is not really that important, unless you are using that knowledge to create something, be it an adaptation of that arcade shoot'em-up or the best-selling novel.

So yes, we all need to learn how to become programmers of a sort, at least if we want to get the most out of our microcomputing investment. Perhaps someday, this kind of "programming" on our part will not be necessary in order to make a computer function, but I hope not, because I suspect that if computers evolve to that point, they will be doing the programming and we will be following the instructions.

The 1670...alive and updated: Fueled by shortages at dealers around the country, for months there has been speculation that Commodore was about to discontinue the extremely popular 1670 1200 baud modem or release a new version of it. I am happy to report that Commodore is indeed making ready a new version of the 1670. Here are some of the highlights:

* Default for auto answer is controlled by a DIP switch.
* 100% Hayes compatible. Does not hang up on + + +. Same firmware as 1680.
* Works with either upper or lower case "AT" commands.
* Defaults to echo on. No need to ATE1
* DIP switch for CD enable/disable.
* DIP switch for DTR enable/disable.
* DIP switch for SI (speed indicate) to tell the speed of incoming call.

No word on final price or retail availability, but given the current competitive nature of modem pricing due to asian imports I wager that it will come in for under $120.

BASIC 8 Book: Software Support International, formerly Computer Mart, has just finished work on BASIC 8 for Beginners, an introductory text on getting the most from PATECH Software's BASIC 8. This book, authored by SSI's own Dan Hill sells for $19.95 complete with companion disk full of programming examples from the book. Software Support International, 2700 NE Anderson Road, Vancouver, WA 98661 Suite D13, 206-695-1393.

Format Executive and Juggler 128 Information: First there was Big Blue Reader and Crosslink (public domain) and now there are two new entries in the multiple disk format reader sweepstakes, but unlike Big Blue Reader and Crosslink these programs run in CP/M mode.

The Format Executive allows your C-128 with 1571 disk drive to read, write and format OVER 100 different MS-DOS (PC-DOS), CP/M-80, CP/M-86, Commodore CP/M and Commodore DOS (prg,seq,usr,rel) disk formats. Format Executive claims to support the 1571, and 1581 floppy disk drives, the 1700/1750 RAM expansion units as well as some hard drives. Format Executive also claims to be compatible with both the new and updated C-128 and 1571 firmware. Another valuable plus seems to be its ability to insert or delete linefeeds automatically during file transfers, thereby alleviating a commonly occuring headache that crops up during such file transfers. Format Executive is available from Powersoft Inc., P.O. Box 7333, Bradenton, FL 34210, 813-794-8818 for $59.95 plus $3.50 shipping and handling.

Juggler 128 is an easy to use CP/M mode program designed for use on the C-128 and a 1571 or 1581 disk drive by Miklos Garamszeghy, the author of the superior public domain C-128 programs Crosslink and SDIR, as well as several articles in Twin Cities 128 and Transactor magazine. Juggler 128 supports over 130 MFM disk formats. Up to three of these extra disk types can be installed on your system and can be changed at any time. In addition, you can format a disk in any of the supported types, not just those currently installed. Once installed however, you can read and write these disks with no further assistance from Juggler, support is fully automatic from the CP/M operating system. Juggler 128 is compatible with all current versions of C-128 CP/M and with all C-128 and C-128D hardware configurations. Juggler 128 is being produced and distributed by Transactor Publishing Inc., 85 West Wilmot Street, Unit 10, Richmond Hill, Ontario Canada L4B 1K7.

Another ROM Upgrade? Free Spirit Software Inc. has released The Super Chips, a custom operating system for the C128 computer. These chips appear to a C-128 programmer's dream come true! The system consists of three 16K chips labeled Basic LO, Basic HI and Kernal. These chips replace the chips labeled U33, U34 and U35 on the motherboard of the C128 computer. The Super Chips add a variety of powerful new commands and functions to the C128 operating system. Some of these include:

TYPE - list a program or file to the screen without disturbing the program in memory.
COMBINE - appends a program on disk to a program in memory.
MERGE - merges program on disk with overlapping line numbers with program in memory.
FILE - returns starting address, length and ending address in both hex and decimal of program file.
CHANGE - changes all instances of a specified string within specified delimiters to a second specified string.
FIND - find instances of text string within specified delimiters.
UNNEW - resurrect a NEWed BASIC program or a BASIC program after a reset.
EDITOR - enters a menu driven, 80 column screen editor which permits the user to draw in any color, with flash on or off, underlining on or off and in any background or foreground colors. When done, a BASIC program can be compiled which can be incorporated into a program and/or saved to disk.

The custom operating system redefines the function keys as follows:

F1 = DLOAD a program from the directory
F2 = BLOAD a program from the directory
F3 = DIRECTORY of drive 8
F4 = DIRECTORY of drive 9
F5 = PRINT D$
F6 = TYPE (list a file from the directory to the screen)
F7 = LIST
F8 = MONITOR

In 80 column mode the F3/F4 keys will simultaneously display the directories from devices 8 and 9 in separate windows on the screen. The operating system will default to FAST mode when powered up or reset with the 40/80 DISPLAY button down. It will default to SLOW in the 40 column mode. Free Spirit claims the Super Chips are compatible with 1541/1571/1581 disk drives and virtually all Commodore software and peripherals. Free Spirit also informs us that a C-128D version will be available soon.
C-128 Super Chips, Free Spirit Software, Inc., 905 W. Hillgrove, Suite 6, La Grange, IL 60525, 312-352-7323
$49.95 and shipping and handling are free.

No more head banging from Abacus: Abacus Software the producers of a host of C-128 software have announced plans to begin offering their software in non-copy protected form and on both 3.5 inch disks as well as 5.25 inch media. For more information, contact Abacus Software, P.O. Box 7211, Grand Rapids MI 49510, 616-241-5510.

The C-128 gets the Write Stuff: In a recent telephone conversation with R. Eric Lee, the author of the fabulous and inexpensive word processing package for the C-64, The Write Stuff, Mr. Lee told us he was busy adapting the Write Stuff for the C-128. He indicated that he hoped to have The Write Stuff 128 ready in April of 1988. For more information contact: Busy Bee Software, P.O. Box 2959, Lompoc CA 93438. Editor's Note: If the C-128 version of The Write Stuff lives up to the quality, power, and value of its C-64 predecessor, it will easily make it into the top five of my list of the best word processing programs for the C-128.

Paint & Publish in progress: Lou Wallace informs me that he and David Darus have finished preliminary design work and have begun coding their 100% assembly language drawing and desktop publishing program. Lou told me in a recent telephone conversation that Paint & Publish will support user definable horizontal and vertical resolutions of up to 2400 x 3150 pixels (laser printer resolution for a standard page). Lou hoped that work on Paint & Publish would be complete by June 1988.

Berkworks promises: Berkeley Software indicated in a recent letter that GEOfile 128 and GEOcalc 128 should be available by early February 1988 as well as Deskpack II featuring new GEOS fonts and utilities sometime later in the spring. Berkeley also indicated that they were exploring the possibility of creating a C-128 version of their GEOpublish program for the C-64. For more information contact: Berkeley Softworks, 2150 Shattuck Avenue, Berkeley CA 94704.

BASIC 8 upgrade info: Paresh Patel of PATECH software tells us that BASIC 8 release 2 is now shipping. The upgrade is available to registered BASIC 8 owners for $8.00 with return of your original BASIC 8 disk. The upgrade fixes two minor bugs in the three dimensional graphics portion of BASIC 8 and features many new fonts as well as three new demonstration applications written in BASIC 8: BASICwrite, BASICcalc, and BASICprint and a new logo making utility. PATECH Software, P.O. Box 5208, Somerset NJ 08873.

Voyager Mindtools Inc. to publish Twin Cities 128: Beginning with Issue #20 of Twin Cities 128, North America's only C-128 specific publication will be published by Voyager Mindtools Inc. VMI is a new company devoted to supporting Owners of the Commodore 128 personal computer in a variety of innovative ways. VMI is currently working on several exciting projects for C-128 owners including a Twin Cities 128 back issue compendium book which is slated to be ready by March 1988, and a practical beginners guide to C-128 usage to be available sometime during the Spring 1988.

Twin Cities 128 T-Shirts still available: Those nifty looking Blue and Gold Twin Cities 128 T-shirts are still available for the modest cost of $10. These high quality 50% cotton, 50% polyester shirts feature the Twin Cities 128 logo and our motto: "Commodore Built it...WE support it!". When ordering specify size: S M L XL. Twin Cities 128, P.O. Box 4625, Saint Paul MN 55104, Attn: T-shirt Offer.

1581 Problems Revealed and Solved by Fred Bowen Twin Cities 128 #19 Page 6

By now most of you have probably heard that there are some folks experiencing problems with the 1581 disk drive. Some of you have also taken the time to write us here at Twin Cities 128 to let us know about your difficulties. I want to assure you that your problems and concerns were not only examined, but in many cases passed on to our good friend at Commodore Engineering, Fred Bowen. Unfortunately, neither we at Twin Cities 128 nor Fred at Commodore always have the time to reply to your letters personally (or promptly for that matter). The following information has been gleaned from hours of telephone conversations between Fred and myself, as well as several hundred kilobytes of electronic mail via USENET, Quantum Link, and GEnie over the past six months. I am proud to say all the following information has been rigorously tested and verified and is the "official" scoop from Commodore.

Fred: *Well, it's time to clear the air regarding the 1581 problems. Hopefully this will address the most commonly expressed grievances, provide an explaination of the situation and describe what to do about it. The 1581 3.5" disk drive had undergone fairly complete and extensive testing both in-house and through numerous beta test sites. It is as clean and reliable a system as one could hope for, and I am satisfied with what we have created. Unfortunately, a couple of early production-related hardware problems crept into some of the first systems, affecting 500 to 2000 drives. These have been investigated and corrected as quickly as possible, and service centers have been sent bulletins. Repairs should be made as warranty repairs for these particular problems, even if your warranty has expired. Now, the complaints and gory details:*

Problem: Looking at the "filecopy" program on the "1581 test/demo" disk included with the new drive, I noticed a bug. There is a line that says:

```
60 :if su<4 or su>31 or du<4 or du>31 or su=du then 10 >
```

but line 10 does not exist. I believe it should read "then 25" which is a jump to a subroutine that restarts the program.

Fred: *Correct. Actually, someone here at West Chester changed my original FILECOPY program for the 1581 Test/Demo disk and this was the result. Newer 1581's are shipping with a version 2.0 Test/Demo disk with this and other minor changes.*

Problem: Using the "uni-copy" program to transfer all my old speedscript files onto the 3.5" disk, the program seems to work fine. But when I list the directory, the middle portion looks like:

```
2   "captured"      prg
0   "quest"        *prg
2   "
0
0
0
17732 L
1   "report"        prg
```

Fred: *There are two problems. The first one was easy: pin 10 of U10 was not properly grounded in some units. This was caused by some unmodified pre-production boards sneaking into a final production run. The number of units affected, I am told, is about 500. The symptions vary from occasional "device not present" errors to data not being written or retrieved from disk correctly. The fix is a simple one- ground pin 10. If you value your warranty, a service center will do this for you as a warranty repair.*

The second problem was more difficult to isolate. Some (not all) disk drives with a WD1770 controller chip (U4), occasionally did not correctly write data to disk. The previous example of a corrupted directory following a disk copy is typical. This was caused by a particular lot or two of WD1770 controllers. The number of units affected is around 2000 I am told. The fix is simple but hard to do i.e replace the WD1770-00 controller with a WD1772-00 controller. The 1581 was built to use either one and, in fact, the vast majority of systems did use WD1772 controllers. I recommend you see a service center to have this repair made. Associated with the controller is a jumper, J1, physically located adjacent to U4. The jumper should always be shorted with a 47-ohm resistor for a 6ms step rate.

And now yet another installment of: "nifty programs from Fred that tell you what you have inside of your equipment without voiding that precious Commodore warranty". This program will tell you what kind of controller you have in your 1581 as well as telling you the status of the J1 Jumper.

```
100 REM  CHECK 1581 DISK FOR CONTROLLER TYPE & J1 JUMPER
110 REM  12/09/87  FRED BOWEN
115 :
120 PRINT"INSERT ANY FORMATTED DISK IN DRIVE."
121 INPUT"CHECK WHICH UNIT";U: PRINT
125 :
130 OPEN 1,U,15      :L=DEC("1DA")AND255: H=DEC("1DA")/256
140 OPEN 2,U,2,"#"
145 PRINT#1,"M-R"CHR$(0)CHR$(192)CHR$(1): GET#1,A$
150 IF ASC(A$)<>192 THEN PRINT"DEVICE"U"IS NOT A 1581.": GOTO360
155 :
160 PRINT#1,"M-R"CHR$(L)CHR$(H)CHR$(5):      REM  MODIFY CONTROLLER CMDS
170 FORI=1TO5:GET#1,A$:B$=B$+CHR$(ASC(A$)OR3):C$=C$+A$:NEXT
180 PRINT#1,"M-W"CHR$(L)CHR$(H)CHR$(5)B$
190 :
200 PRINT#1,"U1";2;0;1;0       :GOSUB500:    REM  SEEK TRACK 1 & RESET TIMER
210 PRINT#1,"U1";2;0;80;0       :GOSUB600:    REM  SEEK TRACK 80 & READ  TIMER
220 :
230 PRINT#1,"M-W"CHR$(8)CHR$(64)CHR$(1)CHR$(0):REM  TEST JUMPER J1
240 PRINT#1,"M-R"CHR$(8)CHR$(64)CHR$(1): GET#1,J$
250 :
260 PRINT#1,"M-W"CHR$(L)CHR$(H)CHR$(5)C$:    REM  RESTORE CONTROLLER CMDS
270 IF VAL(F$)>0 THEN PRINT"SEEK ERROR- CHECK DISKETTE.": PRINT: RUN
285 :
290 PRINT"UNIT"U" CONTAINS A ";:              REM  REPORT
300 IF T>20 THEN PRINT"WD1770";
310 IF T<20 THEN PRINT"WD1772";
320 PRINT" AND J1 IS ";
330 IF J$="" THEN PRINT"OPEN"
340 IF J$>"" THEN PRINT"CLOSED"
360 :
370 CLOSE2: CLOSE1: END
380 :
500 FORI=11 TO 8 STEP-1: POKEDEC("DC00")+I,0: NEXT: RETURN
600 INPUT#1,F$,R$,E$,D$: T=PEEK(DEC("DC09"))*10+PEEK(DEC("DC08")): RETURN
```

If you are a member of the Quamtum Link or GEnie telecommunication networks you might want to save yourself some time and effort and download these programs rather than typing them in by hand. On GEnie, the file can be found in the software libraries of the Flagship Commodore Area. On Quantum Link you will find the file in the Lovhaug's Specials Library of the C-128 Special Interest Group by Twin Cities 128.

An Evaluation of PaperClip III by Loren Lovhaug Twin Cities 128 #19 Page 8

It has been said that the only constant in the universe is change. Since the arrival of Paperclip III, I am now convinced that this is the case. You see, ever since Pocket Writer 2 by Digital Solutions was released during late 1986 it was my belief that I had obtained the ultimate in word processing software for my C-128. For the casual C-128 observer that might not seem like an impressive statement, but when you consider how many excellent word processors are available for the C-128 my statement begins to take on new meaning. There are no fewer than five word processing packages available for the C-128 that I consider superior, packages which can compete quite favorably with any word processing package available on any microcomputer, and probably another three or four that I consider to be very good. However with the arrival of PaperClip III the way I create and edit text has been radically changed.

Let me explain. Pocket Writer 2 is without a doubt one of the nicest word processors I have ever used, and continue to use. It still remains the only full featured truly what-you-see-is-what-you-get word processor for the C-128. In addition, Pocket Writer 2 fully supports all of the peripherals that have appeared from Commodore that really enhance the performance of the C-128 such as the 1750 RAM expansion unit, the 1351 proportional mouse, and the 1581 disk drive. Pocket Writer 2 is capable of virtually every word processing task that you can conjure up, and it is amazingly easy to learn and use. However, Pocket Writer 2 has a glaring weakness, its spell checker. Although greatly improved from the original release of Pocket Writer by virtue of its usage of the C-128's 80 column screen and the 1750 RAM expansion unit, the spell checker in Pocket Writer 2 still left much to be desired. It is impossible to delete a word once it is added to Pocket Writer 2's dictionary. This means that if you accidentally add a misspelled word to the dictionary, (it happens more often then one would think) you are forced to live with it, unless of course you maintain a frequently updated backup of your dictionary. Another design flaw of the Pocket Writer 2 spell checker is its inability to lookup misspellings or suggest alternate spellings thereby assisting you in the actual correction process. This means that the so called "spell checker" is in reality nothing more than a "word finder", finding or identifying those words which are not part of its dictionary.

Spell checking is where PaperClip III really shines. In fact, PaperClip III elevates spell checking to a whole new plane on eight bit microcomputers, into a realm where only very expensive word processing packages on very large and expensive computers previously tread. With a 1750 RAM expander attached to your C-128, PaperClip III is capable of real-time or "automatic" spell checking. This means that PaperClip III automatically checks your spelling as you type. Here is how it works: once you have loaded your dictionary into your 1750 RAM expansion unit, simple activate the auto-checker and immediately after you type a word followed by a space or punctuation mark, your word is compared against the dictionary stored in the RAM expander, if the word is not found in the dictionary it is displayed on top line of the screen and a tone is sounded (the tone can be turned off). Surprisingly the real-time spell checking ability does not retard PaperClip III's ability to process characters while typing. This real-time spell checking ability is fabulous, although I must admit it is a little like having your 8th grade english teacher standing behind you nagging you (I hope Miss Bode will forgive me, after all she was right: spelling always counts). Some users will no doubt, however, find this type of spell-checker annoying and opt for the more conventional approach of writing your text and spell checking it after it is complete. I am pleased to report that PaperClip III is very capable from this standpoint as well. Spell checking from disk is very fast, and spell checking with the 1750 RAM expansion unit is nearly instant. PaperClip III's spell checker comes with a 40,000 word dictionary which is user definable, allowing the user to add or delete words as he or she chooses. Beyond that, when using the non-automatic spell checker PaperClip III will help you to correct spelling errors by allowing you to access the dictionary to help you determine the proper spelling of a misspelled word. And finally if all this was not enough, PaperClip III's spell checker has a third option which I have come to love, namely a user definable auto-expander. This feature, available only to owners of 1750 RAM expansion units, allows you to program PaperClip III with a list of characters that it will automatically substitute (or expand) for other characters. This feature has two principle uses, the first is to eliminate the misspelling of words that you commonly misspell. For example, often when I need to type the word "original" I leave out the first letter i. Since my wife, our chief proofreader has often scolded me on this failure, I decided to add the characters "orginal" to my expansion list and told it to automatically substitute the correct spelling "original" whenever I type the characters "orginal". Incidentally, in order to type the three previous sentences I had to turn off the auto-expander so that the substitution was not made. The second principle use of the auto-expander is for speed or macro typing. For example, many times when I am writing articles or correspondence I need to type the words: Twin Cities 128. In my auto-expansion file I have told PaperClip III to substitute the characters "Twin Cities 128" whenever I type the characters "tc".

Now, if PaperClip III was only capable of the above mentioned wonders it would indeed be an excellent value, but PaperClip III is a full featured professional word processing program capable of a great deal more. Besides all the minimal features you would expect in a good word processing program such as range manipulation, search and replace, mail merge, justification control, margin control, automatic page numbering, header and footer support, global file support etc., PaperClip III also features a whole host

of esoteric features that will surely delight anyone who has sophisticated word processing chores. For instance, PaperClip III has a plethora of column manipulation functions that are wonderful for creating, editing, sorting, and doing calculations based on columns of numbers or text. One feature noticeably missing from PaperClip III's rich collection of column commands is the automatic generation of newspaper or magazine style columns such as those you see on this page, however, with a little practice and effort you can electronically create them without getting out the old scissors and spray mount.

Other advanced word processing features that are included in PaperClip III's arsenal are: the ability to automatically create a table of contents, outline generation with user defined indentation and support both arabic and roman numerals in both upper and lower case, automatic chapter numbering, the ability to read text files created with most word processing programs, 50 line video output, and automatic hyphenation of words with definable breakpoints.

Another advanced feature that should not be overlooked is the PaperClip III's keyboard macro (referred to in the manual to in the manual as "instant phrase") abilities. This feature, like the auto-expander mentioned earlier allows allows you to type large amounts of text with just a few keystrokes. However, unlike the auto-expander, a RAM expansion unit is not required. Instead you simply assign a word or phrase to one of 52 keys on the keyboard. The when you hit the ESC key followed by one of the letters that have been assigned the text that has been assigned to that key is typed on the screen for you. Interestingly enough there is virtually no limit to what you can assign to individual keystrokes including format directives.

And if all this was not enough, PaperClip III also comes complete with a fully functional integrated telecommunications package that can functions in a co-resident manner with PaperClip III and uses the word processor's text area as a buffer. I must admit that at first glance I felt this option to be more or less a gimmick rather than a useful feature, but after utilizing this function I have found it to be a very valuable asset, especially for a person like me that receives and transmits a great deal of text via modem. Without leaving the word processor, I am able send and receive data from local BBSs, national telecommunications networks, or my Model 100 portable computer. This means that I can use all of PaperClip III's text editing and spell checking functions on my messages and electronic mail, as well as having immediate access to those facilities for the mail, messages, and articles I am receiving. Also included in the word processor's editing functions is a special "unformat" command which automatically and instantly strips away carriage returns away from your text after a specified column. This function is a boon for telecommunications purposes since buffered text that is received via modem almost always has carriage returns following each line that must be ultimately deleted from the text.

PaperClip III comes on two 5.25 inch double-sided 1541 formatted disks. One disk contains the C-64 mode version of PaperClip III which I have not examined. The other disk contains the C-128 version on the front side of the disk and the dictionary files and printer drivers. PaperClip III is not copy protected in any way, and unlike previous versions of PaperClip, there is no dongle. Because of the lack copy protection it was a trivial matter to transfer PaperClip III, the dictionary, and their associated files to both a 1571 and 1581 diskettes. Also I am able to load and run PaperClip III from the 1581 configured as unit 9 by making a simple alteration in PaperClip III's BASIC loader, thereby allowing me to load and run PaperClip III in seconds!

The manual that comes with PaperClip III is excellent. It is well designed, including both an exhaustive tutorial section as well as a quick reference guide. Also included is a complete index and table of contents.

With all of the praise I have bestowed upon PaperClip III is there anything that I did not like about it? Of course! What kind of reviewer would I be if I did not find something to nit-pick about? To begin, PaperClip III is a post formatted word processor. This means that unlike true "what-you-see-is-what-you-get" (WYSIWYG) word processing programs like Pocket Writer 2 and GEOwrite, you do not see the text on screen while you are editing it as it will appear in final formatted form. Instead, format directives (e.g. commands for setting margins etc.) and text enhancement directives (e.g. italics, boldface etc.) are embedded within the text as you edit the text. To examine the text in its final outputted form you must either print out a hardcopy of your text or use PaperClip III's video preview option. This makes editing your text somewhat akin to "programming" your text. This approach is more difficult to learn and use, and is not as nice as the WYSIWYG approach when working on complicated layouts. Another feature that is not available with PaperClip III is a an alternate text area. This feature allows you to subdivide the text area into smaller text areas so that multiple documents may be edited at the same time. This is particularly useful when "borrowing" text from one document to the another. Also PaperClip III does not support the 1351 mouse.

In the final analysis PaperClip III is an excellent product and well worth its list price of $39.95. If you are an owner of a previous version of PaperClip, I strongly urge you to upgrade to PaperClip III by sending the cover of your older version of PaperClip and a check for $18 to: Electronic Arts, P.O. Box 7530, San Mateo CA 94404, ATTN: Batteries Included Upgrade - PaperClip III. I now split my word processing chores between PaperClip III and Pocket Writer 2, using each for what they do best.

COLOREZ 128 is a graphics utility for programmers working with color graphics and the C-128's 80-column screen, although it does have a couple of sneaky uses for strict 40-column graphics programmers, too. The main function of COLOREZ 128 is the transferring of "regular" 40-column high-res graphics screens to the VDC chip for 80-column viewing. The 320-pixel-wide graphics can be transferred to the 640-pixel-wide VDC screen in one of three manners: 1. By using the full 640-pixel width; this is done by doubling each of the original 320 pixels. The result is a surprisingly good duplicate of the original picture. 2. The 40-column graphic can be transferred to the left side of the 80-column screen, or 3. to the right side.

There is one very important caveat to the above; the standard Commodore 128 with 16k of VDC RAM doesn't have enough memory to display a full-color 640*200 pixel picture, so the bottom three "lines" (24 lines of 640 pixels) aren't displayed, and the border/background color shows through instead. COLOREZ 128 also doesn't currently support 64K VDC RAM, although an upgrade is promised. Of course, even when COLOREZ 128 does support 64K RAM, programmers will have to tailor their graphics to look good on the vast majority of 128's with the 16K RAM. (The new Commodore 128D's are being produced with 64K VDC RAM, and the new cost-reduced boards for the regular "flat" 128's are also supposed to be produced with 64K. But for the present, 16K is by far the most common.)

COLOREZ 128 does not directly support monochrome high-res graphics (as produced by Ultra-Res, for examples), nor can it work directly with BASIC 8 graphics, although it can save images in BASIC 8 format, or the graphics produced by other Commodore 64 products saved on disk with other than the Doodle-like format of 1024 bytes of color information followed directly by 8000 bytes of bitmap information. But this is no big limitation, as conversion instructions and/or programs for all of the above formats are included!

Here's a list of the capabilities of COLOREZ 128:
Displays standard C64-type high-res graphics full-width on the 80-column screen, with the bottom three "character lines" of the display missing (see above).

Displays standard C64-type hires graphics on the 80-column on either the left or right side of the 80-column screen.

Allows you to load a standard 40-column graphic to the full-width 80-column screen, then save each half of the 80-column screen (left or right) in 40-column mode, ready for editing by a program like Doodle. (This is a feature that a strict 40-column user may find useful, although at present, you're still going to lose those three character lines at the bottom of the screen.)

COLOREZ 128 can save 80-column graphics in its own format, in a space-saving Compressed mode, or in BASIC 8 format!

COLOREZ 128 can save 80-column graphics to Bank 1, for use with the 1700/1750 RAM expander. Included on the disk are sample disk-based and RAM expander slide show programs.

COLOREZ 128 is NOT copy protected, and the entire program and all routines are accessible to the programmer. (This is good, because the icon-driven program shell needs lots of work and polishing.) All of the graphics swapping, loading, and saving features of COLOREZ 128 can also be performed with well-documented SYS calls.

Two run-time packages are included that you can add to public-domain or commercial program disks without licensing or royalties. These packages allow you to load and display the pictures created by COLOREZ 128 in your own programs. It seems obvious to me that programmer Tom Brown gets frustrated (as I do) at programming utilities and languages that make themselves all but impossible to use through heavy copy protection, undocumented uses of memory, clumsy user interfaces, etc. (Are you reading this, Abacus?) Instead, Tom tells you where the routines are located (and why!), tells you how to use them, plus he includes programming examples on the disk, and left all of his support programs in un-compiled BASIC so you can see how things are done. Also included is a graphic tutorial (on disk) that explains a lot of how character and bit-mapped graphics work on the Commodore 64 and 128. Unfortunately, the tutorial stops just as it starts to get interesting, in the middle of a discussion of the 8563 chip. (Finish it for the next release, Tom!) And all for only $14.95!

Of course, not everything's perfect. (Remember that we reviewers don't get paid unless we find and write about a program's shortcomings!) The instruction booklet is printed with a "sorta near letter quality" dot-matrix printer at an uncomfortable to read 8 lines per inch, and the program for reading the disk-based documentation and graphics tutorial was, well, terrible. While the package advertises that it supports the mouse, what they REALLY meant was that it supports the old 1350 joystick-emulating mouse, and not the true 1351 proportional mouse.

The icon "capture" registration is also imprecise. When trying to transfer a 40-column graphic to the right side of the 80-column screen (which you do by moving a little graphic icon to the center, left, or right side of a picture of the 80-column monitor on the screen), I could never figure out exactly where I was expected to "drop" the icon to transfer the graphic properly. (I could have used the function keys, or direct SYS calls, to do the same thing, but the icon-based shell would be the most convenient--if it worked properly.)

COLOREZ 128 is a powerful 80-column utility, with usable routines available to the programmer. Three cheers and a huzzah for a utility program that can actually be used!
COLOREZ 128, B-Ware Computer Systems/Briwall, Box 129 Kutztown, PA 19153 $14.95

According to the figures we get through the grapevine, in spite of the reported sales of that 'other' Commodore machine (I have sworn never to say the name in print), the C-128 and the C-64 are still selling remarkably. This is as the C-128 is rooted firmly in its' 3rd year of production, and the C-64 is the grandfather of the pack at nearly 6 years. The primary reason both of the 8-bits are doing so well is simple: tons of wonderful software continues to be released that people can USE on a day to day basis, without having to upgrade their already significant investment. The product I am reviewing here is no exception.

The Re-RUN Productivity software is, without a doubt, a wonder, both in scope of applications and cost. I speak as someone who uses his computer daily to manage both his personal and professional lives and help create the magazine before you. Within this package is a pretty good word processor, a good spreadsheet, a passable database, and an investment plotter. All for less than $20.

While this package contains versions for both the C-128 and the C-64, I haven't booted up the C-64 versions, since what interests me (and hopefully you) are the 128 programs.

Let me start with the RUN Script 128 word processor. It is a post-formatted word processor, about as easy to use as such applications usually are, but has the added bonus of a spell checker and a user-definable dictionary. Text entry is very simple, and the documentation is quite extraordinary, with both charts within the supplied booklet, and function key overlays that can be cut from the tag-board covers (not the best feature, but the cost is the remarkable aspect here, not doo-dads). It allows the entire PETscii (Commodore's version of ASCII), including the graphic characters, assuming that you have a dot-matrix printer.

The standard bread and butter text handling and formatting features are supported, but the more extravagant "power-user" features such as those found in Pocket Writer 2 and PaperClip III are not.

The program is easily transferred to a 1581 disk drive using any standard copying program. The small minority among us who insist on using our 1581's as our primary drive (#8) appreciate this feature.

The spell checker is, to say the least, interesting. Most spell-checkers now available dump the dictionary into RAM or RAM expander, and run through the file to be checked sequentially. RUN Script 128 reads the dictionary files into memory, and then accesses the text alphabetically, jumping around the document in what I can only say are dramatic leaps and bounds. But the result is as efficient as any, although the small 3000-word dictionary means a lot of adding additional dictionary files to make even the average user a fast spell-checker.

RUN Script 128 is able to import standard Commodore Sequential files easily and without error. The program files created by Pocket Writer and PaperClip load, but with very strange (but correctable) formatting results.

This is not to say that the program is without flaw. The most serious for me was the inability to get my dot-matrix printer to work with it, since I use the User Port to send data to my printer. I was able to get my DPS-1101 Daisy Wheel printer to work flawlessly (without graphics characters, of course), as this accepts PETscii as well as ASCII data. The default driver is the MPS-803 printer, although you can tailor the printer drivers to some degree; this program definitely likes Commodore printers best.

RUN Calc 128 works in the manner of most common spreadsheets, although I did find it lacking support for more than 1 drive. It was the only application on the disk that I needed no documentation for, as a constantly available HELP screen was available at the press of the button. The working space of RUN Calc 128 was a bit limited (26 x 250 cells), but it accepted both text and numeric entries, which make spreadsheets easy to use.

RUN File 128 is a fairly normal database application; it allows up to 30 fields with a maximum of 254 characters in a record (this is one Commodore disk block long), which might make it seem pretty skimpy if you have been using PocketFiler or SuperBase128, but for an average database, like an address list, it seems quite long enough. This application does allow you to access more than one disk drive, and uses the 1581 pretty well, although I was concerned about the User files RUN File 128 creates. I found no conflicts in my experimentation.

RUN Investor seems to be little more than a combination spreadsheet/database, since it allows the listing of the rise and fall of a given stock and keeps a record of it. A separate file is created for each stock, so editing is quite easy. Notepad 128 is a fun little utility that gives a Sidekick-like notepad with clock that pops up on command. It would not work with any of the applications I usually use (Pocket series, Bobsterm, or Superbase), but it did work well with several small BASIC programs that I use.

The Productivity Pack III is a good deal for the new user that has not already found that 'special' program at higher costs. In it, the user finds several programs that will let him use the standard 128 package (128, drive, monitor, and printer) to its' fullest potential, without laying out a small fortune. As far as us long time 128 owners go, the hackers might find it worth-while to crack some of the code on this disk to see what the pros are up to, but for the user who already has his spreadsheet, word processor, and database needs well attended, or the user who requires extreme power and sophistication this package holds little. Re-RUN Productivity Pack III; $19.95. Available from Re-RUN; 80 Pine St.; Petersborough, NH; 03458

I've heard some C-128 owners say that there is no reason to learn about the CP/M mode of their computers because CP/M is dead. This could not be further from the truth. Perhaps these people are really afraid of having to learn a different computer environment. A relatively new implementation of the popular programming language, COMAL, is now available for CP/M. It runs perfectly on the C-128 and uses the RAM expander as a RAM disk if you have one.

Before I talk about CP/M COMAL in detail, I first want to assure people new to CP/M that there is nothing to fear from the "CP/M" part of CP/M COMAL. To enter CP/M COMAL, you must first enter the CP/M mode of your computer. This is accomplished by inserting a backup of the CP/M boot disk into the disk drive and turning on the power for the computer and disk drive (or hitting the reset button on the C-128 if power was already on). The CP/M boot disk was included with your C-128 computer when you purchased it.

Once you are in CP/M mode, you should see the A> prompt familiar to all CP/M users. At this time, you should place the disk containing CP/M COMAL into the disk drive, type the word COMAL, and press the RETURN key. Before you know it, the COMAL copyright message will be displayed on the screen and you will be ready to write or run your COMAL programs. This is all you need to know about CP/M to use CP/M COMAL!

COMAL is a programming language with many of the advantages of BASIC and PASCAL. Like BASIC, COMAL offers the user a command mode to perform simple tasks and a program mode to perform long or complicated tasks. Like Pascal, COMAL offers the user powerful programming structures including named procedures and functions with parameters. Let's try a few things that someone can do their first day with CP/M COMAL.

The easiest command to start learning COMAL with is the PRINT statement. Try entering this line in CP/M COMAL:

```
PRINT "I am learning to program in COMAL"
```

Don't worry about the keyword PRINT being in upper case (shifted) letters. COMAL allows you to enter keywords in either upper or lower case. Using upper case letters is just a convention to make programs easier to understand. More about that later. After you enter the line above and press the RETURN key, COMAL responds with:

```
I am learning to program in COMAL
```

You have just entered your first COMAL command. To turn the command into a program, all it needs is a line number:

```
10 print "I am learning to program in COMAL"
```

To see what is in your program, LIST it:

```
LIST
0010 PRINT "I am learning to program in COMAL"
```

After you typed in the command, LIST, the computer listed your program. It automatically converted the line number to four digits by adding the leading zeroes. It capitalized the keyword PRINT for you too. These features may take some getting used to, but experienced COMAL programmers appreciate them very much.

Running your program is as simple as listing it. Just enter the command, RUN, and press RETURN:

```
RUN
I am learning to program in COMAL
```

Now that you have written a program, you will certainly want to save it:

```
SAVE "learning"
```

This saves the file "LEARNING.SAV" to the default disk drive, usually the A: disk drive. ".SAV" is an extension to the filename that CP/M COMAL uses to remind the user that the file is a SAVed COMAL program file.

To see the other files in the disk directory, use the CAT command (or DIR if you prefer, both commands work identically). You can even send the directory to the printer with these commands:

```
SELECT "LP:"
CAT
SELECT "DS:"
```

The SELECT command changes the output location. "LP:" stands for Line Printer and "DS:" stands for Data Screen. CP/M COMAL also supports "SP:" for Serial Port (the modem port). Note, listing the disk directory does not destroy the program in memory (as it does in Commodore BASIC).

To erase the program in memory, use the command NEW. To load a program file back into memory, use the LOAD command. For example, to retrieve the file, "LEARNING.SAV" type:

```
LOAD "learning"
```

"So COMAL has a few nice easy features. So does BASIC. Why should I use COMAL when BASIC is built into my computer?" One of the nicest features about COMAL is its interactive editor. When you enter a line into the COMAL editor, COMAL first checks the syntax of the line before accepting the line and allowing you to enter another. If it finds an error, it places the cursor on the character causing the problem and suggests a solution to the problem on the line below. When the line is correct, COMAL automatically reformats the line for you. This involves converting keywords to upper case letters, converting variable names to lower case letters, adding optional keywords, and indenting the line the appropriate number of spaces for the structures (if any) it is nested in. For example:

```
0010 for
ERROR number 32 : Name expected
0010 for x
ERROR number 22 : ": = " expected
0010 for x =
ERROR number 37 : Constant or variable expected
0010 for x = 1
ERROR number 21 "TO"/"DOWNTO" expected
0010 for x = 1 to
ERROR number 37 : Constant or variable expected
0010 for x = 1 to 10
0010 FOR x: = 1 TO 10 DO
```

Note, the above example does not fill half your screen and you do not have to retype the line after every mistake. After the error message is printed below the line with the mistake, the cursor is placed back on the line containing the error. All you have to type is the character or two added to the previous line, plus the RETURN key. As you can see, after the line was finally accepted, the keywords were converted to upper case, " = " was converted to ": = ", and the optional word "DO" was added.

However, COMAL still will not let you run a program with this line. COMAL is smart enough to know that you started a multi-line FOR loop, but did not complete it with ENDFOR. This error, too, must be corrected before you can run your program. Does all this error checking slow down COMAL? No! You will probably never detect any delay while COMAL checks for syntax and structural errors. However, you will be pleased with the increased speed of a running program which does not need to double check for these errors.

This should be enough to keep COMAL beginners busy for some time. More advanced COMAL users will be pleased that CP/M COMAL is a full implementation of COMAL 2.0. It runs on the C-128 or CP/M machines running CP/M 2.0 or higher.

CP/M COMAL is quite compatible with the COMAL cartridges for the C-64 and C-128. It provides the four loop structures: FOR, REPEAT, WHILE, and LOOP. It even adds extensions to the FOR and LOOP loops. CP/M COMAL provides the IF structure in all its variations plus the CASE structure. An error handling structure prevents programs from crashing after minor mishaps such as division by zero. Named procedures and functions allow parameters, local and global variables, recursion, and nesting. External procedures and functions can be loaded from disk or RAM disk. This allows long programs to fit in a small space.

Output, and screen formatting are easy in CP/M COMAL. The PRINT statement sends text, numbers, and variables to the screen. The PRINT statement can be modified to PRINT AT a specified screen location. PRINT USING can be used to format numeric output. The SELECT command can be used to redirect output to the printer, modem, or a disk file.

Input is just as easy. CP/M COMAL provides input prompts and a protected input field. This allows the programmer to restrict user input to a specified number of characters. The user cannot cursor out of the input field or use control keys to erase the screen.

CP/M COMAL also provides sequential (text) files as well as random access files (for data bases). The files can be either in ASCII format which is compatible with many word processors or in binary format which is more compact and faster to access.

In addition to the standard COMAL features, CP/M COMAL provides extra commands to take advantage of the Z80 processor. The INP and OUT procedures send data to or from the I/O ports (much like PEEK and POKE do for normal memory). They allow 16 bit addresses so it is possible to program the sound and video chips.

There are three different ways to interface machine language with CP/M COMAL. In-line machine code may be inserted directly into your COMAL programs with the CODE statement. You may CALL machine language routines directly by address. (This works ideally for BDOS and BIOS calls.) The Z80 registers can be initialized for these calls through the CP/M COMAL pseudo variables: AF, BC, DE, and HL. Finally, you can write machine language packages to be linked to your COMAL programs and allow its procedures and functions to be called by name.

One package that has been written for CP/M COMAL is a turtle graphics package. It supports the 640X200 monochrome graphics screen. The graphics package provides routines to plot points, draw lines and circles, and plot text. The line drawing commands allow the turtle to move forward or back, and turn to the right or left. There are also commands to draw lines to specific points on the screen. The pen used to draw the lines can be set to any of three modes. The default is to set points on the screen as the pen passes over. However, you may choose to flip points or to clear points instead.

You can draw pictures using the default screen coordinates where one pixel equals one unit, or you can resize the screen. By changing the screen scale, you can force the screen to fit the data for graphs rather than writing a procedure to transform the data to fit the screen.

CP/M COMAL comes in three forms. The full system allows you to write your own programs and save them to disk to use later. Of course you may also use the programs from others (if they are public domain or you have purchased them).

The demo version of CP/M COMAL allows you to write and run your own COMAL programs and list them to disk or printer. You may also run the programs of others. However, you will

(Continued on Page 24)

Installing a Fan in your C-128D by Roger Milburn Twin Cities 128 #19 Page 14

As we have commented before in Twin Cities 128, the C-128D design we are blessed (cursed?) with in North America is different from the original C-128D design shown at the 1985 Summer CES and marketed in Europe. The original C-128D design is taller and has a plastic, rather than metal, case which has a slot where the keyboard can slide under the system unit. The original C-128D hardware also comes with a built-in fan to cool the internal circuitry of the computer and built-in 1571 disk drive. Our "cost-reduced" units have no such luxury, although there are indications that such was at one time meant to be. Submitted for your perusal: four mounting holes on the rear side of our C-128D just behind the power supply. With this article we are by no means suggesting that the C-128D necessarily "needs" an internal fan, on the contrary, my unit functions extremely well literally around the clock without one, but we thought we would provide the information to you just in case you deemed the addition of a cooling fan a worthwhile pursuit.

As with all hardware modification articles that appear in Twin Cities 128 we cannot take any responsibility for any damage to your equipment or injury either to your person or belongings that might result from attempting the procedure outlined below. Additionally the procedure outlined below will most certainly invalidate any warranty that Commodore provides on your computer. Lastly, this modification should not be attempted by persons that do not feel confident about their own abilities to work with or repair sensitive electrical equipment. If you have any doubts about your abilities it is probably best to have a professional do this work for you.

To add a cooling fan to your C-128D you will need a SUNON model SD1206PTS1 fan (12 VDC), four 6-32 X 3/8 inch pan-head SEMS machine screws, four 6-32 KEPS nuts, four #6 flat washers, and about 6 inches of shrink tubing.

Start by removing the cover from the C-128D AFTER REMOVING POWER FOR AT LEAST 15 minutes (to give capacitors time to discharge). **Please don't take this suggestion lightly. You see, the power-supply for the 128D is a 'switching' supply, and the main rectifier capacitors in it are large, fairly low-leakage, high ESD (the ability to discharge into a load rapidly -- like your body!!)**, and they carry a charge of over 300 volts!!!! Watch yourself and remove all metal jewelry, including rings before working around any switching-type power-supply!!

Next, locate the power supply module at the left rear corner of the chassis. You will find a multi-pin connector wired to the module, the connector plugged into the main board of the C-128D. Disconnect that plug. Also, locate the connector for the front panel power indicator LED and disconnect it from the power supply.

Remove the module from the chassis. There are 2 Phillips machine screws in the back of the chassis, two on the left (with the C-128D is facing you) and one in a recess on the top of the module itself. All of these screws must be removed. **Don't lose the screws!** They're metric. Lift the power supply out and set the computer aside.

Detach the power supply printed circuit board from the chassis to gain working room for installing the fan. Attach the fan to the power supply chassis with the hardware in the following sequence: First, screw the heads against the rear of the power supply chassis, then the 4 flat washers, then the fan itself with the SUNON label facing toward the power supply PC board and finally the KEPS nuts. Note that the washers will slightly space the fan away from the metal chassis. This is necessary to avoid distorting the fan housing when the chassis is reattached to the computer.

Once the fan has been installed, attach the PC board to the chassis and install the chassis into the computer. Next, slide the shrink tube down the fan leads so that it is against the fan housing. Shrink the tubing.

Connect the red lead of the fan to the cathode (banded end) of diode D202 on the power supply board (you'll have to solder it directly to the diode lead). Crimp a #4 ring tongue terminal to the black fan lead and attach same to one of the screws securing the power supply transformer to the power supply chassis. Plug in the power supply connector to the main board and plug the power indicator LED into the power supply.

Double-check your work before the next step. Connect the computer to the 110 volt power line with no other devices connected. Turn on the computer and observe that the fan starts running and blowing air over the power supply. Watch the drive busy LED for signs of activity. Finish up by replacing the cover on the computer. NOTE: If the fan is correctly installed but blows air in the wrong direction you have the leads reversed. With a little luck and skill, you now have a cooling fan protecting your computer investment.

In politics it is often true that what goes on "behind the scenes" is often as important if not more important than what goes on in plain view. Interestingly enough I have discovered this axiom also holds true when working with BASIC 8. There is a big temptation when writing or examining BASIC 8 programs to focus only on that which is visible, when in many cases the most powerful and sophisticated programs actually gain most of their power from that which is invisible or at least functioning in the background.

For example, boot up your BASIC 8 editor and enter the program listed on the right column of this page. This program will work on all C-128s regardless of the amount of video RAM on your system. Once you have entered the program and saved it onto a disk RUN it. The program presents you with a menu describing seven pictures. To the right of the menu appears the picture described by the highlighted menu item. Use the cursor keys (the up & down arrows) to scroll the highlight bar onto any of the menu items and observe how quickly the graphic representation changes and how clean the scrolling of the highlight bar is. This program is a good example of how through the use of right techniques you can create extremely pleasing visual effects from BASIC even without the benefit of expanded video RAM.

As alluded to in my opening paragraph, the real trick which makes the quick flipping of the graphic images and the scrolling of the menu bar possible are not obvious when watching the program in action, or even from an examination of the listing. In both cases I take advantage of an option of BASIC 8's @clear command that unless you read your BASIC 8 manual very carefully probably went unnoticed.

To begin, let's review the @clear command. @clear is used to "clear" or perhaps a better term is "fill" the currently defined window (as defined by the @windowopen command, the default window is the entire screen as defined by the @mode and @screen commands or the @scrdef command) with specified bitmap or color information. The @clear command has three parameters, the first parameter is required, the latter two are optional. The first parameter is called the bitmap fill value. The second and third parameters allow you to specify a background color and a foreground color with which to fill the current window. The bitmap fill parameter determines what value is to be placed into every byte of RAM in the currently defined window. If the bitmap fill value is zero, then bitmap of the currently defined window is erased. Values between 2 and 254 fill the window with various vertical striped patterns. These patterns reflect which individual bits (pixels) are turned on given the value that is placed in each byte. For instance, a value of 128 would turn on the left-most bit in each byte of the bitmap, thereby causing the currently defined window to be filled with vertical lines beginning with the far left-most column of pixels repeating every eighth column of pixels. If a value of 255 is used as the bitmap fill value then all of the bits in all of the bytes in the window are turned on, thereby rendering a solid box. And now comes the often overlooked part, if you specify a bitmap fill value of 1, the bitmap itself remains unchanged and the color memory for the window is simply filled with the values specified by the second and third parameters of by the last settings specified by the @color if the @clear command's background and foreground parameters are omitted.

```
100 def fnr(x) = int(rnd(1)*x) + 1
110 c$ = chr$(3) + chr$(5) + chr$(144):cq$ = chr$(3) + chr$(144)
120 ch$ = chr$(145) + chr$(17) + chr$(13)
130 n(1) = -1:n(2) = 1:n = 1
140 @walrus,0
150 @mode,0
160 @screen,2
170 @color,0,0,0
180 @clear,0
190 @box,0,0,0,39,39,0,0,0,1
200 @line,40,39,0,60,0,0,1
210 @line,60,0,0,79,39,0,1
220 @line,79,39,0,40,39,0,1
230 @circle,100,19,0,19,1
240 @arc,140,19,0,19,19,0,360,72,1,0
250 for i = 1 to 20
260 @dot,fnr(39) + 160,fnr(39),0
270 @line,fnr(39) + 200,fnr(39),0,fnr(39) + 200,fnr(39),0,1
280 next i
290 @char,254,31,0,1,3,4,cq$ + "QUIT"
300 data "Box",13,2
310 data "Triangle",13,10
320 data "Circle",13,4
330 data "Pentagon",15,8
340 data "Dots",6,2
350 data "Lines",13,8
360 data "Quit",0,15
370 @char,254,16,64,1,1,2,c$ + "Use cursor keys to scroll"
380 @char,254,16,72,1,1,2,c$ + "Press RETURN on Quit to exit"
390 for i = 1 to 7
400 read a$,fc(i),bc(i)
410 @char,254,20,(9 + i)*8,1,1,2,c$ + a$
420 next i
430 do
440 @windowopen,160,80,64,56,0:@clear,1,0,15:@windowclose
450 @windowopen,160,72 + (n*8),64,8,0
460 @clear,1,bc(n),fc(n)
470 @windowclose
480 @copy,2,(n-1)*40,0,40,40,2,248,88
490 @windowopen,240,80,56,56,0
500 @clear,1,bc(n),fc(n)
510 @windowclose
520 getkey z$
530 ch = instr(ch$,z$):if ch = 0 then 520
540 if ch = 3 and n = 7 then exit
550 n = n + n(ch)
560 if n < 1 then n = 7
570 if n > 7 then n = 1
580 loop
590 @text
```

Now let's dissect the listing. Line 100 sets up a random number function which will come in handy later on when I need to draw random points and lines for the "Dots" and "Lines" pictures. Line 110 sets up two color strings for later use with the @char command. Ordinarily, most BASIC 8 programmers place these strings directly in the @char command itself in literal form, however, since these characters appear as reversed letters I chose to define them as concatenated ASCII values since our laser printer has not yet been taught how to print reversed characters. The first string variable (c$) is used when we wish to print white characters with a black background, the second string variable (cq$) is used when I wish to print black characters on a black background (explained later). Line 120 defines ch$ as the valid keypresses for the scrolling menu, specifically the up arrow key, the down arrow key, and the return key in that order. Line 130 defines three numeric variables which will be used for housekeeping chores during menu scrolling. The numeric variable n keeps track of which menu item (1 - 7) is currently highlighted, and the numeric array variables are the offsets which adjust n up or down depending whether the up key or the down key is pressed.

Lines 140 - 180 do the setup work, telling BASIC 8 that we will be using a maximum of 16K of VDC RAM, for a 640 x 176 pixel screen which has a color cell size of 8 x 8 pixels and is to be erased and set with black as the foreground color as well as the background color. The setting of both the background color as well as the foreground color to black is an integral part of the smooth graphic flipping technique. This allows us to actually draw all of the shapes and pictures that are going to be displayed on the screen without them being seen since they have the same background color and foreground color. Keep in mind that even though this technique renders these pictures "invisible" the bitmap memory itself actually does contain the bitmap data which forms the images. The actual drawing of these images is done in lines 190 - 290. All of the images are drawn in the upper left hand corner of the screen in a rectangular area that spans from the upper left-most corner of the screen (0,0) to the pixel coordinates (240,39). Each image is spaced to begin every 40 pixels horizontally and is 40 pixels in height. This is done so that we can easily reference each image later algorithmically.

Lines 300 - 360 contain the data for the menu. Each data line contains the text for the menu entry followed by the numbers which represent the foreground and the background colors I want the entries to possess when they are highlighted. Lines 370-380 place the instructional text on the screen and lines 390 - 420 form a for..next loop which places the text for the menu itself onto the screen and stores the highlight colors for later reference.

The do-loop which spans from line 430 through line 580 is where the actual "work" of this program takes place. In line 440 we define a window which spans over all of the menu items on the screen and use the @clear,1 option discussed earlier. This is done to set the foreground color for the menu options to white and the background color to black, which in effect will "erase" any highlighting of the menu options. This is necessary because whenever either the up or down arrow keys are pressed, a new menu option needs to be highlighted and the previously highlighted option needs to be unhighlighted. By simply resetting the entire menu to its unhighlighted state we avoid the hassles involved with keeping track of which item was previously highlighted. The speed of the transfer of color information to video RAM via the @clear command and the relatively small size of the menu makes this approach feasible.

Lines 450 - 470 actually highlight the current menu option denoted by the value of the numeric variable n. Again we use the the @clear command with a value of one for the bitmap value parameter to make the color change necessary for highlighting the menu option without disturbing the bitmap data (text).

Lines 480 - 510 handle the display of the graphic images. The first step is to copy the proper "invisible" image (the ones drawn above with the same foreground and background colors) to the area just to the right of the menu. This is accomplished in line 480. But at this point the image itself is still invisible since the @copy command copies color information as well as bitmap information. Again, in lines 490 - 510 we use the @clear,1 technique, this time to render our image visible by replacing the black on black color information with the foreground and background color information of the highlighted menu item.

Finally in lines 520 - 570 we process key presses. Once the user presses a key it is stored in the string variable z$ in line 520. In line 530 the key that was pressed is compared against the string which contains the three valid keypresses. If the key the user pressed is not one of those three keys then program flow is redirected to line 520 until another key is pressed. Line 540 determines if the return key was pressed while the quit menu entry was highlighted, if this is the case the do-loop is exited and we are returned to the text screen. However, if the keypress was either the up or the down arrow the value for n is adjusted up or down accordingly in line 550. If the value of n becomes less than 1 or the value of n becomes greater than 7 then it is "wrapped" around to the opposite extreme of the range so that menu options are always highlighted properly.

The @clear,1 technique demonstrated in this article is indeed a powerful and I hope you will use in your programs. It has a variety of uses beyond the menu highlighting and pseudo-double buffering applications found in my program. One that sticks out in my mind is using the @clear,1 technique for animation on machines with only 16K of VDC RAM. Consider pre-drawing animation sequences with the same foreground and background color and then making them visible in sequence via @clear,1. For added effect color cycling could be employed to add depth or suggest great speed.

Most of our disk drives creep out of alignment, without us even realizing it. Until it becomes severe, the problem doesn't show up with program and sequential files, because they're pretty forgiving of slight mis-alignments. Relative files, however, need to be able to accurately go to a particular spot on the disk, and even a slightly mis-aligned drive can cause trouble if the disk was formatted on a different drive. If you move data and programs from drive to drive like I do, all of your drives should be in the same alignment, or you're likely to start corrupting your disks as each drive puts information where it believes is correct. And finally, of course, we all know how well copy-protection and fastload schemes work on mis-aligned drives!

Free Spirit Software's 1541/1571 Drive Alignment System is a do-it-yourself disk drive alignment tool for the Commodore 1541 and 1571 drives. One side of the included floppy disk holds the alignment software, while the other side contains calibration tracks. The calibration disk makes this product virtually pirate-proof. (Who would want to align their drive to some copyist's suspect copy?)

As a test for this review--and because I needed to, anyway--I used the 1541/1571 Drive Alignment System to align an elderly 1541 (with "1540" printed on the bottom) and a middle-aged 1571, and I was able to get both aligned accurately enough so that they would load Accolade software--and that's no small accomplishment! However, I was unsuccessful in aligning a third drive, which is an early prototype 1571. The Drive Alignment System software isn't necessarily to blame in this case; the tolerances in even a NEW Commodore drive are somewhat sloppy, and once a drive is well worn you can adjust back and forth 'till the cows come home and not get consistent results.

Home alignment programs like the 1541/1571 Drive Alignment System and others like it CAN work, at least enough to get you in the ballpark again, but we should realize they can't achieve near the accuracy of the regular technician's method of aligning a drive. (This involves a dual-trace oscilloscope and a special Commodore alignment disk with sine waves recorded on both sides of track 17.) The drives I aligned with the 1541/1571 Drive Alignment System showed up in "good enough alignment" on the scope (to quote a technician at an authorized service center), with the exception of the drive that I had aligned a full track off! (More on that, later.)

Part of the problem of home-alignment programs is that, without electronic hardware like oscilloscopes, the programs are forced to rely on the disk drive itself to tell where the head is. Even with the most wonderful of calibration disks, this can't achieve the necessary accuracy consistently because of the tolerances and software of the Commodore drives. The drive will try more than once to read data, and can pick up magnetic information from more than a half track away. And, because the information is digital, not analog, that "weak" information from a slightly off head will appear just as strong as if the head was in the dead center of the track.

There are real dangers with home alignment programs. Taking a disk drive apart and changing its adjustments is NOT a task to be taken lightly, yet the writer of an alignment program has no control over whether the person doing the job has the skills required! Because of the potential for making a real mess out of the drive, clear and foolproof instructions are necessary. Unfortunately, the instructions included with the 1541/1571 Drive Alignment System are, in a word, poor. Not that the information included is wrong, it's just that many necessary steps and precautions are left out.

[Editor's note: the documentation for this program has been improved since Bruce did his review. Several of the diagrams have been revamped and are now much more helpful. However, there still are several areas where the documentation is either sparse or ambiguous.]

More serious is the lack of any precautionary information on loosening the stepper motor itself. Commodore repair stations love home-alignment programs the way dentists love candy bars--because of the work it brings them! The most common reason for botched home repairs is hacked, stripped or otherwise butchered screws holding the motor in place. The 1541/1571 Drive Alignment System manual doesn't warn, you, so I will:

The screws holding the stepper motor in alignment in a Commodore disk drive are frozen in place with a locking-type "paint," which MUST be chipped or cut away with a hobby knife before you try and turn them. Making the problem worse is the fact that the screws are cheap, cast out of something that makes pot metal seem like tungsten steel. They are EASILY stripped, and you need everything going for you to get them out the first time. REMOVE the locking paint, and use a Phillips screwdriver bit that fits the screw slots PERFECTLY. Press HARD on the screwdriver to lock it into the screw, and turn the screw counter-clockwise to break it free.

(Make sure that your hand and fingers on the other side of the drive mechanism aren't touching anything fragile as you push against the screw. Pay particular attention that you're not stretching or distorting the metal "belt" that comes off the stepper motor and moves the head up and down; it's somewhat fragile and can't take much abuse.)
The manual doesn't tell you this either, but it's also far easier to work on the 1541-type stepper motor if you first remove the bottom shroud (if any) from the drive. While it is possible to get at the stepper motor screws through the access holes in the shroud, it's awkward to adjust the positioning of the motor with the shroud in place. Another reason for removing the shroud is that the screws holding it on are the same as those holding the stepper motor in place,

so you can replace one or both of the stepper motor screws if either of them was damaged in cracking them loose!

The manual DOES, thankfully, warn against touching the exposed circuit board with the metal drive mechanism, causing disastrous shorts and burning out chips and other components. According to the technician I talked with, this is the second most common reason home drive-aligners bring in their drives for servicing.

Here's the part I still have trouble believing: The manual also doesn't tell you how to align the drive, once you get it all apart. Honest, it doesn't! For some unknown and inexplicable reason, that information is on the program disk, and not in the printed manual. So, if you're checking or aligning a 1541 drive and need to review the instructions, you have to flip the disk from the calibration side to the program side, read the instructions and write them down on a legal pad, then go back to the alignment procedure!

(Mercifully, you needn't flip the disk when using a 1571 drive. Or do you? Depending on where I called the Align the Drive procedure from in the main program, sometimes I was asked to insert the Calibration disk, and sometimes I wasn't. Do you flip the disk over for the 1571, or not? The manual doesn't say, and the program is ambiguous.)

Those on-disk instructions are also disjointed, with the target figures for what you want to see on the screen separated by several screen pages from the procedure on how to arrive at them.

A final flaw of the 1541/1571 Drive Alignment System is the program design itself. The programmers chose to use the left arrow and up arrow keys to stand for "backward" and "forward" when paging through instructions, or when moving the disk head from track to track. Besides not being very standard keys for these purposes, those two keys are awkward to find and press, particularly so when your hands are already full supporting your drive mechanism and holding screwdrivers. All that aside, the REAL problem is that the alignment procedure practically ignores the fact that you're running it on a computer! You have to tediously move the head back and forth, using those darn arrow keys, and watch for certain numbers to appear. Remember that your hands are already pretty busy with the drive, and your eyes are locked on the stepper motor to see that you don't move it too far.

It would have taken practically no programming at all to have a simple keypress initiate a whole head-movement procedure, and a simple bar graph or sound effect would have been a lot more effective than looking for the number 17 and an asterisk! (Or was if the number 16 and an asterisk? I'd look it up, but that's not in the manual, either--it's on the disk.)

This is how I clumsily managed to align my drive a full track off the first try. You're supposed to adjust the stepper motor's positioning until certain numbers appear. (The numbers are different, depending on whether you got to the test track 16.5 from above or below it.) I blew it that time, probably because I couldn't read the instructions that I'd scribbled onto a legal pad from the disk instructions. I was pretty red-faced when the technician told me I'd done a good job of aligning the disk--to the wrong track. This, of course, wasn't REALLY the program's fault, but the poor documentation and user interface helped make the goofup possible.

I had the drive professionally aligned, took it home, and ran the 1541/1571 Drive Alignment System test disk on it. It said the drive was out of alignment. 'Nuff said.

Yes, the 1541/1571 Drive Alignment System works, within the limits of accuracy possible with this type of program. But the documentation and program need to be run through the polishing mill a few more times before I can recommend spending. 1541/1571 Drive Alignment System, Free Spirit Software and BSD Software, 905 W. Hillgrove, Suite 6, LaGrange, IL 60525, (312) 352-7323 $34.95

People who would have purchased **Twin Cities 128 T-Shirts** if they were alive today:

Leonardo da Vinci
Abraham Lincoln
Issac Newton
Elvis Presley
Albert Einstein
Martin Luther King
John Lennon
Harry Truman

People who would not have purchased **Twin Cities 128 T-Shirts** if they were alive today:

Adolph Hitler
Benedict Arnold
Caligula
Joseph Stalin
Attila the Hun
Henry VIII
Judas Iscariot

Where do you fit in?
See page 5 for order information.

One of the nicest things about owning a Commodore 128 computer is the immense amount of quality software available for it at a reasonable price. Of course some of the most reasonably priced software comes with no price tag at all, namely public domain software. Public domain software is software which for one reason or another the author has decided to allow to be distributed freely without requiring compensation for his or her programming and design efforts. The C-128 user community is blessed with a public domain library of literally thousands of programs, many of which are written by top flight software designers. From time to time we make an effort in columns like this one to tell you about programs that we feel are superior and should be sought out.

There are several ways obtaining public domain software. Perhaps the easiest way is to simply get it from your fellow C-128 users via user groups, local bulletin boards, or national telecommunications networks. Another method of obtaining public domain software is by ordering it from companies which market disks full of public domain software. Usually these companies put together collections of what they feel are the best public domain software packages and sell it for a "finders or collectors fee" which compensates them for the time and expense involved in putting together such collections.

Utilities
Disk Whiz v1.1 by Kevin Hisel
This program is truly a gift from heaven. Disk Whiz is a nice clean disk housekeeping utility that does it all: batch file copying, sequential file reading and copying, batch file scratching and more supports 1541, 1571, and the 1581!

Disk Doctor 128 v2.0 by Kevin Hisel
More than just a track and sector editor for advanced hackers, Kevin's program and documentation provide you with a first rate tool for learning about how Commodore disk drives work. Invaluable for repairing damaged disks. Supports 1541, 1571, and 1581 drives!

Crosslink by Miklos Garamszeghy
This freebie equivalent of Big Blue Reader does a fantastic job of transferring data from MFM disk formats such as those used by the IBM personal computers and various CP/M machines. If you are forced to use an MS-DOS machine at work and bring data home to your 128, you need this program.

RAMDOS 128 by Fred Bowen & Hedley Davis
This program fools your C-128 into thinking that a 1750 RAM Expansion Unit is a disk drive. Not completely transparent, when it comes to commercial software, but useful.

BASIC 8 Programs
BASIC 8 RTL by Lou Wallace & David Darus
This package allows you to RUN BASIC 8 programs regardless of whether you have purchased BASIC 8 from PATECH software. BASIC 8 owners don't need this, as BASIC 8 comes with the RTL and programs can be excuted from the BASIC 8 editor.

BASICwrite by Lou Wallace
This graphics word processor allows the creation of stunning video displays and printed pages. Complete with colorful documentation.

Towers of Hanoi by Willyum1
This game requires 64K of VDC RAM installed in your C-128 and lots of skill!

Mastermind 128 by Loren Lovhaug
A nice implementation of a classic game despite the author. It requires 64K of VDC RAM.

Games and Entertainment
128 Invaders by Mike Spice
Without a doubt, the finest public domain game yet to be implemented on the C-128. This adaptation of Space Invaders for the C-128's 80 column screen has it all. Supports both joystick and keyboard play and has adjustable speed by skill level just to keep you humble.

SDI by Ricky Vazquez
This clever C-128 remake of the arcade classic missle command takes on an eerie sense of realism in these days of "Star Wars". Let's all hope this game stays just a game.

SID/Maniac by Mark Purdy
This program allows you to play music files created with the enhanced Sidplayer music system for the 128. There are thousands of these music files in the public domain for a wide range of musical tastes, everything from pop/rock to classical music and original compositions. Playing these files on your computer are sure to bring a smile on your face, and the price tag is a heck of a lot cheaper than compact discs.

Sidplayer 1750(b) by John Pich
Like the SID/Maniac player above, Sidplayer 1750 allows you play selections from the myraid of Sidplayer files in the public domain, however this version allows the use of a 1700 or 1750 RAM expansion as a storage medium for the music files. Up to 333 music files (about 12 hours worth) can be placed into the 1750 and played in programmed sequences.

Isoplot by Bruce Bowden
If you ever studied three dimensional isometric surfaces in math class (I was never very good at finding the surface area of those puppies) then you may be interested in this well written program, which plots these surfaces on the C-128's 40 column graphics screen. The program includes an option to save figures to disk, or print them. It also features a few interesting sound effects.

01000110: Mouse driver for split screens from Fred Bowen: I get a variety of requests for C-128 information and routines, recently, someone out there was looking for a version of the 1351 port-2 mouse driver that worked with split screens. I played with this the other day and came up with some simple patches to the driver called M1351.128.BIN on the 1351 test/demo disk.

I suggest you BLOAD the above program, use the monitor to make these changes, and save the new driver under a new name. Okay? Let's start... First, let me fix a minor bug in that driver that does not relate to the split screen stuff. The byte at $188C should be $8E (not $8D). We want to restore the keyboard lines from the X register and not the A register. Now for split screens. At the locations below, make the indicated changes:

```
.A 184C JMP $1900   ;install patch
.A 184F LDA #$80    ;or $40 for port-1

.A 188F JMP $FA6B   ;continue IRQ

.A 1900 JSR $C194   ;do split, keyscan
.A 1903 BCC $1908   ;wrong IRQ, skip
.A 1905 JMP $184F   ;right IRQ, run
.A 1908 JMP $FF33   ;prend RTI
```

That's it. Save it with something like S"new driver",8,1800,1910 change the .BAS program to BLOAD"new driver", and you are set.

01000111: 1670 Noise Mystery solved..another one from Fred Bowen: I finally got a chance to look into the mysterious interaction between the 1670 modem and the 1571 disk drives. Here's the poop. There is a slight difference between the 1571/1581 circuits and the 1541 circuits, more or less what we've been saying all along. There is also a difference between the C128 and C64 user port. Both of these contribute to the situation. The situation, for all you first time readers, is what happens when you try to use the modem when you have one or more 1571 drives attached to the serial bus that are turned off- you usually hear what has been described as a "frying egg" sound, and the modem appears not to work. The solution (and this is still the case!) is to turn the drive on or remove it from the serial bus, like you're supposed to. Okay. The reset pin of the user port (pin 3) on the C128 has a buffer on it to help protect the sundry devices attached to the serial bus and user port. This protects the system from all you folks who like to plug and unplug modems and powered-up disk drives on a powered-up C128. Anyhow, the buffer requires a pullup which does not exist on the C64. Attach the 1571, and that circuit, when not turned on, pulls the reset line low, and guess what? The modem on the user port sees an endless reset.

I am in no way recommending this, but if you want to hack your system here's what I would do (this voids warranties, folks, and is described here for you reading pleasure). Cut the trace at pin 3 right at the user port. Jumper pin 3 to pin 11 of U57 (in can near VIC chip- probably best done underneath the board). What this does is move the user port reset line to the other side of the buffer & pullup (the serial bus reset line is still protected).

01001000: Quantum Link Reply Saver from Loren J. Lovhaug: Quantum Link is a great telecommunications network for Commodore owners and its custom terminal software make it extremely easy to learn and use. However, the same custom software that makes Quantum Link easy to use also limits the power and flexibility you have when using Quantum Link. One of the most glaring weaknesses involves your ability to buffer text for offline reading or manipulation. The Quantum Link software has a very small buffer they call the message text area, which you can save to disk when reading messages, articles, and email. However, this area gets cleared each time you read a new message, articles, or piece of email. This means that in order to store a multiple part article or message that has replies you must save each individual message text area before advancing to the next part of the article or message. Ordinarily this means that after a typical session where you might save 10 or 15 message text areas to disk you will have 10 or 15 separate files (one for each message area saved) on disk. This is makes the reading or manipulation of this data extremely tedious. Fortunately there is a trick which allows you to append message areas to previously saved files, thereby avoiding the multiple file dilemma. Here is how it works: When you choose to save the current message text area to disk (usually by pressing the F3 key or when reading email, by selecting the F7 key) you are asked to provide a filename for the sequential file that the data is about to be saved to. Normally you would provide a unique filename for each message area; however, if you specify a previously saved sequential file followed by a comma and the letter a, the current text area will be automatically appended to the previously saved file. For example, suppose you were reading a message on converting graphics files to various formats and there were several replies you wished to save with the message for later reference. After reading the first message you might save it to disk as CONVERTMSG and go on to read the first reply. After reading the reply, save it using the following filename: CONVERTMSG,A. This will append the reply to the previous message automatically. By the way, this trick will work with many other programs.

The 1581 disk drive is perhaps the best peripheral device Commodore has given the C64-128 community. It offers both speed and large storage capacity. Considering the current cost of hard disk drives, I like to consider the 1581 my "mini hard drive". Large capacity drives such as the 1581 and hard disks require a way to organize the files by way of partitioned sub-directories. Unfortunately, Commodore's DOS, which is "carved in ROM", does not allow a simple, direct mode command that accesses the 1581's sub-directories.

This seemed the perfect situation to alleviate with an addition to the direct-mode command set. I have prepared a machine language wedge that adds a command to access the 1581's sub-directories directly. The syntax is: cd/0:partname to open the partition called partname, and cd/ for the root directory (cd for "change directory). The default device that is accessed by this command is device 8. In order to access a 1581 set as device 9, type "u9". I added this little command to allow this wedge to be used without assembling two different versions (alternatively, if you use the 1581 exclusively as device 9, you may assemble the wedge with 9 as the default drive. Typing "u8" will return the access to device 8.

The method I've used to make the new command is called the "syntax error wedge". Every direct mode command is evaluated for correct syntax. If an error is found, the program is diverted to the Basic error handler vector at $0300 with the x register loaded with the error number. My wedge re-writes this vector to point first to a bit of code that ensures that we're in bank 15, then jumps to the wedge's main routine. The error number is stored, and compared to the syntax error number ($0b). If not, the wedge returns to the Basic error handler routine at $4d3f. If the wedge does detect a syntax error, the buffer where the disk command is set up (com'buf) is initialized with zeros, and the command input buffer at $0200 is parsed to see whether the letters 'cd' appear. If so, the input buffer is read and stored in com'buf. The wedge writes a '/' ($2f) to the command buffer where it belongs (the input buffer for some reason reads a '/' as $AD--I don't know why...) and jumps to the send'com routine.

Sending a disk command works similarly to opening a sequential file for reading. The protocol involves a series of kernal jumps. First call SETBNK ($ff68) with the accumulator containing the bank number for the data (in this case there is no data to send, so a 0 is loaded into the accumulator), and .x register loaded with the bank number where the filename is stored. Next, call SETLFS ($ffba) with .a loaded with the logical file number, in this case we want the command channel number ($0f), the .x register containing the device number, and .y containing the secondary address ($0f). Notice the similarity to the OPEN 15,8,15 command from Basic. Next call SETNAM ($ffbd) with .a loaded with the length in characters of the command you're sending, .x loaded with the least significant byte of the address of the filename's first character, and .y loaded with the most significant byte. Note that if you plan to open a channel and send text in memory to a printer, set the file length to 0, x and y are irrelevant.

By then, most of the work has been done, all that remains to be done is a call to OPEN ($ffc0). This essentially sends the command. Once this is complete, load the accumulator with the file number ($0f) and call CLOSE ($ffc3). Then you are ready to exit the wedge--but first one address, the system call to the error routine, is removed from the stack. The wedge then jumps to the ready prompt.

Of course, this program is a wedge that can be used while other programs are running. I BLOAD the wedge and install it with SYSDEC("1500") when I use my editor/assembler combination (Power Assembler/Buddy). Then I can easily go to the source sub-directory, where all my source code is stored.

```
*= $1500
.obj "@0:part.obj"

; Kernal jumps

mmu'reg = $ff00
setlfs = $ffba
setnam = $ffbd
open = $ffc0
close = $ffc3
setbnk = $ff68

; Vectors and Equates

ierror = $0300
error = $4d3f
swtch'bnk = $03e4
chrget = $0380
buf = $0200
ready = $4d37
zp = $fa

; install the wedge!

install = *

 lda ierror
 cmp #$e4              ; is wedge already installed?
 beq +

 lda #<error           ; read the error vector and
 sta error'rtn         ; store for eventual jump there.
 lda #>error
 sta error'rtn+1

 lda #<swtch'bnk       ; load the error vector
 sta ierror            ; w/ swtch'bnk address
 lda #>swtch'bnk
 sta ierror+1

 ldx #$07              ; move switch code to $03e4
- lda swtch'code,x
 sta swtch'bnk,x
 dex
 bpl -
+ rts

error'rtn .byte 0,0

swtch'code = *

 lda #$0               ; code to be loaded to be $03e4
 sta mmu'reg
 jmp error'eval

; the preceding code to be moved to $03e4

error'eval = *

 stx error'num         ; save error number
 cpx #$0b              ; is it a syntax error?
 bne err'cont          ; if not then continue err routine

 jsr initialize

 lda buf               ; read the input buffer
 cmp "u"               ; is it a 'u'?
 beq dev'chan          ; if so change device #

 lda buf               ; read the input buffer
 cmp "c"               ; is it a 'c'?
 bne err'cont          ; otherwise continue
 lda buf+1             ; next char
 cmp "d"               ; is it an 'd'?
 bne err'cont

 ldx #0                ; initialize counters
 ldy #0
 beq +

- iny
 inx
+ lda ($3d),y          ; read command
 beq +
 sta com'buf,x         ; and store in buffer
 bne -

 continue = *

+ stx nam'length       ; store name length
 lda #$2f              ; write a $2f to com'buf
 sta com'buf           ; for '/'
 lda #0
 sta mmu'reg
 sta $7a
 jmp send'com

nam'length = *
 .byte 0
error'num = *
 .byte 0
com'buf = *
 .buf 21
dev'num = *
 .byte $8

err'cont = *          ; return to err routine

 lda #$0
 sta mmu'reg
 ldx error'num        ; recall error and continue...
 jmp (error'rtn)

dev'chan = *

 lda buf+1            ; read the device # from key buff
```

1581 CD Wedge Continued:

```
and #%00001111          ; make it an 8 or 9
sta dev'num             ; and put it where it counts
 jmp done

send'com = *

 lda #0                 ; name found in bank 15
 tax
 jsr setbnk
 lda #$0f               ; set 15,8,15
 ldx dev'num
 tay
 jsr setlfs
 lda nam'length
 ldx #<com'buf
 ldy #>com'buf
 jsr setnam             ; send command
 jsr open
 lda #$0f
 sec
 jsr close

done = *

 pla                    ; remove one address and return
 pla                    ; to ready prompt!
 jmp ready

initialize = *

 lda #0                 ; initialize command buffer
 tax                    ; by writing 0's
- sta com'buf,x
 inx
 cpx #21
 bne -
 rts
```

CP/M COMAL Continued
run at a later time. As the demo version is nearly free (the COMAL Users Group USA owns the copyright, but has placed it on bulletin boards including QLink and allows it to be given away) it is ideal for anyone wanting to try COMAL before purchasing the full version. It is also perfect for schools in that the teacher can write programs on the full version and let the students use the demo version to run the programs. The students can also write short program assignments with the demo version to hand in to the teacher who can run the programs from the full version.

CP/M COMAL was the first version of COMAL in the United States to offer a compiler. Yes, you can attach a RUNTIME system to your CP/M COMAL programs to allow them to run on CP/M machines without first having to load in the COMAL language.

To order CP/M COMAL or to receive more information on any COMAL product, please contact the: COMAL Users Group, USA 6041 Monona Drive, Madison WI 53716

The Back Page

So here we are at last, the back page of another issue of Twin Cities 128. The other day I stopped to take a look at this endeavor *in toto*. I shocked to find that when you take all 19 issues of Twin Cities 128 that have I have published so far you get something that is about as thick as a phone book for a good sized city! I think this issue again reflects another giant step in our evolution. The observant folks out there will notice that I figured out how to do half-tones with the laser printer so that I could add the 15% gray shade title bars to each page..all with just Paperclip III and Pocket Writer 2 and lots of playing around with the laser during the wee hours. The next step: More sophisticated graphics and text interaction. Mostly likely this will involve the hacking of some kind of pseudo-desktop publishing utility..stay tuned.

But this issue was a little different that the past three or four issues of Twin Cities 128. Oh sure, this issue still included the obligatory cheap shot directed at Amiga owners (all in fun guys, honest), but beyond the continual improvement in appearance and a little more attention directed upon spelling and grammar this issue is probably the most balanced issue we have published in a while. By balanced I mean this issue has a fairly equal ratio of technical vs. non-technical information. This balance is a difficult one to achieve. First, because once you reach a certain level of computing expertise it is difficult to write about or maintain an interest in topics which might appeal more to beginners more than more advanced users. Secondly, because teaching, be it in a classroom or on the pages of a magazine, is a difficult and demanding task. I am indeed interested in hearing what you think about this issue, especially as far as the balance between technical and non-technical articles is concerned.

Another improvement found in this issue was the addition of some new and talented faces which joined forces with myself and our regular crew. I am particularly pleased to have material from someone as talented as Bruce Jaeger. Bruce is a good friend of this effort and is a professional writer who has been published by a hoard of computer magazines. John Clark made it two in a row with his excellent CD Wedge article, and Richard Bain's COMAL review is first class as is Bill Nicholson's. And of course I can't praise enough Fred Bowen for his efforts for Twin Cities 128 and the C-128 community as a whole. The addition of material from these individuals was timely since three of our regulars, Miklos Garamszeghy, John Kress, and Avonelle Lovhaug, were unable produce articles for this issue due to other commitments. But fear not, I am hopeful that issue #20 will feature articles from all three of them!

Speaking of Issue #20, here is what is tentatively in the works for next time:
An evaluation of the Fontmaster/Spellmaster 128 upgrade from Xetec (we had hoped to bring this one in for this issue but things just did not work out, An Evaluation of The C-128 Helper from SSI, and A feature on using and getting the most out of GEOS 128 and its add-ons. Plus our usual assortment of programming techniques, application ideas, and technical info.

Till next time...

Loren