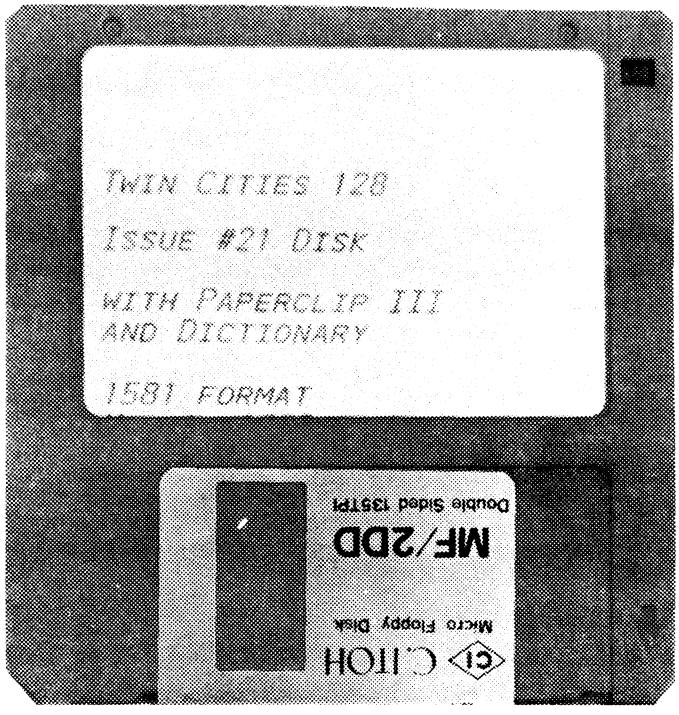
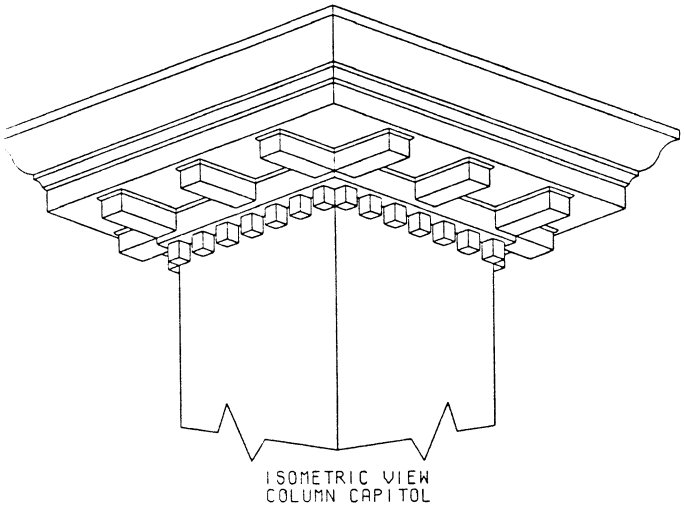
TWIN CITIES 128

THE COMMODORE 128 JOURNAL

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



ISSUE #21 $2.50



IN THIS ISSUE:

RUMOR/OPINION/MAYHEM

by Loren Lovhaug

When the going gets weird...turn Pro

Or maybe a more appropriate start to this column can be stolen from A Tale of Two Cities, "*It was the best of times, it was the worst of times*". This is because in many ways that well worn opening aptly describes what has happened to me and Twin Cities 128 since our last issue hit the streets.

There is no doubt that things have never been better for me as a computer journalist. Not only has the readership of Twin Cities 128 grown significantly, but our compendium book (the best of issues #1 - #18) is selling so well that we are having a difficult time keeping it in stock. Sources at Quantum Link tell me that the C-128 area we manage on Q-link is one of the most successful areas online. Of course, the shear volume of electronic mail I receive and the number of public postings in our message areas indicated we were doing well long before we received any official declarations. But beyond the recent successes of Twin Cities 128, my writing services are now peaking the interest of many other outlets in the Commodore world. Those of you who read a variety of Commodore publications probably have noticed my byline appearing on glossy stock. This would seem to validate the work Avonelle and I have been doing over the past two and a half years, suggesting that this effort has been something more than just an ego trip or a desperate attempt to provide C-128 owners with information other outlets were not providing. I also think the fact that other publications are interested in my work is a positive sign for C-128 owners that they have indeed not been forgotten, and that those in editorial positions are beginning to take note of the size and needs of the C-128 community. Another positive aspect of my work being published by other outlets is the simple fact that the financial compensation received from such endeavors helps make this unorthodox career of computer journalism more viable.

But I would be a liar to suggest that all is well. The truth is, this issue should have been in your hands a long time ago. Ordinarily, when a company the size of Twin Cities 128/Voyager Mindtools has trouble getting product out the door it is a sure sign that they are having financial difficulties. But as the previous paragraphs would seem to indicate this is not our problem. Our problem is something far more subtle than plain red ink, but something that is no less a threat to our ability to serve you. Business professors have coined the term "*Hypergrowth*" to describe this phenomenon. One of the classic stages of hypergrowth is when the administrative duties of a company grow to a point where they become so overwhelming that they begin to effect the ability to create or distribute the product itself. In our case, administrative tasks such as filling book and single issue orders, recording new subscriptions, and working through the myriad of governmental paperwork that are thrust upon people who deviate from the 9 to 5 norm takes its toll. Add to this the fact that our database is growing beyond the capacity of even 1581 disks and that there are only 24 hours in a day and you have a recipe for trouble.

Of course the paradigm for success for a start-up publication in the Commodore world is INFO magazine. When INFO reached the point we have arrived at they employed the classic solution to this problem: simply hire other people to handle some of the administrative work load, thereby freeing the principals to go about the actual work of producing product. The key to this solution is obtaining enough commercial success that you can actually sustain someone on a "real" payroll. So far in this aspect, our operation has resembled the amateur rather than the professional. At this point we easily bring in enough money to print and mail out our publication, buy whatever hardware we need to assist us in production, and pay our authors a small gratuity, while periodically giving me a paycheck, however at this point the resources are not in place to hire clerical staff or an office to put them in.

So why am I telling you this? I can assure you it is not a ploy for your sympathy, or an excuse for the delay between issues. Neither of those in reality would be all that helpful. I am explaining all of this to you because I quite frankly am stumped. As our readership has continued to grow, the delay between issues has also grown and I have been unable to come up with a way to reverse the trend. So I am asking you, our loyal readers for your ideas. I have been quite impressed in the past with the advice and suggestions you have rendered and I am hopeful that somewhere within our collective minds we can come up with some kind of scheme for getting this thing back on a regular schedule.

While we are on this theme, at least one reader has written in to tell me that in his mind it did not matter how long it took us to get an issue out, just as long as we keep producing issues. While I am sure this person speaks for many people who are just appreciative that someone is doing a C-128 exclusive publication, I am also quite sure that there may be some people who are angered or are upset. For those of you who may be upset, I want to remind you that if for any reason you are unhappy with Twin Cities 128, we will cheerfully refund the unused portion of your subscription.

On a more cheerful note, at least one expert in this field has told me that he thinks our production delay dilemma will soon take care of itself. He bases his opinion on our rate of growth over the past year and the amount of exposure we are beginning to receive. It is his conviction that with a product that is being well received and that is showing no signs of losing quality or demand, this stage of hypergrowth will pass and we will be able to expand and afford the clerical help we need to maintain a healthy schedule. He also points to the fact that INFO also went through the same kind of production schedule lag just before they really burst into the "big-time". If he is correct, than maybe this discourse will have been for naught, but I just wanted you to know that I am in fact concerned about the problem, and am looking for ways to correct it.

The biggest and the best

A couple of quick notes about this issue of Twin Cities 128. This issue of Twin Cities 128 is the biggest issue we have ever put together. I also believe it is the best looking issue we have ever put together. With this issue we have really concentrated on the details. Beyond ordinary, but crucial details such as spelling and grammar, Avonelle and I have been particularly careful about making sure that our layout was more attractive and easier to follow. You might note that when an article is continued onto a following page an effort has been made to direct the reader to that page.

Also, over the past couple of issues I have been playing with the command language of our laser printer to create more sophisticated look. Some of you have asked about how I do such things, so here is the story. First I drew the C-128 and the bitmapped text in the left corner using Basic 8. I then diverted printer output from Basic 8's @hcopy command to a sequential disk file and appended to that disk file the proper control codes from the laser's printer control language to position and draw the black line at the top of the page. Then I used the external file option available in Pocket Writer 2 to send the page header stored in the sequential file to the printer at the beginning of each page. Then using Pocket Writer's redefine character (macro) option, I was then able to position the page number and article title using the laser printer's control language and print the article text for each page. It sounds more complicated than it really was, but the result is that both the header and the text are positioned automatically by the laser printer without any external cutting or pasting.

C-128 Price & Progress Report

by Loren Lovhaug

Burning The Midnight Oil
Mountain Wizardry Software has just announced its Midnight Assembly System for the C-128. It functions in 80 columns, contains a full screen scrolling text editor, and also its own 1700/1750 RamDos for fast operation with the RAM Expansion units. It also allows multiple statements per line of text and free format text entry. Although we don't as yet have an official price, you can contact Mountain Wizardry Software, P.O. Box 66134, Portland, OR 97266, (503) 265-2755 for more information.

Teach An Old Dog New Tricks
Release 4 of WordStar is now available from MicroPro. They claim to have added over 120 enhancements from the last release, while still maintaining the previous "look and feel". Enhancements include an expanded dictionary for the spell checker, a facility for finding rhyming words and anagrams, improved print capability, on screen boldfacing and underlining, and increased printer support. Wordstar 4 runs under CP/M and is available for $89.00, and includes a 90 day warranty period with free technical support. Contact: MicroPro International Corporation, 33 San Pablo Avenue, San Rafael, CA 94903, (415) 499-7676.

Fonts For The Master
For those of you who use Fontmaster 128 and are looking for more fonts and borders, you will be interested in Suzart, a collection of fonts and borders specifically for this program. The fonts are of the "pictograph" variety: using a picture for a character instead of the character itself. For instance, one particular set prints an apple for the letter "a". Some of the pictographs include fish, disks, computers, dogs, trains, furniture, etc. The borders are also quite good. For order information, write to: Suzart, P.O. Box 410852, San Francisco, CA 94141-0852.

When The Doctor Is Out, Call A NURSE
Free Spirit Software has released Nurse, a comprehensive diskette sector editor for the Commodore 64 and Commodore 128 computers. On the C-128 Nurse operates in 80 column FAST mode. Nurse supports the 1541, 1571, and 1581 disk drives as well as many older PET and Commodore compatible drives. Support is provided for drives 0 and 1 on dual drives and disk devices from 8-11. Nurse edits in hexadecimal, decimal, or text, recovers damaged data by reading through errors, scans a disk or file to find errors, searches the disk or file for strings, does operations on segments of a sector such as fill, and, or, add, subtract, exclusive-or, disassembles the current byte, copies from one sector to another or even another disk and performs many other disk editor functions. Also included with the Nurse package is a variety of useful, user usable programming routines, and some very slick utilities including a routine that enhances the C-128's built-in monitor by adding scrolling "on the fly" disassembly. Nurse is retails for $34.95 and is available from: Free Spirit Software Inc., 905 W. Hillgrove Suite 6, La Grange IL 60525, 800-552-6777 or 312-352-7323.

Book Balancing Act
Professional Software has released a C-128 version of their Pro Tutor Accounting, designed to help people master basic accounting principles. The tutorial package carries a list price of $99.00 with an educational version including many valuable teacher utilities available for $250. In addition to Pro Tutor Accounting, PSI is currently developing similar tutorial programs in Language Arts, Physics, Consumer Math, and Pro Tutor Accounting II which will deal with more detailed accounting situations. For more information, you can call them toll free at: 800-343-4074 or direct at 617-444-5224

If Ivan Boesky Had Used A 128...
Strategy Software has announced the release of The Strategist, a Commodore 128 market timing program for investors in stocks, bonds, mutual funds and commodities. It allows the user to chart issue prices against one or several indicators so that he/she can visually pick the one or ones that seem to call the turns in the market, then use it or them to time his trades. Strategist takes the direct approach to the investor's question of which strategy is best. Starting with a historical quote file for the issue of interest and a strategy specified by the user, it goes through the file making realistic simulated trades to see how much the strategy would have paid in real life. Then Strategist can also take this approach step farther: starting with the user's initial strategy, the program goes through the historical file over and over, varying the strategy slightly each time until it arrives at a strategy that gives the optimal payoff. The Strategist package includes the main program, Strategist, and two supporting programs. One creates historical files and the other tracks week-to-week price activity (in the present) for trading signals. The package also includes a telecommunications program. The Strategist includes master disk, an 84 page manual, and 90 day money back guarantee for $29.95. Interested investors can order the Strategist directly from: Strategy Software, Box 14-2403, Anchorage, Alaska 99514

Here Comes The Stuff
Busy Bee Software is proud to announce the release of "The Write Stuff 128" (TWS) the C-128 version of their popular C-64 mode word processing software. For just $29.95, TWS 128 looks to be quite a bargain, especially since it boasts most if not all of the features found in top-flight C-128 word processing products such as Pocket Writer 2 and PaperClip III. Here is just a subset of the seemingly endless list of features: Multiple text areas (up to ten), split screen text editing of single and multiple text areas, 1700/1750 RAM expander support, user defined keyboard macros, printer macros of up to 32 ascii characters, automatic multiple column generation, built-in outliner, non-copy protected (can be placed on 1581 disk), automatic carriage return stripper, user modifiable (by assembly language programmers) complete with patch area and jump table, 1581 sub-directory support, multi-function calculator, and mail merge. Author, R. Eric Lee also indicated he is working on a spellchecker/dictionary lookup module and a Quick Brown Box version of TWS 128. So far I am impressed, look for our review in issue #22. Busy Bee Software, P.O. Box 2959, Lompoc CA 93438, 805-736-8184.

CP/M gets a boost with QBB
Miklos Garamszeghy (you know the guy that keeps us and Transactor busy formatting lots of text) has created a utility that allows you to use Brown Boxes Inc.'s wonderful 64K battery backed CMOS RAM cartridge (Quick Brown Box) a RAMdisk under CP/M. Even better is the fact that you can still use the 1750 RAM Expander as a volatile 512K RAMdisk and QBB cartridge as a 64K non-volatile one (provided of course you have a multi-slot expansion bus). For Mike's next trick, he is going to see about getting CP/M to boot right off the cartridge itself. Meanwhile yours truly is going to work on some nifty native mode QBB applications (as soon as I get caught up with the 100 other things on my agenda). Anyway, if you are interested in obtaining more information on the QBB CP/M patch, or the cartridge itself I strongly urge you to write: Brown Boxes Inc., 26 Concord Road, Bedford MA 01730.

Page Illustrator Gets Improved
PATECH software's recently released graphics program, Page Illustrator has been upgraded. Version 1.1 is being offered free of charge to all owners of the original package. To receive the upgrade, simply return your original disk to PATECH in the disk mailer provided and PATECH will send you the new upgrade. Alas, still no sign of Page Builder 128, but whose fault is that? PATECH Software, P.O. Box 5208, Somerset NJ 08873.

HOME DESIGNER 128 REVIEW

Review by Bruce Jaeger

Home Designer is a CAD (Computer Aided Design) program for the C-128 that's not going to get many decent reviews. But wait a minute! I'm speaking from a professional writer's standpoint, not a CAD user's. Home Designer is a powerful, complicated program that's difficult to get to first base with; you've got to be willing to dedicate some time to it. Most of the reviews you'll see (and the ones I've seen already) just re-write the press release a bit, say "Gee! This is powerful!" and reprint some of the sample printer outputs. THAT is not a decent review, which explains my opening remark. (And I bet I had programmer Russ Kendall squirming there for a minute!)

CAD programs should not be confused with drawing programs. A real CAD program gives you computer versions of (at least) the following drafting tools: ruler, triangle, protractor, compass and architect's scale. For input devices, Home Designer supports joysticks and either the 1350 or the proportional 1351 mouse--I've tried them all, and definitely recommend the latter! There are no other requirements to run the program that a normal system isn't likely to have, although the Abacus BASIC-128 compiled programs are kind of large, and I wouldn't recommend using a slow 1541 disk drive unless you're quite young and have a lot of life ahead of you. No use is made of RAM expansion at present, and the program runs in graphics (40-column) mode.

A decent CAD program should produce output (either printer or plotter) that is independent of screen size or resolution. (There are some "CAD" programs for Commodore computers where the lines and dots on the screen are the end result, and that give you a screen dump for your output--take THAT to a machinist sometime, and see what kind of reaction you get!) Home Designer, on the other hand, is an object oriented drafting program. That means that the individual elements of the drawing are made up of lines, arcs (or circles) and text. This makes it easy to change the scale of a drawing, or to "zoom" in on part of a drawing to see it in more detail,

Home Designer can either drive a plotter, or produce "C" size (16x22) drawings in one or two passes on a variety of dot matrix printers. (With printers that can handle only 8.5" wide paper, you have the option of either reproducing your design half-size to fit on one page, or of making two separate drawings that you can paste together for your final 16x22 version.)

I must not try and fool you into thinking I'm any kind of a CAD expert, but I am exposed to them in the line of business. (I'm a freelance technical writer.) I've seen PC-based and VAX based CAD programs in use, work with their output all the time, and even wrote my own simple CAD program to do accurate plan view ("top view") drawings of machines. Home Designer has many of the same features, and in fact the same sort of operating environment and syntax, as the professional systems I've seen. (It's not surprising, as programmer Kendall works with state-of-the-art equipment in his "day" job!)

Here are some of the more powerful features and capabilities of Home Designer:

OBJECT-ORIENTED

As mentioned before, the individual elements of the drawing are made up of lines, arcs and text. If the detail is too small to see in the full-size drawing, you can see it by "zooming in" and getting a more close-up view.

5 LAYERS

Your drawing can have from one to five "layers" or levels, which you can display on the screen all together, or with different layers left out. This means you can have your main drawing on one layer, your dimensional data (the little arrows and lines and numbers that tell how long an object or dimension is) on another level, and plumbing or wiring or text or what-have-you on another. By choosing to not display the text or other at's worth it, as redraws can take a frustratingly long time--more on that later.) By the way--text can also be turned off separately.

You can also print or plot the different layers all together or individually. Another good use for layers is for users with access to multi-pen plotters--you can have each layer in a different color.

LIBRARY FIGURES

You can convert any drawing to a *LIBRARY* figure. This means that you'll be able to call up from disk and *INSERT* the drawing into another drawing, converted to the scale of the new drawing--kind of like a word processor for drafting! A library of windows, doors, sinks and other household-type stuff is included.

SCALING AND "REAL WORLD" VALUES

Home Designer lets you draw lines and arcs by inputting the actual dimensions for a line length or a radius, so you don't have to do all sorts of mental conversions all the time. ("Let's see--I'm drawing this 1/2" to a foot, so a 126.5" line would be, um, 126.5 times .5, divided by 12, um, 5.27 inches. I think.")

With Home Designer, I did the above by setting the scale to 1/2" to the foot (by entering *SELECT SCALE .5" = 1'*) and by inserting a line 126.5" long (with the command *INSERT LINE HORIZONTAL 126.5*").

By the way, you don't have to type those long words all the time; only the first three letters matter. "*SEL SCA*" and "*INS LIN HOR*" work just as well. The latter often-used command is even on a function key!

DRAWING COMMANDS

Almost everything you do with a CAD program starts with an *INSERT*. You insert lines, circles and text. Lines can be drawn either horizontally, vertically, or at specified angles; a drawback of Home Designer is that you can only used the scaling LEN modifier with horizontal or vertical lines--I could not have drawn that 126.5" example line at a 45 degree angle.

It's an added step, but I could take that horizontal 126.5" line, then use the *ROTATE* command to put it at a 45 degree angle. Then, I could use *TRANSLATE* to move the line (or any group of lines and arcs, up to 50) anywhere I wanted it on the drawing. Other drawing commands are *MIRROR*, to make a mirror image, *TRIM* (to lengthen or shorten a line to fit the rest of my drawing), *DIVIDE* (to divide a line into two, perhaps more manageable sections), *CHANGE* (to change the parameters of something I'd drawn previously), *OFFSET* (to create parallel lines), *VERIFY* (get text information about where a line is--start points, end points, layer, etc.) and finally, *DELETE*, a command that you'll be using a lot as you learn to use Home Designer.

You can locate lines on the screen/paper by using absolute coordinates from the keyboard (X1Y2 would mean 1" from the left, and 2" from the bottom), by digitizing a point or points with the mouse or joystick, or by "snapping" onto the end of a previously-drawn line, by typing END or NEAR as part of your line command, then positioning the cursor on the end of the desired line and pressing the mouse/joystick button. I'm not sure about the accuracy of some of the snaps. If you'll look at the sample vial filler drawing, you'll see several little "hanging" ends on corners; these are all at the ends of lines I'd snapped onto. (I did the snapping at different levels of zoom, which may have affected things; however, since this is object-oriented, it shouldn't have mattered.)

HOME DESIGNER 128 REVIEW

Continued from Page 4

Home Designer is undeniably powerful, especially considering the price, and the limited computing resources of the Commodore 128. You pay for this power in terms of complexity and speed. As far as complexity goes, the drawing commands have (to me, at least) non-intuitive and seemingly inconsistent punctuation requirements. You'll be constantly flipping pages in the manual for quite a while.

The manual itself is adequate, but just so. Part of the problem is simple typography you can barely tell the bold-faced entries from the regular type, and there's nothing to break up the pages. Big, fat headlines are desperately needed to make looking up commands easier. (My manual became twice as useful after I went through it with a highlighting pen.) The index is incomplete, and the tutorial teaches you how to type in individual commands, but needs to cover some groundwork in how to start a design, where and why to locate the first lines, etc.

Speed? I've seen CAD programs running on XT's, VAX's and 80386-based machines, and no matter what kind of whiz-bang machine you have, redrawing a screen is never fast enough, especially once you get used to using the program! Home Designer's screen must be redrawn whenever you shift *ZOOM* levels, and, even though FAST mode is used, it still takes long enough to get frustrating. It would probably be faster if written in other than compiled BASIC; frankly though, there isn't a decent alternative language for a program of this size and complexity on the Commodore 128, especially when you consider trying to get a product to market quickly enough, and the economic realities of how many copies a complex CAD program is likely to sell.

PRINTER and PLOTTER OUTPUT
Home Designer currently supports only the toylike Commodore 1520 plotter, and the Hewlett Packard 7470A (or equivalent). I have a surplus Apple plotter which is not compatible with either of the above, so I wasn't able to test plotter output. I DO have access to HP plotters, and I suspect many Home Designer users will as well; I wish future versions of Home Designer would include an option to divert the plotter output to a disk file, so that we can transfer it to an IBM-compatible disk and dump our drawings to a plotter using other than a C-128. Home Designer also needs to support the "standard" type of RS-232 interface for users who do have plotters they can connect to their 128's. At present, only the serial-bus type of RS-232 interface is supported.

Although Home Designer's plotter support is disappointing, its dot matrix printer support is outstanding. When printer in "high-res" mode, you'd swear that some of the drawings came from a plotter! Even the "draft-mode" printouts are good. The circles are actually round, and the lines (even the angled lines) are straight, without the dreaded "stairstepping." Not all of the supported printers work as well as others, of course. On the negative side, it takes a long time to do a high-res printout; a half hour or so. So use draft mode for all your test prints.

While Home Designer supports a variety of printers, among them the Epson RX (which most dot matrix printers emulate), I could not get it to work with my "ancient" Epson MX-80 with an early version of Graftrax. On the other hand, the new WordPerfect 5.0 for IBM computers doesn't work with it, either, so Home Designer shouldn't take too much blame.

I finally borrowed a Star NX-1000, and, by ignoring the STAR SG10 and SG15 drivers and calling it an Epson, I was able to get the print program to work except that I had to flip the dipswitch inside the printer to turn on Auto Linefeed. The moral is that, unless you're luckier than I, you're going to have to spend some time fiddling with printer drivers and dipswitches (on both your printer and your interface). A few examples are given in the manual, but it's impossible to cover even a small percentage of the many possible combinations users might have.

Okay, now that I've discussed what the program does, here is what I'd like to see improved:

A LARGER DRAWING AREA ON THE SCREEN
The right inch or so of the drawing area is lost to a list of menu commands, which you can "pull down" with the mouse. This is a wasted cuteness, because almost all commands require some extra typing of parameters at the keyboard anyway. I'd rather have the space for drawing.

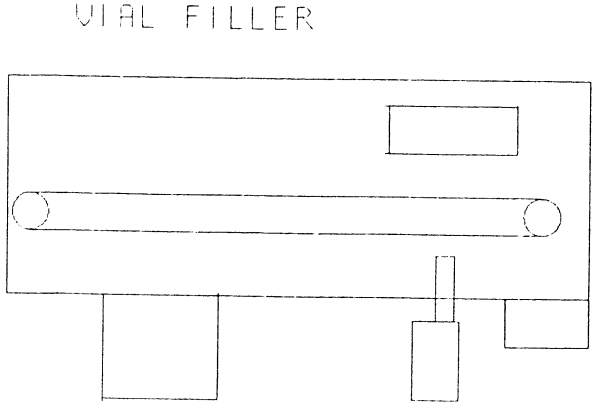
SPRITE CURSOR COLOR
At present, everything is white on black, which is all right. But the mouse-controlled crosshair "cursor" and line endpoints are also white, making them unnecessarily hard to pick out sometimes in a crowded drawing.

SOURCE CODE FOR OUTPUT DRIVERS
Maybe I'm getting greedy, but if I had these I could do the modifications to get my Apple plotter to work, make the save-to-disk modification I want, and get my MX-80 to work. Actually, due to the tremendous job Russ Kendall has done in the printer drivers, I don't blame him for keeping the code to himself! And for the same reason, it probably wouldn't be trivial to modify the printer driver. (Russ told me on the telephone that he'd considered a parameter-entry type program to let users plug in their own printer's commands, but it just didn't seem practical, and probably wouldn't work well enough to get the desired results.)

GO TO FAST MODE FOR ALL DISK ACCESS
This is not only for speed, but to eliminate the annoyance of watching the sprites and the split-screen mode flicker as the drive is accessed.

NEED A "MOVE TO" COMMAND
I needed to position a circle a scale 8" from one line, and a scale 10" from another. The only way to do it I could think of was to *SNAP* onto the ends of either line, draw two dummy lines (8" and 10") to be deleted later, then *SNAP* onto the intersection for the center of my circle. (Instead of having to delete all these "dummy" lines, I could have put them on another *LAYER*, then just not printed that layer in my final drawing.

CONCLUSION
It's a powerhouse, and a bargain, and a program I will use.
Home Designer 128, by K&K Software Distributed by: BRIWALL, Box 129, Kutztown, PA 19530, (215) 683-5699 $49.95



LANDMARK TCRB 128

Review by Avonelle Lovhaug

The Commodore 128 world is filled with spreadsheets, databases, and word processors. Graphics packages are also becoming increasingly popular. However, there is more to life than productivity packages and paint programs. And specialty software is also becoming a prominent part of the C-128's existence. An example of this is PAVY Software's "Landmark The Computer Reference Bible" (Landmark TCRB).

Landmark TCRB is a tool which helps you to study the Bible using your computer. Version 2.0 comes on 31 1571 disks consisting of 1 program disk, 24 Bible text disks and 6 concordance disks. The manual and disks are packaged together in a giant (12" x 10" x 2"), spiral bound, plastic casing. The manual comes in large very readable type and the disks follow the manual packaged in clearly labeled plastic pockets, making it easy to return the disks to their correct location after use. Although each disk is not labeled except for the disk number, a card next to each disk pocket clearly states the disk number and the section(s) of the Bible represented on the disk. The program also comes with pockets for 9 extra disks, so the the person who creates new Bible disks can store the disks with the program's other disks, along with extra mini-labels to match the rest of the numbering system. The program supports the 1571, the 1581, and a hard disk drive. It does not yet support the RAM expansion, but there are plans to offer an update which will have RAM expansion support. The program also is currently in 40 columns. Apparently, many of those people who were waiting for the C-128 version of TCRB requested it to be in 40 columns, presumably because it is easier on the eyes. An 80 column option is also planned.

Before I examine the functions and uses of the program, let me give you a little background information which I hope will give you a feel for how the program works. Landmark TCRB begins in a broad main menu area. This menus consists of several options: start the program, color menu, disk menu, printer menus, save selections, and restore selections. The color menu, disk menu, and printer menus all allow you to set your options so that you can fully utilize your particular system configuration. The color menu allows you to set your background color, border color, text color, outline color, and words of Christ color. This flexible option is convenient for those of us who hate being stuck with the programmer's color choices. The disk menu allows the user to read a directory, choose the number of drives the program will utilize, and set the pause or delay time for reading messages and directories. As previously mentioned, the program can utilize a variety of disk drive configurations, including 1 or 2 drives, dual drives and hard drives. The print menus allow the user to set formatting commands such as margins, spacing, characters per line, justification, centering, and printer type. Once you have set up the system for your particular configuration, you can save these settings to disk.

To get a menu in the main program area, you must press the up arrow key (the one next to the restore key). This brings up a three line by six words across menu of choices. The cursor starts on load, which is probably the most used. Loading is fairly straightforward: type in the file name and press return. The file names for the Bible consist of an abbreviated version of the particular book of the Bible, combined with the chapter number. For example, for Genesis 1 the file name is "GE1". To me, this a fairly intelligent way of dividing things up, as opposed to creating files as big as the program can handle, and dividing things up where ever it gets too big. It also makes things easy to find. You are allowed several options when loading a file. You can load with or without references, from devices 8 or 9, drives or 0 or 1 (dual drive), with or without abbreviations, with index markers or with verse designations. The first choice, with or without references, allows you to load the file with the references to other verses included in the reference area, or without references, so that you can create your own references placed into this area. Loading with or without abbreviations allows you the choice of loading the text with or without accents on proper names. Loading with index markers brings files into memory with each verse labeled like this "-001", "-002", etc. Loading with verse designations would change this to: "GE1:1", "GE1:2", etc.

The function menu allows for fast cursor movement with a screen up and screen down option, as well as a goto command. Unfortunately, the goto does not allow you to choose a specific line, but instead allows you to choose from line 100, 200, 300, or 400. However, to choose to move to a specific verse, the user should instead utilize the hunt command, and use the verse number as the search string. This seems a bit clumsy to me, as personally I would assume the goto command would be used for a verse by verse movement. Also, the home option will take you back to the beginning of the file. The hunt option of the function menu allows you to search for a string of characters within the particular file you are viewing. It will also scroll you to a particular verse (as verse numbers are included in the text). It is not case sensitive. After it finds the string, you are asked whether you would like to continue searching for more occurances of the string or quit. The hunt option begins at the cursor, so you can affect the hunt by moving your cursor past an area you may wish to skip.

TCRB also includes a very useful reference feature. This feature provides a listing of verse material which is similar or pertains to the material currently being viewed. Normally, the references reside in a separate area, and although they are a part of the same file, the user views either the main Bible text area, or the reference area, or the buffer area, not any together. However, TCRB includes a special option which allows you to view the references at the bottom of the screen for each verse as the cursor moves. In this way, the user can view both the verse and the reference at the same time. This is feature is similar to the listings found in printed Bibles which contain the references for each verse in the right, left, or middle margins. Another nice feature allows the TCRB user to create his/her own personal notes within the reference area.

PAVY suggests three primary uses for Landmark TCRB: 1) To conduct searches of the Bible for words, phrases and sentences; 2) To create files covering any Bible related subject or section of scripture, probably to be converted for use in a word processor; and 3) To assist in the creation of a personal Bible by outlining text, adding notes and additional references, and saving this back to disk. This personal Bible can be constantly updated and added to, keeping all references and notes organized. Let's examine how well the program facilitates these applications.

Central to all of these applications for TCRB is the programs ability to quickly and effectively search and recover bible text. The search facility is designed to search for various strings, placing the verses found by the search in files to be viewed later. This means that except for disk swaps, the search facility runs pretty much on its own. This function also allows you to search by individual files, old or new testament books, entire testaments, or the entire Bible. The first time I tried this, I used the single drive setting. If you are searching for a fairly common word or phrase, I would strongly advise against using a single drive, even to search just a book. The program searches as many files as it can, placing the verses it finds that match in a buffer. When the buffer runs out, it must save to disk before it can continue. Unfortunately, this means some serious disk swapping. I had asked it to search for three different strings, all fairly common words. Landmark had to save to disk after every three to five files. This took a little bit of time, and a lot of disk swapping. However, with two drives this went much better.

LANDMARK TCRB 128

Continued from Page 6

A nice touch in this section allows you to format a disk before a save if you were out of space. Unfortunately, the program does not appear to check the disk drive status and trap or handle these errors adequately. At one point I encountered a disk error, presumably caused by a corrupted or splat file, and the program did not indicate to me the problem. My drive light flashed, and the program requested that I insert the disks with the file on it and press return. I repeated the process, and was greeted with the same response, but with no message of an error. Instead of allowing me to skip the file, I was forced to abort.

PAVY Software goes into great detail in their manual and other literature regarding the search facility. Apparently there is a perception that Landmark's search facility should be changed so that it can resemble Bible programs found in the MS-DOS world which query the user on each match as opposed to the "batch" method PAVY uses. PAVY argues that the query on each match method means you are tied to your computer until your search is complete, because with this kind of program you must look at each verse found by the search, and then decide if you want to save it, ignore it, or whatever. I found this argument extremely persuasive. I agree that for most people, the method of searching and creating a file for later perusal is superior to the query as you go method. However, if there are a lot of people who are really interested in the alternative, perhaps PAVY should offer both methods. I found the search facility to work very well, and I was pleased with it. However, I would again state it functions much better with multiple drives.

The second primary use of Landmark outlined above is the creation of topical files consisting of any biblical information available. The program allows the user to load any of the files on disk (either Bible files, concordance files, or files created by the user), add notes and additional information. The files can also be converted to the format of many word processors. The file converter is a separate module; although it is part of the program, you must save your current file and then reinsert your program disk to use this function. The converter allows you to choose between saving to an sequential file, a program file, or converting a sequential to a program file. The verses are saved first, with the references at the end. I had no problem loading a bible file converted to a ASCII sequential file into my word processor, Pocket Writer 2.

The third primary use of Landmark TCRB is the ability to create your own personal Bible. The program divides the computer's memory into three main areas: main, reference and buffer. The main area is where you usually load the Bible files. The references are also loaded into the reference area when you do this, unless you choose the option not to load references. The buffer area is where you can create your own files with copies of text and verses found it the main area, references from the reference area, and your own personal notes. You can also place additional references, alternate translations (Landmark features the King James version), or whatever you might find useful. TCRB provides useful, but somewhat crude text editing abilites within the buffer area. While I would advise against writing your religious dissertation within this area, it is certainly adequate for what it is designed to do: copy Bible text and add notes. For bigger jobs, you should export the file to a real word processor.

With a printed Bible, a person studying a section of text might scribble notes in the margins, and underline or highlight text of importance. However, this tends to get messy and disorganized. With Landmark, you can place a color marker before text you want to be "highlighted", and a end color marker when you wish it stop. If you want the words of Christ highlighted in a separate color, the program can accommodate this. Your notes can be typed in exactly where you want them. You can also add to the references.

I did have a few small grievances about the program. For instance, I was a little annoyed at the way the cursor is used in Landmark, or should I say, isn't used. The program uses an arrow across the bottom of the screen to mark the vertical column. To mark the row however, it is simply the row at the bottom of the screen. The up/down cursor keys do not move a cursor, but instead scroll the screen so that you can get the line you want at the bottom of the screen. This means that you can't ever see past what you are looking at for your cursor. It seems awkward to me.

Another problem I had with the program, which may or may not be a good thing is that the function keys are not redefined or disabled with Landmark TCRB. This wouldn't be a problem, except that if you are accustomed to using a function key in another program, and you forget and try to use that key, you could use data. For instance, if you try to type F1, the default BASIC 7.0 definition "graphic" will be printed across your screen, overwriting any text that happened to be there, unless you happened to be lucky enough to be in insert mode. Of course, this could be put to good use by redefining your function keys to something useful before you start the program. In this way, you can set up a macro that might be useful within the program, perhaps a word or phrase used frequently in your notes, or a command used often. Perhaps in a future update PAVY will add a macro definition seuqence which allow the user to redefine the function keys from within the program. This could greatly add speed when moving through menus and increase the programs effectiveness. In the meantime, you could set up your function key definitions using BASIC 7.0's KEY command before starting Landmark.

Standard disk commands are accessible through the disk option. The user can type in a disk command, such as formatting a disk, scratch a file, rename a file, validation of a disk, etc. There is also an option for reading the directory into a file.

The print accessory allows you to print your files in several different ways. It will print the text and references side by side, print the references separate from the text, print only the text, or print only the references. Text you outlined during the main part of the program can be underlined when printed. You can set it to slightly indent the first line of each verse so that each verse's integrity can be maintained. And you can send special codes to your printer so that you can take advantage of special features of your printer, such as fonts, etc.

The manual is fairly complete and is extremely useful for guiding the new owner of the software through the use of Landmark. However, it is typical of most software manuals that it doesn't mention things not related directly to the software. Hence, things like the function keys are ignored in the manual, and only by mistake or by trial and error would you find that the function keys can destroy data in memory. One problem with the documentation is its physical size, due to the huge packaging which holds the manual and the disks, it is difficult to read and refer to without a lot of desk space, unless you removed the pages from their 3 ring binder. Also while the manual includes a table of contents, it lacks an index, which would have been helpful.

Customer support seems to be an important feature of Landmark TCRB. Rick Pavy has indicated to us that he is very willing to hear customer complaints and suggestions. Because of this, and its abilities, the Landmark Computer Reference Bible is an extremely useful program for those persons interested in using their computer to research and study the Bible. The program is available from PAVY Software, P.O. Box 1584, Ballwin, MO 63022 for $164.95.

ULTRATERM III REVIEW

by Ken Harwell

Ultraterm III is a disk based telecommunications program released June 1, 1988 by Steve Boerner for the C-128 in 80-column mode. The entire package consists of a disk and a user's manual (hole-punched but unbound). The program is NOT copy-protected so you don't have to play with dongles, codes, passwords, or 3D glasses in order to boot it up or back it up. In fact, when you order the package, you are asked to choose whether you would prefer 3 1/2 or 5 1/4 format. This is a nice touch for 1581 devotees.

The documentation is through (51 single-sided pages), well-organized, and easy to understand. This in itself is a plus for any type of program but especially one for telecommunications. In addition to fully supporting Ultraterm III, the appendix of the documentation has several pages of valuable telecommunication background information. For instance, as a PunterNet sysop, I can tell you that I have never read a more concise and easy to understand description of how Punter C1 protocol works.

Ultraterm III utilizes and/or supports: baud rates of 300 450 600 1200 or 2400, a 61440 byte buffer, a buffer editor, real-time and on-line clocks, a built in two-drive copy/move utility, direct and indirect DOS access in addition to 1581 subdirectory manipulation, DOS support for up to 4 external disk drives in addition to a RAM disk if you have the 1700 or 1750 REU connected, a full color screen display, local echo toggle, a programmable multidialer phone index, programmable printer controls, block punter xmodem xmodem/crc windowed xmodem and xmodem-1k (sometimes referred to as ymodem) transfer protocols, an xmodem stripper option, and a text file reader. This list is not complete but includes most of the features of Ultraterm III.

Ultraterm III is user-friendly and becomes even more friendly as the user becomes acquainted with the program operation. This is accomplished by "quick key" commands. With only 3 exceptions (save screen, print screen, and conference mode), every command in Ultraterm may be executed by 2 different methods. The first way entails using pull down menus (i.e. Help, Modem, Index, Buffer, Transfer, Disk, Reader, Other) in order to cursor to your command choice. Once the user is familiar with the commands he/she may opt for the faster and more flexible "quick key" method. This second way entails selecting the command choice by pressing the Commodore logo key and a second case-sensitive key (you may have to press the SHIFT key as well) at the same time. For example: A user wishes to display the key commands help screen. This may be accomplished by 1) hitting RUN/STOP to display the command line, cursor over to the Help bank, hit RETURN to display the window, cursor down to key commands and hit RETURN to make the selection; or, 2) hit C= h. For those who dislike a screen of windows overwriting windows and/or excessive menu selection as found on other terminal programs, the quick key method is an efficient alternative.

As a further convenience, included on the disk is a program called CONVERT which allows the user to update the phone index from previous versions of Ultraterm to Ultraterm III.

Another plus of this package is the "play before you pay" attitude of the author. All prior versions of Ultraterm have been released as shareware. This is not the case with Ultraterm III since the author has decided to serve as the sole distributor. However, there is a file UTERM3/DEMO.SDA which may be found on GEnie, CIS, Qlink, and perhaps on your favorite local bulletin board system. It comes in every package and you are free to distribute the demo wherever you choose. This self-dissolving arced file is an exact duplicate of Ultraterm III except that the transfer protocols have been removed. Mr. Boerner has provided this demo to the C-128 community with no strings attached (in other words, if you don't need to use transfer protocols in your personal telecommunications then this program is FREE).

In my opinion, there can be no honest review of a telecommunications product for the C-128 without a comparison to BobsTerm Pro. Ultraterm III does not have all the bells, whistles, and features of BobsTerm Pro but then again it does not have BobsTerm Pro's high price tag (refer to Twin Cities 128 Issue 7). The "quick key" option and ease of movement thru the program results in my personally preferring Ultraterm III over BobsTerm Pro but I am sure that not everyone would agree on that point. It really comes down to a question of personal tastes and needs.

My complaints with Ultraterm III are minimal. The status line is a bit "cutesy" and while someone may gain benefit from the light show effect of the flickering FLG, RXD, and TXD rates, I found it to be an annoyance. I could find no way to define the function keys other than from within the phone index file. For the average user this probably will not present a problem but for a remote sysop with heavy maintenance responsibility, this is definitely a drawback. A wish list for the next update would include support of CP/M format during file manipulation and support of macros (a convenience when utilizing PC Pursuit and other such services).

The price of Ultraterm III is $25. If you are already a registered owner of Ultraterm then the cost to upgrade is $10. If you have a copy of the demo then the author asks that you use the order form enclosed therein. If not, you may obtain Ultraterm III by contacting: Steve Boerner, P.O. Box 364, Brockport, NY 14420 and remember to designate what type disk you wish to receive. In conclusion, I found Ultraterm III to be an outstanding addition to my C-128 program library and I have no hesitation in recommending it to anyone on or off a budget who is desirous of a superior telecommunications program for their C-128.

The Dao and Zen of Metal Art and Computer Music
Original Video Art Created With the C-128 by Raymond C. Bryan

I set limits to the piece in these bounds: use only the 2 resident languages and the operating system, one disk drive and no other inputs, no external devices except for output of video and audio, make the piece self-starting on power-up, present it in a gallery as an installation (among my metal pieces where possible), present audio, video and textual images, superimpose movable object blocks over the video and switch and switch between the video drawings (and show the program itself as well as computer memory banks) for animation. The video phosphographs are presented to one monitor and a TV (one of these monochrome). These were prepared in advance and are stored until needed on disk. Another monitor receives the text one line at a time. All three are sent the sonic portion. A print of the program text is presented as an integral part of the work. The structure of the piece is governed by the music's structure. The words and images with movable object blocks randomly directed over them are cycled through until the music has cycled it self twice through. Then the music presents a coda and the monitor screens present accreditation and copyright text.

The work is to be presented in August 1988 at Theatre in the Round in the Twin Cities. For further information, contact Nancy Conroy, curator (612) 870-0168 or R.C. Bryan (612) 375-7656. (Look for coverage in issue #22 of Twin Cities 128)

C-128 DEVELOPER'S PACK

Review by Miklos Garamszeghy

When I first unwrapped my DEVPAK (thanks Fred!!), I thought 'Oh great! They've sent me the wrong one.' The huge bright letters on the front cover spell out 'Commodore Amiga'. However, on second glance, I saw the much smaller 'C-128 Developer's Package' title printed in black and white in a window at the top of the cover.

The DEVPAK consists of a cerlox bound (like plastic fingers) 8 1/2 x 11 by about 3/4 inch thick manual and two 5-1/4 inch flippy disks. The 'manual' is more of a collection of various pieces of printed documentation. It covers the basic instructions for the main programs on the disk: an assembler, an editor and an object file loader; as well as long listings of some of the source code files contained on the disk. The manual also contains a brief summary of the differences amongst the various ROM versions for the C-128, 1541, 1571 and, oddly, the SX-64 (remember that portable C-64?) along with a good discussion of the C-128's BASIC 7.0 ROM routines for handling various math and number type conversion routines (from relatively simple routines such as integer and floating point arithmetic to the more complex exponentiation and trig functions). All of these routines can be called by user machine language programs (why re-invent the wheel?).

In addition to those described above, there are many more programs on the disks, such as a sector editor (the same one which is on the 1581 demo disk), for which there is no documentation supplied at all. I think it would have been much more helpful to omit the fully two thirds of the manual which contain the source code listings (they are all on the disks anyway, and a good chunk of them deal with fast load routines for a C-64 at that. Who needs these for a 128?) and replace it by some docs for the other useful programs on the disk. I would also like to have seen an updated version of the 1581 'Software Specification' guide which was released to some developers in the early stages of the development of the 1581 and contained some very detailed technical info (including a memory map of the RAM areas, which by the way is next to useless because most of the important ones changed by the time the final version of the drive came out) along with a similar guide for the 1571. (Since we are on making a wish list, a discussion of the MFM capabilities of the new CR 1571 drives (which lack a true WD 177x disk controller), such as those in the North American C-128D's and the new 1571 drives, would have been nice.)

The main disk contains the editor, assembler, loader, etc. When you boot this disk, you get a nice menu from which to select the program that you want. The menu then loads the selected program from disk and runs it. Note that some of the programs, notably the object file loader, do not make a very graceful exit when they are finished. (I even had it crash on me a couple of times by specifying that I wanted to load the object file into $1c00 of bank 0, the normal start of BASIC area, which is a perfectly legit place to stuff code. The manual warns about putting code into other areas used by the loader, but not here. This can be fatal because the idea of the loader is to get the assembler produced object files into memory so that you can save them to disk as binary (i.e. executable machine code) files.)

The editor, ED128, is apparently designed to be similar to the DEC mainframe editor EDT, in terms of look and feel. Great if you spend most of your time programming mainframes, but not so hot if you are used to other Commodore style editors. Once you get used to it, the editor is very flexible and powerful allowing full screen cursor movement along with all of the usual text formatting, search, copy, cut & paste, etc. features found on most good editors. An on-line command summary is available at the touch of the 'HELP' key. This is a nice feature, especially to new users of the editor who may be more than slightly confused by some of the commands.

A couple of general comments about the editor. It seems that you are always in "insert mode". That is, if you enter new text into the middle of a line, the rest of the line shifts over and the new text is inserted into the line rather than over-printing the existing text. There does not appear to be any way to turn off this 'feature' which I find distracting when working with an editor that is basically designed to handle assembler source code. Another feature I did not like about the editor was the exclusive use of the numeric keypad for entering editing commands. It does not function as a normal keypad so it cannot be used for entering long lists of numeric data in assembler .BYT statements.

The status lines at the bottom of the screen display (which CAN be turned off if you don't like them), display system messages, line and column info, etc. An interesting point about it is that it also displays the number of bytes remaining in the text buffer. This is especially useful when working with long files which may have a danger to become too large. For smaller files, however, a method to display bytes used instead of bytes left would be nicer.

The editor is quite flexible in that you can enter virtually any character code, including non-printing ones, by means of a special command followed by a decimal character code (such as 10 to insert a linefeed command into a string). This feature allows you to embed control codes into text strings used for messages in your assembler source code file. The codes are entered as decimal numbers, but in your text they are displayed as hex numbers. This can be confusing, especially in assembler source code files.

You can also toggle the display between PETSCII and ASCII character sets by the usual < SHIFT > C = key toggle. Thus your text messages can contain virtually any mix of PETSCII and ASCII characters.

The printer routine is quite simple: what you see on the screen is what you get. There are no special drivers for different printers. If you want special formatting, you must manually insert the appropriate control codes into the text file. Note, however, that if you do this in a source code file, these codes may confuse the assembler.

In all, if you can live with its quirks (or otherwise get used to them, remember the old saying: You can't teach an old dog new tricks), then ED128 is quite serviceable for tasks such as creating assembler source code, but I wouldn't want to use it to write my version of 'War & Peace'.

Next up is the assembler: HCD65 macro assembler. This is quite a powerful beast, although it can be a trifle slow because it is a disk based assembler, meaning that it reads all of its source files from disk then creates an object file, again on disk. This can create a few problems if you are using linked files, especially when they are all on the same drive, because of the number of open file limitations on most drives. Attempting to do even moderately complex file linking creates havoc. Fortunately, the manual recognizes this fact and states it as a limitation. The docs for the assembler assume that the user is familiar with assembly language programming. It only contains a listing of the opcodes supported by the assembler, but no indication as to how one might use them.

The assembler comes up through a BASIC shell which has a number of menu screens. This gives you great flexibility in setting up the parameters for the assembly by allowing you to easily change the BASIC portion of the assembler. Fortunately, once the parameters have been set, the actual assembly is done with a machine language routine for speed.

Continued from Page 9

After entering the name of the source file (the assembler assumes that all source files will end in '.src', so if yours do not, you will have to rename them), the first menu gives you a selection of various pre-programmed assembly options, such as what you wish to do with the object file, listing file, error listing etc. If none of the pre-programmed combinations suits your fancy (for example, there is no 'pre-view' type option where all you get is a screen listing of the assembled file without creating an object file), just press <return> to go through more detailed menus step by step. (The main menu does not list this option, but it is buried in the documentation.) Once everything is set, the assembly starts. When done, you will be returned to the BASIC prompt. If you want to re-run the assembler, you must be sure to have the assembler disk in the last accessed disk drive because the machine language portion of the assembler is re-loaded each time the program is run. It would be nice to go back to the opening menu, but that is easily solved by adding the appropriate line to the BASIC portion of the assembler.

The macro capabilities and conditional assembly features are quite powerful indeed, with many levels of nesting possible. The limitations of using macros and some common pitfalls are nicely outlined in the docs. Unfortunately, you cannot include already-assembled machine language in the assembly of a source code file (this would be nice if you are working with mixed Z80/6502 code, which is quite possible, and even desirable with the C-128 in some cases).

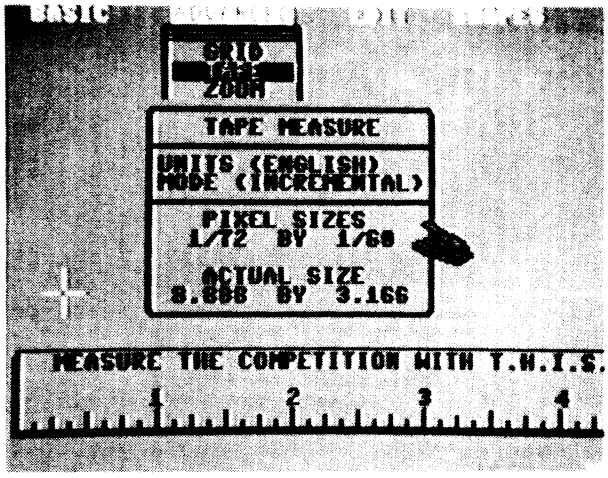
My one basic complaint about the assembler is that it does not produce executable machine language output. It produces an 'object' file which contains a series of data records in 'MOS' format containing a header of byte count and address information along with the actual code. This is fine, except that to do anything with it, you must use the object file loader, also contained on the disk, to read it into memory, then BSAVE the resulting image to a disk file. Although this type of assembler output is fairly common in some other operating systems, such as CP/M or MS-DOS, it is not too common in 8 bit Commodore machines. All of the assemblers that I have worked with on VIC, C-64 and C-128 machines have produced executable output directly without having to go through an intermediate "object" file. This is a much less tedious way of doing things. (You call it an OBject file, I call it an obJECT file.) With most other assemblers, you can also assemble source code directly from memory and/or to memory which is a quick and dirty way to test your code (assuming of course that you do not over-write the area occupied by the assembler).

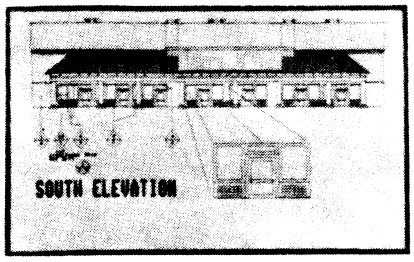
C-128 DEVELOPER'S PACK

The remainder of the disks contain various utilities, such as file copy (Jim Butterfield's ubiquitous UNICOPY) and disk back up programs, a file cruncher, sector editor, auto boot maker, RAM disk program, C-64 sprite and SID editors, etc. along with many source code files for 1351 mouse routines, burst utilities, RAM expansion tools and more. The source code is very well documented and should be easily modifiable by any reasonably experienced programmer to suit specific purposes. However, the disclaimer at the front of the DEVPAK manual states that these routines cannot be used in commercial software without the express written consent of Commodore. What is the point of giving them out on the disk if they are adopting a 'look but don't touch' attitude? To their credit, however, Commodore does say that they can be used freely in non-commercial programs.

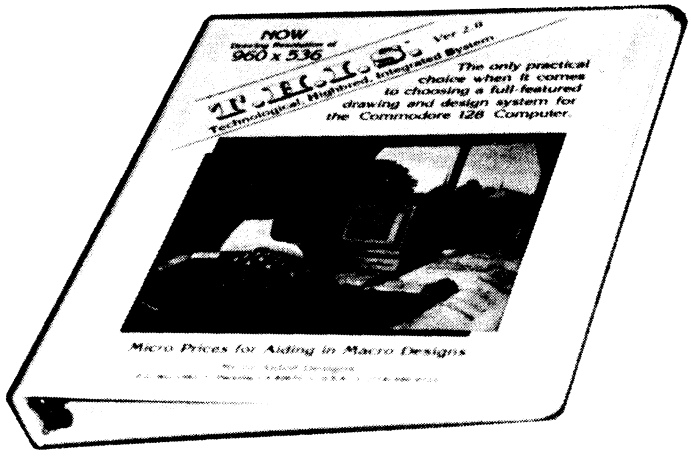
In summary, I think the DEVPAK is an excellent idea. Too bad the execution was a bit disappointing (and long overdue) for the advanced programmer. However, it can still be a very useful source of ideas (lets not waste all that good source code) and can be quite beneficial to someone who is looking to get a start in serious machine language programming. *C-128 Developers Package, available only from Commodore, 1200 Wilson Drive, West Chester PA 19380 ATTN: CATS orders: C-128 Devpak, $49.95*

Micro Aided Designs
P.O. Box 1982
Placentia, CA 92670
(714) 996-0723





T.H.I.S.



THIS *(Technological Highbred Integrated System) is a program designed exclusively for the Commodore 128. Listed as one of the best programs of 1987 by "Commodore Magazine," THIS contains many advanced features not found in any other program. THIS proves to be the most user-friendly graphics program, complete with a step-by-step easy to understand manual. Using a 1700 or 1750 RAM and a light pen or mouse, THIS is a drawing system capable of producing a simple doodle or a sophisticated blue print.*

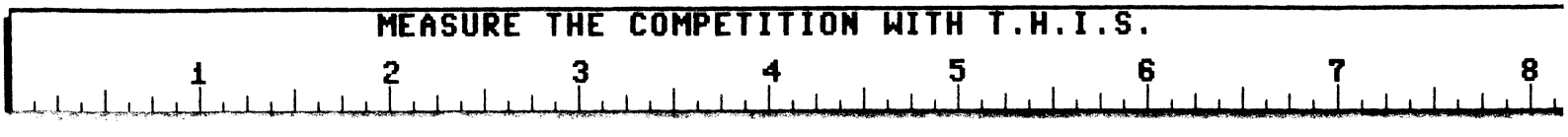
With THIS you have a possible resolution of 960 by 536 dots per drawing page and seven (7) totally separate pages to draw upon. Giving the user a larger area to draw on not only allows for larger sized drawings but also increases the amount of details that can be added.

When it comes to the final step of printing a complete drawing, THIS offers exact-scaling. With exact-scaling the completed drawing can be printed to scale within 1/64 of an inch. THIS allows for the possibilities of templates to be created and used in real life applications. The printout sizes can vary anywhere between 2 by 2 inches to 480 by 321 inches.

EXCLUSIVE FEATURES

- *Up to 7 pages of graphic designs*
- *Maximum virtual screen resolution of 960 by 536 pixels per page*
- *Exactly scales printouts as large as 480 by 321 inches*
- *Real time object rotation and move*
- *Real time cut, copy, paste, and zoom*
- *Elastic modes include lines, boxes, circles, ellipses, arcs, and other geometric shapes*
- *Easily accessed graphics library including over 100 detailed electronic symbols*
- *Over 700 type sizes*
- *10 font styles*

Retail Price $63.99



1581 CRC MYSTERY & FIX

by Miklos Garamszeghy

In the past few months I have been contacted a number of times by 1581 owners who have managed to corrupt the directory track on a disk full of irreplaceable (read NOT BACKED UP) data. The first time, I thought it was a fluke. After all the person in question was a very experienced programmer who would not deliberately trash his disks. Then it happened again, with someone else in a different place (another country actually) under a different set of circumstances but again by an experienced programmer. Not having a trashed disk to work on myself, I could offer no more than a few vague suggestions and my deepest sympathy. With the latest occurrence, I decided to try to get to the root of the matter. The result is the accompanying program that 'repairs' a trashed disk track.

In all cases the 1581 owners were getting an error message indicating a read error on Track 40, Sector 0 which is the 1581 BAM header sector. If you can't read this sector, you can't access anything on the drive by normal methods, even using most sector editors. However, here's the crunch: the physical disk error which causes this crash can be anywhere on that track, not just sector 0. The reason is the track cache buffer. When you do any kind of normal disk access, the 1581 DOS reads everything from a complete physical track into the buffer. While this improves disk speed, it can also foul things if an error occurs anywhere on the track.

One would think that the DOS error message would report the actual location of the error but it doesn't. It reports the track and sector that DOS was trying to access as the location of the error. So, even if there was an error on track 40, sector 10 when you tried to log in the disk (i.e. read track 40, sector 0), you get the error message. Fortunately, in most cases DOS has already read the sector into memory before it decides that an error is present. The error is detected by comparing the calculated CRC checksum of the sector with that stored on the disk. If the two don't match for any reason (even if it is only a single bad bit), an error is reported, but the sector is in the drive's memory. Unfortunately, DOS will not let you proceed if it detects the error and an error anywhere on track 40 is fatal because it will not allow the disk to be logged in.

I am not sure why all of the cases that I have encountered to date have had the problem on track 40 (the BAM track) and I still don't know its cause, but at least I am able to cope with it after the fact. Perhaps this is because it is the most used track and is being subjected to lots of wear and tear by being constantly read and written during all kinds of disk accesses. Perhaps it is a bug in the 1581 or perhaps it is just due to random disk flaws (not likely).

Fortunately, all is not lost. The trick to recovering data from the trashed track is to read the sectors on the bad track directly, without using the track cache. This is done by direct programming of the DOS job queue. The procedure is to do a physical sector read, then write for each sector on the bad track. When the sector is re-written, the new CRC checksum will be written to the disk and all will be hunk-dory again (almost). The only problem that may arise is that some data MAY be corrupted. (There are two reasons for a bad CRC error: one is that the CRC bytes are bad and the other is that the data bytes are bad. In all of the cases I have come across, the problem seems to be with the CRC because data was successfully recovered in all cases.)

1581 Fix Track is a very simple BASIC program which will work on any 8 bit Commodore computer with a 1581 drive (C-64, C-128, Plus/4, etc.). After specifying the device number for the 1581, the program asks you for the logical track and sector which contains the error. This is the track and sector displayed in the DOS error message (xx READ ERROR, TT, SS). The logical track is then converted to a physical track (PhysicalTrack = LogicalTrack - 1). The physical disk side is computed from the logical sector number (logical sectors 0 to 19 are on side 0 and 20 to 39 are on side 1). The program then loops through a series of physical sector reads/writes which by-pass the track cache and read/write directly to/from the job queue data buffers.

Of note to disk drive programmers is line 1240 which contains a simple method to tell a 1581 from a 1571. The 'U0>B' command set is not implemented on the 1571 (it is equivalent to the 1571's 'U0>M' command). If you get a syntax error after trying the command, the drive is not a 1581. If no syntax error occurs, it is a 1581. (Of note to non-programmers is that none of this technical stuff matters as long as your disk is recovered.)

While I cannot guarantee that Fix Track will cure all evils, it has recovered data for several trashed disks. One final note: after doing the repair, immediately copy all of the files on the disk to a new disk just to be sure and be wary of any program files which were on the bad track (it is not quite so bad with the directory track or text files or even BASIC source code because data errors in these files are relatively easy to spot and correct).

```
1000 rem ***********************************
1010 rem *      1581 fix track v1.1        *
1020 rem * <c> 1988 herne data systems ltd *
1030 rem *      by  m. garamszeghy         *
1040 rem ***********************************
1100 bs$=chr$(157)+chr$(157)+chr$(157)
1110 goto 1160
1120 print chr$(147);"    **** 1581 fix track v1.1 ****"
1130 print "   <c> 1988 herne data systems ltd."
1140 print : print : return
1160 gosub 1120
1170 print"this program will repair a 1581 disk"
1180 print"with a crc error damaged track." : print : print
1190 print "select target device #    8";bs$;
1200 input td
1210 if td<8 or td>12 then end      : rem exit bad drive number
1230 close 15 : open 15,td,15       : rem open command channel
1240 print#15,"u0>b0"               : rem test drive type
1250 input#15,a$
1260 print#15,"uj"                  : rem reset drive
1270 if asc(a$)<>48 then goto 1680 : rem not a 1581
1290 print : print "insert damaged disk in device";td
1300 print"then press a key to continue ..."
1310 get w$ : if w$="" then 1310    : rem wait loop
1330 print#15,"u0"+chr$(10)         : rem log in disk
1350 gosub 1120
1360 print : print "enter logical track,sector to repair"
1361 input "==> ";lt,ls
1370 t=lt-1      : rem convert logical to physical track
1380 if t<0 or t>79 then print"error invalid track": goto 1360
1390 si=0 : if ls>19 then si=1      : rem select disk side
1410 print : print "repairing physical track ==>"t
1420 print : print "                    sector ==>";
1440 rem the following code does a physical sector read
1450 rem then write for each sector on the track
1460 rem by-passing the track cache buffer
1480 for s=1 to 10           : rem do entire physical track
1490 print s;
1500 print#15,"m-w"chr$(206)chr$(1)chr$(1)chr$(si):rem side
1510 print#15,"m-w"chr$(11)chr$(0)chr$(2)chr$(t)chr$(s)
1530 print#15,"m-w"chr$(2)chr$(0)chr$(1)chr$(164): rem read
1540 gosub 1630     : rem wait till done
1560 print#15,"m-w"chr$(2)chr$(0)chr$(1)chr$(166): rem write
1570 gosub 1630           : rem wait
1580 next s               : rem repeat
1600 print :  print : print "**done disk is repaired **"
1610 print#15,"i0" : close 15 : end : rem reset drive and quit
1630 print#15,"m-r"chr$(2)chr$(0)chr$(1) : rem job status
1640 get#15,a$:if asc(a$+" ")>127 then 1630
1650 return
1670 rem exit if not a 1581 drive
1680 print:print"error device";td;"not 1581 drive":goto 1610
```

MORE MEGA-HIRES GRAPHICS

by Fred Bowen

In the last issue of Twin Cities 128 (TC-128 #20), I discussed one method of generating large (640x400) bit mapped displays on C-128's with 64K bytes of display RAM. While that particular method involved an interlaced display, it also involved a rather nasty bit of programming because the system processor was used to alternate between the odd and even fields of the interlaced display. Of course it had advantages, no display memory was "wasted" and, if you were astute, you would have seen how that technique could be used to generate more colors instead of more resolution. 128 colors, in fact, by alternating between attribute areas instead of bitmaps. Even more tantalizing would be alternating both bitmap and color information between odd and even frames. Food for thought.

In this issue, I want to discuss more conventional interlaced bitmap displays on the C-128. As in the previous article, I will precede some examples with an elementary discussion of some of the principles involved. The examples will utilize BASIC 8 like before too, but that is for the convenience of rendering displays; it is not required for, nor does it directly support, interlaced and overscan displays. The chief advantage of the "true" interlaced displays I will discuss in this article is that you can do other things while in this mode the VDC does all the work and the processor is unencumbered.

The type of CRT display used for the C-128 and most modern television sets, monitors and terminals is called a raster scan display. In this type of display, the CRT's electron beam is driven in a zigzag pattern from the top to the bottom of the display, not unlike the print head of your typical printer. If the intensity of the beam is sufficient, it "excites" the fluorescent coating on the screen and tiny dots begin to glow for a short time. The beam "draws" in only one direction, then, like an old printer, the beam must be "returned" to the start of the scan line. In display terminology, this is called horizontal retrace. When the beam reaches the bottom of the display it does a "form-feed", moving back to the topmost scan line. This is called vertical retrace. How far the beam can move from left to right and top to bottom before retrace is called, respectively, the Horizontal and Vertical Total. This is more-or-less equivalent to the printer's page size. The number of pixels and scan lines that the beam can effectively "draw" on is called, respectively, the Horizontal and Vertical Display Total. The display totals partially define the display's margins, just like the number of rows and columns of text a printer can fit on a page.

All these parameters, and more, are programmable on the C-128 simply by stuffing suitable numbers into the appropriate registers of the VDC (Video Display Controller). There are, of course, inherent limits to these numbers and the maximum possible size of the display. We are limited by available display RAM in the C-128, the capabilities of the display monitor, and a some physical standards. One thing we cannot program is the speed at which the CRT's beam is moved. In the U.S., the beam is typically swept horizontally at a rate of 15.75 kHz and vertically at a rate of 60 Hz. This sets the maximum possible vertical resolution at about 240 scan lines (15,750 / 60 minus some time for retrace, typically about 22 scan lines). A few more scan lines are used for a border, making the displayed total come to 200 scan lines (25 character lines). The horizontal resolution is a function of the scan frequency (15.75 kHz) and the pixel frequency (sometimes called video frequency; how fast the monitor can modulate (change) the beam intensity to draw individual dots), which is determined by the monitor hardware.

For the home enthusiast (that's you, right?) interlace mode is really the only option for increasing the size of the display, and last month I quickly described what interlace mode is- two fields of display per frame, one field displayed on "odd" scan lines and the other on "even" scan lines, resulting in a refresh rate of 30 Hz, half the normal rate. Half the display rate means we effectively double the number scan lines that are displayed. You might be interested to know that interlace is used in broadcast television (television does not suffer from "flicker" because adjacent scan lines are similar). Before I go on, however, a word of caution is in order. Tinkering with the horizontal scan rate can be harmful to some monitors, particularly cheap, often monochrome ones. Here's what happens- the beam is typically driven by powerful a magnetic yoke and a "flyback" transformer. The horizontal retrace is performed by allowing these magnetic fields to collapse, which causes a high voltage to appear at the output transistor. If the transistor is turned on at this point by you fiddling with the VDC the transistor most likely will turn into a puddle of goo. Normally, this would not be a problem, as any well designed monitor will not let this happen. But such protection may not be found on cheap monitors. Commodore monitors are protected, but consider yourself warned!

The table below contains the magic numbers which, when used in conjunction with the normal bitmap mode setups, will provide the indicated resolution. All the values are in hexadecimal, expressed in terms of characters, for NTSC monochrome displays.

	HorzTot	HorzDisp	HorzSync	VertTot	VertDisp	VertSync
	R0	R1	R2	R4	R6	R7
640 x 480	7D	50	66	4C	4C	47
640 x 600	7D	50	66	5C	5C	57
720 x 480	7D	5A	68	4C	4C	47
752 x 600	7D	5E	6A	5C	5C	57

The horizontal numbers are easy to understand: you can easily see that $50 = 80 characters times 8 bits per character yields a 640 pixel wide screen. Similarly $5A = 90, 90*8 = 720 pixels and $5E = 94, 94*8 = 752 pixels. HorzSync is used to center the display horizontally. The vertical numbers are easy too- a normal, non-interlaced display has a vertical total of $20 for a 200 scan line display like I discussed above. The vertical total for interlaced screens can be roughly calculated by simple ratios. Solving for X,

$$\frac{32}{200} = \frac{X}{480} \text{ gives } 76 (\$4C) \qquad \frac{32}{200} = \frac{X}{600} \text{ gives } 96 (\$60)$$

The VertTot for a 480 scan line interlaced display is, in fact, $4C. We have to fiddle a bit for the 600 scan line display, and I find a VertTot of $5C works best (actually, this is right on if we call it a 580 scan line display).

Unlike the "bit banging" interlace method I described in my last article, this "true" interlace method requires you to render into each odd and even field separately due to fact BASIC 8 does not directly support this mode. Also, I mentioned earlier that this scheme "wastes" some display RAM. These two things are related: during the vertical blanking period which occurs between the two fields the VDC's internal address display fetch register continues to increment so that the odd and even display fields are not contiguous. The "gap" between them varies in size based upon the Vertical Total and Character Size parameters.

We can calculate where the second field begins, using the formula below:

```
    R6                R9                  R1
( (VertDisp / 2) * (CharTot + 1) +1 ) * HorzDisp
```

For the screens described above, we can plug in the values of R6 and R1 and come up with the following table, assuming a CharTot of 6:

MORE MEGA-HIRES GRAPHICS

Continued from page 13

	Even field Address	Odd field Address
640 x 480	0	21360
640 x 600	0	25840
720 x 480	0	24030
752 x 600	0	30362

The smaller interlaced displays (640x480 and 720x480) should be acceptable to most 80-column monitors. The larger displays, particularly those with serious overscan (640x600 and 752x600), will probably work only if you have a monitor equipped with an external vertical hold adjustment. And if you are fortunate enough to have a monitor with external size & position adjustments, you are in excellent shape to view these screens. I have several versions of Commodore monitors, including 1902, 1902A, and 1084 models, and on only the 1902 did I experience a vertical hold problem, and fortunately that's the only one of these monitors with a vertical hold adjustment. If your monitor is not happy with these displays, and you happen to have a monochrome monitor available, it has been my experience that most monochrome monitors tend to fair better at interlaced displays. Of course, if you have access to a multiscan monitor, you will have no problem. Whichever monitor you have, flicker can be minimized by selecting screen colors and adjusting the monitor's contrast and brightness.

All these interlace displays also involve varying degrees of overscan, which is in effect displaying data in the border area. Not all monitors can display much overscan, in which case you should consider limiting your drawing to a 640x400 region. Using BASIC 8, the "normal" viewing area for each of the displays is:

	Ymin	Ymax
480 scan lines	40	439
600 scan lines	50	499

Here is a BASIC 8 program which performs interlace and overscan magic. I know of only one such software package which fully supports interlaced graphics on the C-128, Graphic Booster 128 by Combo AG., and I believe it is only available in Europe. I imagine it only supports PAL displays, but it would be fairly simple to adapt that to NTSC. Like BASIC 8, Graphic Booster 128 contains 80-column mode graphic extensions to the C-128's built-in BASIC language.

(Editor's Note: Although this is a BASIC 8 program, this program should be run from the 40 column display. Fred uses the 40 column display for text and the 80 column display to show off his tricks. Using a dual monitor like the 1902 or if you have two monitors hooked up to your 128 is ideal.)

```
10 REM C-128 VDC 64K RAM INTERLACE DEMO
20 REM
30 REM FRED BOWEN              JUNE 1988
45 :SLOW:LIST10-30
50 PRINT" 1. 640 X 480
51 PRINT" 2. 640 X 600
52 PRINT" 3. 720 X 480
53 PRINT" 4. 752 X 600
55 :
60 PRINT:INPUT" SELECT DISPLAY FORMAT";M$
65 M=VAL(M$): IF M<1 OR M>4 THEN 60
70 RESTORE M*1000
80 READ X,Y,S,HT,HD,HS,VT,VD,VS
90 SYSDEC("E179")        :REM INIT VDC
95 FAST
100 @WALRUS,1
110 @SCRDEF,0,0,0,720,720,0,0
120 @SCRDEF,1,0,0,X,Y,0,0
130 @SCRDEF,2,0,0,X,Y,S,0
150 @COLOR,1,15,0:@SCREEN,0,0:@CLEAR,0
160 A$="tHIS IS "+STR$(X)+"X"+STR$(Y)+" !"
190 @SCREEN,2,2
200 :@CHAR,254,10,180,1,1,2,A$
210 :@LINE,1,0,0,X/2,Y/2/2-1,0,1
220 :@BOX ,0,0,0,X/2,Y/2/2-1,0,0,0,1
230 :@CIRCLE,460,100,0,75,1
250 @SCREEN,1,1
260 :@CHAR,254,10,180,1,1,2,A$
270 :@CHAR,254,10,160,1,1,2,A$
280 :@LINE,0,  0,0,X-1,Y/2-1,0,1
290 :@BOX ,0,  0,0,X-1,Y/2-1,0,0,0,1
300 :@CIRCLE,460,100,0,75,1
310 :@CIRCLE,540,100,0,75,1
330 SYSDEC("CDCC"),HT,0    :REM HORZ TOTAL
340 SYSDEC("CDCC"),HD,1    :REM HORZ DISPLAYED
350 SYSDEC("CDCC"),HS,2    :REM HORZ SYNC POSN
360 SYSDEC("CDCC"),VT,4    :REM VERT TOTAL
370 SYSDEC("CDCC"),06,5    :REM VERT TOTAL ADJ
380 SYSDEC("CDCC"),VD,6    :REM VERT DISPLAYED
390 SYSDEC("CDCC"),VS,7    :REM VERT SYNC POSN
400 SYSDEC("CDCC"),03,8    :REM INTERLACE MODE
410 SYSDEC("CDCC"),06,9    :REM CHAR TOTAL VERT
420 SYSDEC("CDCC"),00,27   :REM ADR INC PER ROW
500 R=65:P=.01: FORI=0 TO 2*<pi>+P STEP P
510 : XX=75+R*COS(I): YY=160-R*SIN(I)
520 : IF YY AND 1 THEN @SCREEN,2
530 : YY=YY/2: @DOT,XX,YY,0: @SCREEN,1
550 : SYSDEC("CDCC"),3,8:SYSDEC("CDCC"),0,27:NEXT
600 FOR YY=0 TO 7 STEP 2
610 : @COPY,1,79,160+YY,150,1 ,1,78,140+YY/2
620 : @COPY,1,79,161+YY,150,1 ,2,78,140+YY/2
630 : NEXT
700 SLOW:RUN
999 :END:  X   Y    S    HT HD  HS VT VD VS
1000 DATA 640,480,21360,125,80,102,76,76,71
2000 DATA 640,600,25840,125,80,102,92,92,87
3000 DATA 720,480,24030,125,90,104,76,76,71
4000 DATA 752,600,30362,125,94,106,92,92,87
```

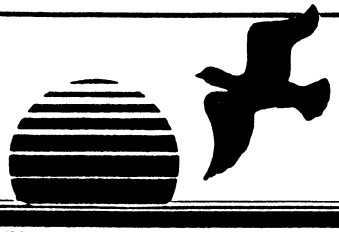
This program puts a box around the screen, draws a diagonal line though it, and places some text and circles too. The graphics that flicker are those that are rendered into one field only; those that do not flicker have been rendered into both odd and even fields of the interlaced display.

The routine at 500 draws a circle properly for the interlaced display- putting odd lines into the odd field and even lines into the even field.

The routine at 600 copies the text string to both odd and even fields for proper display on the interlaced screen. A similar routine can be used to convert an entire BASIC 8 screen to an interlaced display. Consider the following example which converts the "BASIC 8" Title screen to a 640 x 480 format.

```
100 @WALRUS,1:@COLOR,0,15,0
110 @SCRDEF,4,0,0,640,400,16000,0:@SCREEN,4:@CLEAR,0
120 @SCRDEF,0,0,0,640,200,0,0
130 @SCRDEF,1,0,0,640,200,21360,0
140 @SCREEN,0:@BOX,0,0,0,639,199,0,0,0,1
150 FOR Y= 1 TO 199
160 @COPY,0,0,Y,639,1,Y AND 1,0,(Y/2)
170 NEXT
180 @SCRDEF,2,0,0,640,100,8000,0
190 @SCRDEF,3,0,0,640,100,29360,0
200 @SCREEN,2:@CLEAR,0
210 @SCREEN,3:@CLEAR,0
220 @SCREEN,0
230 FOR I= 1 TO 6
240 READ R,D$
250 BANK 15:SYS DEC("CDCC"),DEC(D$),R
260 NEXT
270 DATA 4,4C,5,6,6,4C,7,47,8,3,9,6
280 GETKEY A$:@TEXT
```

Free Spirit Software Inc.



UTILITIES

THE SUPER CHIPS
Custom Operating System for the C128

Three 16K ROM chips that add several powerful features to Basic 7.0 including FIND, CHANGE...THIS...TO...THAT, TYPE, UNNEW, COMBINE, MERGE, START, FILE, EDITOR and more! Simultaneous split screen directories of devices 8 & 9. Compatible with 1541/1571/1581 and virtually all software and peripherals. Only **$49.95!**

Super Chips, Custom Operating System for the C128D - Two 32K ROM chips - Only **$49.95!**

Super Chip, Custom Operating System for the C64 - One 16K ROM chip - Only **$29.95!**

Super Chip, Custom Operating System for the 64 mode of the C128 - Only **$29.95!**

SUPER AIDE

All-purpose utility program for the C64 provides:

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

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

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

1541/1571 DRIVE ALIGNMENT — *Compute!'s Gazette Dec., 1987*

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

SUPER 81 UTILITIES

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

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

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

Super 81 Utilities is now available for the C64!

$39.95!

RAMDOS Lightning Fast RAM-DISK

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

GAMES

EYE OF THE INCA

Four text adventures on one disk for the C64 and Apple II series computers. Eye of the Inca, Shipwrecked, Son of Ali Baba and Perils of Darkest Africa. Four perilous adventures for only **$19.95!**

REVENGE OF THE MOON GODDESS

Four text adventures on one disk for the C64 and Apple II series computers. Revenge of the Moon Goddess, Frankenstein's Legacy, Night of the Walking Dead and The Sea Phantom. Four terrifying adventures for only **$19.95!**

SEX VIXENS FROM SPACE

Three text adventures for the C64 and Apple II series for MATURE ADULTS ONLY. Sex Vixens from Space, Bite of the Sorority Vampires and Hatchet Honeymoon. Three sizzling adult adventures for only **$29.95!**

SUPER BIKE

Action-packed, fun-filled motor cycle arcade game for the C64. Race the clock in Motocross, Enduro, Supercross or Trials. Fly through the air on spectacular jumps. Bounce over woop-de-doos. Avoid logs, trees, water holes, brick walls, other bikers, etc. as you vie for the gold cup.

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

GALACTIC FRONTIER

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

Only **$29.95!**

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

MasterCard VISA

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

In England contact Financial Systems Software 0905-611-463

A CP/M Memory Map

by Miklos Garamszeghy

```
C-128 CP/M Memory Map
---------------------
Rev 2   (c) M. Garamszeghy  1988
Herne Data Systems Ltd.
Toronto  Ont
```

The memory map of the C-128 is complicated to say the least. It is even more complicated in CP/M mode because of the variety of RAM, ROM, and input/output chips used by the CP/M operating system. This article attempts, for the first time, to map out the memory configurations used in CP/M mode. It is not claimed to be complete, but is a very useful starting point for your own explorations. The labels for the various memory locations are taken from the CP/M system source code files.

It is difficult to produce a detailed memory map of the CP/M operating system because much of the operating system is RAM based, and therefore subject to quick and easy revisions. There are currently 4 versions of C-128 CP/M generally available (apart from various beta test versions), as denoted by the dates displayed on boot up or by pressing the <F8> key. There are some major differences in the memory maps of each version. For clarity, these are defined here as:

AUG = Version dated 1 Aug 85
DEC = Versions dated 6 Dec 85 or 8 Dec 85
MAY = Version dated 28 May 87

There are also significant differences in the memory map depending on which processor (the Z-80 or 8502) is currently in use. This is important even in CP/M mode because most of the low level BIOS routines, such as standard serial port operations (some disk operations and all printer output), RAM disk operations, etc. use the 8502 mode.

PART 1: Z-80 operations

BANK 0 : MMU configuration register value $3f

BANK 0 is the system bank. The CP/M BIOS and BDOS operate primarily in this bank. System calls are made by transient programs via the common memory.

```
0000 to 0FFF     Z-80 ROM code

1000 to 128F  SYSKEYAREA   ; string data for programmable keys
                             each key can be defined as a string.
                             definitions "float" (i.e. you do not have
                             to adjust any pointers elsewhere) and are
                             terminated by a zero byte.  Vector at FD0B
                             points to this table.

1290 to 13EF  KEYCODES     ; ASCII codes for each key, 4 values for each key
                             (normal, "alpha mode", shift and control)
                             arranged according to key scan code table
                             on pg 687 of C-128 Programmer's Reference Guide.
                             Note that "alpha mode" defaults to uppercase
                             and is toggled on and off by pressing the C=
                             logo key.  It is equivalent to a software
                             "caps lock", but is not related to either
                             the <caps lock> or <shift lock> keys.
                             Vector at FD09 points here.
```

Value	Meaning
0	null (equivalent to no key press)
1 to 7f	normal ASCII codes (letters, numbers, symbols, etc)
80 to 9f	key has been programmed as a string, defined in SYSKEYAREA (32 possible "programmable" keys)
a0 to af	80 col character color (ctrl with number keys on main keyboard)
b0 to bf	80 col background color (ctrl with number keys on numeric keypad)
c0 to cf	40 col character color
d0 to df	40 col background color
e0 to ef	40 col border color
f0	toggle disk status line on/off (ctrl-run/stop)
f1	system pause (no-scroll key)
f2	track cursor on 40 col screen (ctrl-no-scroll)
f3	move 40 col window left 1 char (ctrl <- is defined as 4 f3's)
f4	move 40 col window right 1 char (ctrl -> is defined as 4 f4's)
f5	unlock MFM disk types (ctrl-home)
f6	select ADM31 emulation mode (ctrl- on numeric keypad) (MAY version only)
f7	select VT100 emulation mode (ctrl+ on numeric keypad) (MAY version only)
f8 to fe	not currently defined reserved for future expansion
ff	system cold re-start (control-enter)

(note: when bit 7 of FD22 (STATENABLE) is on, key codes of $80 and greater are returned without executing special function as outlined above. FD4C contains a vector to the address of the table of execution addresses of key codes f0 to ff. This vector can be changed to point to your own code.)

```
13F0 to 13FF  COLORTBL     ; logical color values 00, 11, 22, ... FF
                             for use with (esc-esc-esc) color value.
                             vector at FD0D points here.

1400 to 1BCF  SCREEN40     ; pseudo 80 column screen character buffer for 40
                             column screen

1C00 to 23CF  COLOR40      ; pseudo 80 column screen color buffer for 40
                             column screen


2400 BANKPARMBLK           ; (system parameters and flags)
2402 CUROFFSET             ; position counters used by psuedo 80 col screen
2404 OLDOFFSET             ; for 40 col screen
2405 PRTFLG                ;
2406 FLASHPOS
2408 PAINTSIZE             ; number of rows to move over for 40 col screen
                             shift ($18 or $19)
2409 CHARADR40             ; pointer to current char in pseudo screen RAM
240B CHARCOL40             ; 40 col character position - column      0-79
240C CHARROW40             ;                           - screen row 0-24
240D ATTR40                ; 40 col attribute  (character color)
240E BGCOLOR40             ;       background color
240F BDCOLOR40             ;       border color
2410 REV40                 ; reverse video flag

2411 CHARADR               ; pointer to current char in 80 col RAM
2413 CHARCOL               ; 80 col character position - column      0-79
2414 CHARROW               ;                           - screen row 0-24
2415 CURRENTATR            ; 80 col attribute
                             bit : 7 - 0 = alternate (block graphics) char set
                                   1 = ASCII character set
                                   6     reverse video 1 = on, 0 = off
                                   5     underline       "     "
                                   4     blink           "     "
                                   3     red             "     "
                                   2     green           "     "
                                   1     blue            "     "
                                   0     intensity       "     "

2416 BGCOLOR80             ;       background color
2417 CHARCOLOR80           ;       character color

2418 PARMBASE              ; pointers to currently active attribute sets
```

C-128 SUPER SORTER

by Patrick Edwards

Did you ever want a sort that would sort a string array on more than one field; and super fast? Well, it has arrived. This sort will let you sort on five (5) fields at the same time and sorts a 1000 items in about 15 seconds.

THE LOADER

First type in and save the loader program. To create the sort on disk insert the disk you want the sort on and type RUN. The loader creates two (2) program files on the disk. The first file is "sort.set". This file sets up the pointers and data used by "sort", then calls the sort.

USING THE SORT IN YOUR PROGRAMS

Once the sort is on disk it is available for use in your programs. To load the sort into memory use the following BASIC line:

POKE48,8:CLR:BLOAD"sort.set",B0:BLOAD"sort",B1

The actual sort resides in bank 1 (variable storage) of the C-128. The poke moves the start of variables to protect the sort program from being destroyed. The program "sort.set" resides in memory from $1300 to $1392.

Several parameters must be passed to the sort before it can start executing. The first five parameters are the numbers of the fields to be sorted, then comes the number of used records, followed by maximum records allowed plus 1, then number of fields and lastly the starting element of the array. The following BASIC statement will call the sort; there are two irregularities:

BANK15:SYS4864,,,,,field1,field2,field3,field4,field5,used records,max # of records + 1,# fields,array$(1,field1)

The first thing that looks odd about this SYS statement is the five (5) commas after the address. These are required because of the enhanced SYS on the 128. They bypass the built-in parameters. The next irregularity is the parameters that are passed to the sort.

Field1 thru Field5 - These are the fields in the array to be sorted. They must have a value of 0 to 255, and can be a variable or formula. All five of these must be present. Use a zero to fill in the ones not needed.

Used records - This is the number of elements that have data in them. It can be a variable or a formula.

Max # of records - This is the size of the array. That is to said the number of elements in the array. For this parameter use the first number in the DIM statement for the array to be sorted and add 1.

fields - This is the second number in the DIM statement for the array to be sorted.

Array$(1,field1) - This is used to find a pointer to the first element in the array to be sorted. The first number must be a one or the sort will give unpredictable results.

All of these parameters, except the last one (the array name), can be floating point values, integers or formulas.

RULES OF THE GAME

In order for the sort to work properly the array must be defined this way:

DIM name$(max records + 1,number of fields + 1)

This sort sorts only in ascending order but if you need descending order just access the array from the bottom up. i.e. element 250,249,248. For further examples type in the sort.create and sort.demo programs below:

SORT.CREATE:

```
0 POKE 48,8:CLR
1 RESTORE 100:BANK 0:FOR T= 0 TO 145:READ A:N=N+A:POKE4864+T,A:NEXT
2 IF N<>11554 THEN PRINT"ERROR IN DATA LINES 100-118":END
3 RESTORE 200:BANK1:N=0:FORT= 0 TO 573:READ A:N=N+A:POKE1024+T,A:NEXT
4 IF N<>54017 THEN PRINT"ERROR IN DATA LINES 200-271":END
5 PRINT"INSERT DISK YOU WANT THE SORT ON":PRINT"PRESS ANY KEY WHEN READY"
6 BSAVE"SORT.SET",B0,P4864 TO P5011:BSAVE"SORT",B1,P1024 TO P1599
99 REM SORT.SET
100 DATA 160,031,169,000,153,080,002,136
101 DATA 016,250,032,100,019,141,088,002
102 DATA 032,100,019,141,089,002,032,100
103 DATA 019,141,090,002,032,100,019,141
104 DATA 091,002,032,100,019,141,092,002
105 DATA 032,100,019,141,084,002,142,085
106 DATA 002,032,100,019,141,080,002,142
107 DATA 081,002,032,100,019,141,094,002
108 DATA 142,095,002,032,114,019,141,096
109 DATA 002,142,097,002,169,001,133,002
110 DATA 169,004,133,003,169,000,133,004
111 DATA 008,104,133,005,032,110,255,165
112 DATA 005,072,040,096,032,125,019,032
113 DATA 150,175,032,012,175,165,022,166
114 DATA 023,096,032,125,019,032,150,175
115 DATA 165,073,166,074,096,169,044,160
116 DATA 000,133,121,032,201,003,197,121
117 DATA 208,003,076,128,003,162,011,108
118 DATA 000,003,-1
199 REM SORT
200 DATA 238,080,002,173,080,002,208,003
201 DATA 238,081,002,141,082,002,173,081
202 DATA 002,141,083,002,160,002,024,173
203 DATA 082,002,109,080,002,141,082,002
204 DATA 173,083,002,109,081,002,141,083
205 DATA 002,136,208,234,056,173,084,002
206 DATA 233,001,141,084,002,173,085,002
207 DATA 233,000,141,085,002,173,088,002
208 DATA 141,087,002,169,001,141,086,002
209 DATA 173,096,002,141,098,002,173,097
210 DATA 002,141,099,002,160,003,024,173
211 DATA 098,002,109,084,002,141,098,002
212 DATA 173,099,002,109,085,002,141,099
213 DATA 002,136,208,234,173,096,002,133
214 DATA 080,133,082,173,097,002,133,081
215 DATA 133,083,173,098,002,133,084,173
216 DATA 099,002,133,085,173,084,002,141
217 DATA 100,002,173,085,002,141,101,002
218 DATA 024,110,101,002,110,100,002,032
219 DATA 004,005,176,001,096,160,003,024
220 DATA 165,082,109,100,002,133,082,165
221 DATA 083,109,101,002,133,083,136,208
222 DATA 238,169,000,141,102,002,032,028
223 DATA 005,032,017,005,032,050,005,176
224 DATA 008,032,076,005,169,001,141,102
225 DATA 002,032,039,005,144,025,173,096
226 DATA 002,133,080,133,082,173,097,002
227 DATA 133,081,133,083,173,102,002,240
228 DATA 003,076,157,004,076,144,004,024
229 DATA 165,080,105,003,133,080,165,081
230 DATA 105,000,133,081,024,165,082,105
231 DATA 003,133,082,165,083,105,000,133
232 DATA 083,076,182,004,173,101,002,201
233 DATA 000,208,005,173,100,002,201,001
234 DATA 096,160,002,177,082,153,251,000
235 DATA 136,016,248,096,160,002,177,080
236 DATA 153,139,000,136,016,248,096,165
237 DATA 083,197,085,208,004,165,082,197
238 DATA 084,096,160,255,200,196,251,176
239 DATA 011,196,139,176,006,177,252,209
240 DATA 140,240,241,096,196,139,176,001
241 DATA 096,076,227,005,032,185,005,032
242 DATA 145,005,174,094,002,160,002,177
243 DATA 082,153,104,002,177,080,145,082
244 DATA 185,104,002,145,080,136,016,239
245 DATA 032,114,005,202,208,231,032,206
246 DATA 005,096,024,165,080,109,082,002
247 DATA 133,080,165,081,109,083,002,133
248 DATA 081,024,165,082,109,082,002,133
249 DATA 082,165,083,109,083,002,133,083
250 DATA 096,172,087,002,136,240,033,056
251 DATA 165,080,237,082,002,133,080,165
252 DATA 081,237,083,002,133,081,056,165
253 DATA 082,237,082,002,133,082,165,083
254 DATA 237,083,002,133,083,136,208,223
255 DATA 096,165,080,141,107,002,165,081
256 DATA 141,108,002,165,082,141,109,002
257 DATA 165,083,141,110,002,096,173,107
258 DATA 002,133,080,173,108,002,133,081
259 DATA 173,109,002,133,082,173,110,002
260 DATA 133,083,096,238,086,002,172,086
261 DATA 002,192,003,176,003,032,185,005
262 DATA 192,003,144,002,104,104,185,087
263 DATA 002,240,034,032,145,005,172,086
264 DATA 002,185,087,002,141,087,002,032
265 DATA 037,006,032,017,005,032,028,005
266 DATA 032,050,005,176,008,032,050,006
267 DATA 032,206,005,024,096,032,050,006
268 DATA 032,206,005,056,096,172,087,002
269 DATA 136,240,006,032,114,005,136,208
270 DATA 250,096,160,001,140,086,002,185
271 DATA 087,002,141,087,002,096,-1
```

Continued on Page 17

FUN WITH SEQ FILES

by Miklos Garamszeghy

SYS 65478 (that's JSR $FFC6 to you machine language fans) seems like an innocent and well documented routine. Opening a file as an input channel. What could be simpler? While this may be its normal use from within a program, the KERNAL CHKIN routine can be put to some much less obvious uses in immediate mode. One of the more interesting of these enables a Commodore 128 (and other CBM machines with a similar KERNAL structure) to execute a series of commands contained in a disk file, similar to an MS-DOS batch file or a CP/M SUBMIT file.

The CHKIN routine resets the input device flag (normally 0 to indicate the keyboard) at zero page location hex $99 on the C-128 to a value corresponding to the device from which normal input would be received. The main BASIC immediate mode input loop checks this location before trying to fetch an input byte. If the value is 0, a normal entry occurs by fetching a byte from the keyboard buffer. However, if the value is 8, for example, the fetch routine will attempt to read a byte from serial port device 8 (usually a disk drive). If device 8 has an open file capable of giving output, a byte is read from this file and placed in BASIC'S input buffer, just as if it had been entered from the keyboard. If a carriage return is encountered, the commands contained in the input buffer are executed, assuming there is no line number at the beginning. Thus you can execute commands contained in a disk file. It should be noted that even with normal input redirected to respond to an external device, the keyboard is still scanned by the IRQ routines and the results placed in the keyboard buffer only, up to the maximum number of characters allowed for the buffer. BASIC'S input editor will not see any of these characters until control has been restored to the keyboard.

So what, you say? Everyone knows an easier way to do that: Its called a program or PRG file. Let's make one thing perfectly clear. Executing what I call a sequential disk command (SDC) file is not the same thing as LOADing and RUNning a disk PRG program file. There are several major differences: A regular PRG type file contains a series of tokenized BASIC lines (complete with link addresses and line numbers) or machine language op codes. The sequential disk command file, on the other hand, contains a series of immediate mode commands written out in plain English, just as you would type them in from the keyboard. In addition, a PRG file must be resident in RAM for execution. In most 8 bit computer operating systems (Commodore KERNALS included), only one program can be in memory for execution at a given time (assuming that you have not artificially partitioned the memory into separate work spaces.) A SDC type program is not memory resident! It resides entirely in a disk file and is called up and executed statement by statement, without affecting any program(s) stored in the computer's main RAM unless you deliberately want to. However, since it resides on disk, a SDC program usually takes longer to execute than a RAM resident program because of Commodore's notoriously slow disk access speeds. Even this can be minimized, however, through the use of fast drives like the 1571 or 1581 or even some RAM disks.

SDC's are useful as utility and easy to use reference data files since they can be called up and executed without fear of erasing main computer memory. This allows a programmer to interrupt work, call up and consult an on-line data table, for example, and then resume the task at hand, all with relative ease and speed. SDC's can also be used for storing customized keyboard macros. (For those unfamiliar with the concept, a keyboard macro is an often lengthy and/or complicated series of frequently used keystrokes/commands that can be automatically invoked by using a shorter, easier to remember key sequence. Can you honestly remember the POKE sequence to play the first few bars of Happy Birthday with the SID chip?)

Setting up an SDC

A sequential disk command file is very easy to create. You use your favorite word processing program to create a SEQ file containing a series of immediate mode BASIC commands, just as you would type them in to execute from the keyboard. BASIC keywords (such as PRINT) can be entered in either their long or abbreviated forms. Of course, the same limits on line length as for normal programming apply to the lines in your SDC file (e.g. 160 characters for the C-128, etc). This is a restriction imposed by the size of the input buffer on the computer.

Any immediate mode command can be used except for disk access commands. You should not use OPEN, LOAD, SAVE, DIRECTORY, etc. because these commands will reset the input device to the default keyboard value after they have executed, thus cutting off the rest of your command file. A DIRECTORY can be used as the last item in an SDC because it will return control to the keyboard upon execution which is desirable in this case.

The program mode only commands, such as INPUT, GET, GOTO, GOSUB, etc., cannot be used in SDC's because an SDC is executed in immediate mode not under program control. Line numbers should not be used in SDC's (except on the 128 where they provide a different effect as outlined later).

In order to be properly interpreted when they are read in, the BASIC keywords must be typed in a style that allows them to be saved as un-shifted PETSCII characters in a disk file. (This is the way that you would normally enter them from the keyboard.) With most word processors, such as Paperclip or Pocket Writer-128, this means that they are typed in lower case only and the file is saved as a SEQ file. With word processors such as Timeworks Word Writer 128 which save text in true ASCII format, the BASIC keywords must be entered in upper case only. Word processors which use PRG type screen code files only should not be used for creating SDC's.

Making a graceful exit from an SDC back to keyboard input can be somewhat tricky. The easiest way is to include a statement at the end of your SDC which POKEs a 0 value back into the input device flag location described earlier. Simply CLOSEing the disk file from within the SDC or using the BASIC commands END or STOP, will not return control to the keyboard because they do not automatically reset the input device flag. Another way to exit is to include a garbage statement or deliberate syntax error as the last line. Upon reaching such an error, the SDC will crash and control will be returned to the keyboard. The least elegant way to exit is by the familiar <run-stop>/<restore> key combination. Crude but effective.

Once the SDC has been entered, it should be saved as a SEQ disk file with an appropriate name.

Executing the SDC

Now comes the fun part. Executing the SDC is really quite simple. All that is required is to open the disk file in immediate mode with a statement such as:

OPEN 1,8,8,"filename"

Second, you must activate the CHKIN routine. On the C-128, with the above OPEN statement you would use:

SYS 65478,0,1

The commands in the SDC will then be executed, one line at a time until control is returned to the keyboard by one of the methods previously outlined. You will note that the actual commands are not printed to the screen before they are executed, but the "READY." message is printed after each line has been executed. Once the SDC has finished executing, the disk file should be closed with a DCLOSE or CLOSExx as applicable for your machine.

Continued from Page 16

Merging Programs on the C-128

The input routine in the C-128 KERNAL will accept line numbers in SDC's. These line numbered files will "execute" exactly as if the line, complete with line number, had been entered from the keyboard. That is, it will be added to any program currently in memory. This little trick is a simple yet powerful way to MERGE two or more program files on the C-128. Unlike other pseudo merge routines which merely append one program to the end of another, this technique allows full intermeshing of line numbers. Only a few steps are required. First, LOAD one of the programs into memory in the normal manner. Next convert it into a SEQ disk file listing with a series of commands such as:

OPEN 8,8,8,"0:filename,S,W":CMD8:LIST
PRINT#8:CLOSE8

LOAD in the second file. With the second file now in memory, activate the SEQ listing of the first file as a SDC as outlined above. The programs will now be MERGEd. The merge will terminate with a mysterious "OUT OF DATA" error message. This is a good sign: the process worked. The error message is caused by the last line of the listing in the disk file "READY.". (Commodore BASIC listings always include this.) The computer, not being able to recognize its own writing, thinks that someone typed in "READ Y". Since there are no accompanying DATA statements, the out of data error occurs and control is restored to the keyboard. If the READY. message did not appear at the end of the listing (for example if you edited it out with a word processor to give it a neater appearance), keyboard control would only be restored with a <run-stop>/<restore>. The programs will, however, be merged correctly if you wait until all the statements have been read from the disk file. This may take some guess work on your part. This technique can also be used for "loading" program listings produced on machines with incompatible keyword tokens and programs transferred into SEQ files via a modem download.

Example SDC's

LISTINGs 1, 2 and 3 are short examples of SDC's. While it may appear that some of the statements are repetitive, it should be remembered that they were created with a word processor. (If your word processor does not have cut, paste and copy commands, then perhaps it is time to splurge for a new one!!) LISTING 1 is an example which prints out a simple calendar for the month of January 1989. You will note that most of the lines begin with the sequence "print up$". "up$" is defined in the first line as three cursor ups.

FUN WITH SEQ FILES

This allows you to get around the nasty habit of immediate mode BASIC of printing a few carriage returns and a READY after each line it executes. up$ is included to properly format the screen display in this case. Note also that special control characters are given as their chr$() values. This is the only way to enter them with a word processor.

LISTING 2 will print out a handy hex-dec conversion chart on the 80 column screen of the C-128. It is similar in nature to the above example, but works in C-128 FAST mode. The 80 column screen is required because of the width of the table.

LISTING 3 is interesting for several reasons. It is essentially a self contained data file which can read and display itself on the screen automatically! The WAIT and POKE values of 208 in the lines are set up for a C-128. This is the location of the keyboard buffer flag which indicates the number of characters in the buffer. The file will display one entry at a time and wait for you to press a key before displaying the next entry. (Remember, the keyboard scan and buffer filling still occurs when a SDC is being executed even if its input is ignored by the BASIC input routine, hence the need to clear the buffer by POKEing a 0 to it before reading the next entry.) To stop the display before it reaches the end, a <run-stop>/<restore> should be used. This last example demonstrates that although you cannot read the keyboard directly with GET's, INPUT's, etc, you can still obtain data from the keyboard via direct PEEK's and POKE's to the keyboard buffer areas.

LISTING 4 is one final fun application. I will not tell you what it does, but run it on the C-128 in 40 column mode and see for yourself. Note that the 4 groups of "play" statements are identical. You need only type them in once, then copy them with your word processor.

Variations on a Theme

The re-directed input is not limited to disk files. The procedure works equally as well with the user port. You could, in theory, control your computer in immediate mode from a remote location with an external keyboard or even a modem and an auxiliary terminal.

See listings on next page

Super Sort Continued from Page 15

SORT.DEMO:

```
 5 POKE48,8:CLR:BLOAD"SORT.SET",B0:BLOAD"SORT",B1
10 DIMA$(11,5):FOR T= 1 TO 10:FOR A=1 TO 5:READ A$(T,A):NEXT:NEXT
15 PRINT"** ORIGINAL ORDER":FOR T=1 TO 10:FOR A=1 TO 5:PRINTA$(T,A)" ";:NEXT:PRINT:NEXT
20 PRINT"** SORTED BY FIELD NUMBER 1":BANK15:SYS4864,,,,,1,0,0,0,0,10,10+1,5,A$(1,1)
25 FOR T=1 TO 10:FOR A=1 TO 5:PRINTA$(T,A)" ";:NEXT:PRINT:NEXT:PRINT"PRESS ANY KEY":GETKEYB$
30 PRINT"** ORIGINAL ORDER":FOR T=1 TO 10:FOR A=1 TO 5:PRINTA$(T,A)" ";:NEXT:PRINT:NEXT
35 PRINT"** SORTED BY FIELD NUMBER 2 AND 1":BANK15:SYS4864,,,,,2,1,0,0,0,10,10+1,5,A$(1,2)
40 FOR T=1 TO 10:FOR A=1 TO 5:PRINTA$(T,A)" ";:NEXT:PRINT:NEXT:PRINT"PRESS ANY KEY":GETKEYB$
45 PRINT"** ORIGINAL ORDER":FOR T=1 TO 10:FOR A=1 TO 5:PRINTA$(T,A)" ";:NEXT:PRINT:NEXT
50 PRINT"** SORTED BY FIELD NUMBER 3 AND 2":BANK15:SYS4864,,,,,3,2,0,0,0,10,10+1,5,A$(1,3)
55 FOR T=1 TO 10:FOR A=1 TO 5:PRINTA$(T,A)" ";:NEXT:PRINT:NEXT:PRINT"PRESS ANY KEY":GETKEYB$
60 PRINT"** ORIGINAL ORDER":FOR T=1 TO 10:FOR A=1 TO 5:PRINTA$(T,A)" ";:NEXT:PRINT:NEXT
65 F=5:PRINT"** SORTED BY FIELD NUMBERS 5,2,1":BANK15:SYS4864,,,,,F,F-3,F-4,0,0,10,10+1,F,A$(1,F)
70 FOR T=1 TO 10:FOR A=1 TO 5:PRINTA$(T,A)" ";:NEXT:PRINT:NEXT:PRINT"PRESS ANY KEY":GETKEYB$
100 DATA BUGS,BUNNY,LA,CA,90053,MICKY,MOUSE,DISNEYLAND,CA,99999,FRANK,BURNS,MASH,KOREA,11111
101 DATA PAT,EDWARDS,NASHVILLE,TN,37209,ALEX,KEATON,COLUMBUS,OH,66666,MINI,MOUSE,DISNEYWORLD,FL,22222
102 DATA GOFFY,DOG,DISNEYLAND,CA,99999,DAN,RATHER,CBS,NY,77777,AMY,EDWARDS,NASHVILLE,TN,37209
103 DATA DAFFY,DUCK,ANYWHERE,US,00000
```

FUN WITH SEQ FILES

Continued from Page 17

Listing 1: Calendar

```
up$=chr$(145)+chr$(145)+chr$(145)
poke53280,6:poke53281,6
printchr$(147)chr$(5);
print up$"        january 1989"
print up$" sun  mon  tues  wed thur  fri  sat"
print up$"+----+----+----+----+----+----+----+"
print up$"  1    2    3    4    5    6    7  "
print up$"+----+----+----+----+----+----+----+"
print up$"  8    9    10   11   12   13   14"
print up$"+----+----+----+----+----+----+----+"
print up$"  15   16   17   18   19   20   21"
print up$"+----+----+----+----+----+----+----+"
print up$"  22   23   24   25   26   27   28"
print up$"+----+----+----+----+----+----+----+"
print up$"  29   30   31"
print up$"+----+----+----+----+----+----+----+"
poke 153,0
```

Listing 2: HEX-DEC converter (80 column only)

```
up$=chr$(145)+chr$(145)+chr$(145)
printchr$(147)chr$(5);
fast
print up$"          hex-dec converter"
print up$"       00  01  02  03  04  05  06  07  08  09  0a  0b  0c  0d  0e  0f"
print up$"+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+"
print up$" 00 :  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15"
print up$" 10 : 16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31"
print up$" 20 : 32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47"
print up$" 30 : 48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63"
print up$" 40 : 64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79"
print up$" 50 : 80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95"
print up$" 60 : 96  97  98  99 100 101 102 103 104 105 106 107 108 109 110 111"
print up$" 70 :112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127"
print up$" 80 :128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143"
print up$" 90 :144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159"
print up$" a0 :160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175"
print up$" b0 :176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191"
print up$" c0 :192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207"
print up$" d0 :208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223"
print up$" e0 :224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239"
print up$" f0 :240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255"
print up$:slow:poke 153,0
```

Listing 3: C-128 Phone File

```
up$=chr$(145)+chr$(145):printchr$(147)chr$(5);
print up$"         mini phone file  ":print
print up$" name: john doe    phone:123-456-7890":wait 208,1:poke208,0
print up$" name: mary brown  phone:123-456-7890":wait 208,1:poke208,0
print up$" name: jim smith   phone:123-456-7890":wait 208,1:poke208,0
print up$" name: jane green  phone:123-456-7890":wait 208,1:poke208,0
print up$" name: bill black  phone:123-456-7890":wait 208,1:poke208,0
print up$" name: fred right  phone:123-456-7890":wait 208,1:poke208,0
print up$" name: sally ho    phone:123-456-7890":wait 208,1:poke208,0
poke153,0
```

Listing 4: (Surprise)

```
color0,1:color4,1:color1,6:color2,3:color3,10:graphic3,1
draw,80,15to90,30to85,30to95,50to90,50to110,90to50,90to70,50to65,50to75,30to70,30to80,15
paint,80,50:box3,77,90,83,100:paint3,80,95
circle2,90,32,2,2:paint2,90,32
circle2,95,52,2,2:paint2,95,52
circle2,110,92,2,2:paint2,110,92
circle2,50,92,2,2:paint2,50,92
circle2,65,52,2,2:paint2,65,52
circle2,70,32,2,2:paint2,70,32:color2,7
circle2,85,75,2,2:paint2,85,75
circle2,75,50,2,2:paint2,75,50
circle2,82,40,2,2:paint2,82,40:color2,8
circle2,70,60,2,2:paint2,70,60
circle2,105,80,2,2:paint2,105,80
circle2,90,65,2,2:paint2,90,65:color2,2
circle2,65,85,2,2:paint2,65,85
circle2,80,30,2,2:paint2,80,30
circle2,80,12,3,3:paint2,80,12
char,5,21,"m e r r y  c h r i s t m a s"
tempo30

play"v1 o5 t0 u12 x0":play"v2 o3 t0 u12 x0"
play"v1 o5 i#d d qd i#d d qd i#d d"
play"v1 o5 qd v2 o3 wg v1 o5 h#a i#a a"
play"v1 o5 qg v2 o3 wg v1 o5 ig f q#d i#d d"
play"v1 o5 qc v2 o3 w#d v1 o5 hc id c"
play"v1 o5 qc v2 o3 w#d v1 o5 id c qc id c"
play"v1 o5 qc v2 o3 wc v1 o5 ha ia g"
play"v1 o5 q#f v2 o3 wd v1 o5 i#f #d qd id c"
play"v1 o4 q#a v2 o2 wg v1 o4 h#a"
```

(Editor's Note: Listing 4 is present in dual column format due to space limitations. When creating your seq file you should type in the left column followed immediately by the right and not in dual column format.)

```
play"v1 o5 t0 u12 x0":play"v2 o3 t0 u12 x0"
play"v1 o5 i#d d qd i#d d qd i#d d"
play"v1 o5 qd v2 o3 wg v1 o5 h#a i#a a"
play"v1 o5 qg v2 o3 wg v1 o5 ig f q#d i#d d"
play"v1 o5 qc v2 o3 w#d v1 o5 hc id c"
play"v1 o5 qc v2 o3 w#d v1 o5 id c qc id c"
play"v1 o5 qc v2 o3 wc v1 o5 ha ia g"
play"v1 o5 q#f v2 o3 wd v1 o5 i#f #d qd id c"
play"v1 o4 q#a v2 o2 wg v1 o4 h#a"

play"v1 o5 t0 u12 x0":play"v2 o3 t0 u12 x0"
play"v1 o5 i#d d qd i#d d qd i#d d"
play"v1 o5 qd v2 o3 wg v1 o5 h#a i#a a"
play"v1 o5 qg v2 o3 wg v1 o5 ig f q#d i#d d"
play"v1 o5 qc v2 o3 w#d v1 o5 hc id c"
play"v1 o5 qc v2 o3 w#d v1 o5 id c qc id c"
play"v1 o5 qc v2 o3 wc v1 o5 ha ia g"
play"v1 o5 q#f v2 o3 wd v1 o5 i#f #d qd id c"
play"v1 o4 q#a v2 o2 wg v1 o4 h#a"

play"v1 o5 t0 u12 x0":play"v2 o3 t0 u12 x0"
play"v1 o5 i#d d qd i#d d qd i#d d"
play"v1 o5 qd v2 o3 wg v1 o5 h#a i#a a"
play"v1 o5 qg v2 o3 wg v1 o5 ig f q#d i#d d"
play"v1 o5 qc v2 o3 w#d v1 o5 hc id c"
play"v1 o5 qc v2 o3 w#d v1 o5 id c qc id c"
play"v1 o5 qc v2 o3 wc v1 o5 ha ia g"
play"v1 o5 q#f v2 o3 wd v1 o5 i#f #d qd id c"
play"v1 o4 q#a v2 o2 wg v1 o4 h#a"

dclose:poke153,0
```

ERRATA FROM ISSUE #20:

1. Avonelle made a slight mistake in here review of Fontmaster 128, for the dozens of you who wrote us to tell us that by hitting CTRL-L and then the HELP key you get a directory from which you can choose and load your file, we stand corrected.

2. We accidentally left off ordering information for the Church Management Database. The author of the article, Duane Baun, wrote to point out the error. To order this Superbase application, please write to: Rev. Robert Goldstein, 4630 W. 59th Terrace, Fairway, KA 66205.

A CP/M Memory Map Continued

by Miklos Garamszeghy

```
241A PARMAREA80
241D PARMAREA40
2420 BUFFER80COL

2471 KEYBUF              ; buffer for currently pressed key code

2488 CONTROL CODES       ; flag for control/shift keys pressed

                           bit        key pressed
                           ----------------------
                            2         control key
                            4         right shift key
                            5         C= key
                            7         left shift key

                           0 and 1     character mode
                            00     =    lower case
                            01     =    alpha mode
                            10     =    shift
                            11     =    control

2489 MSGPTR              ; pointer to current function key message string

248B OFFSET              ; cursor pointers used by various screen
248C CURPOS              ; printing routines

248E SYSFREQ             ; power line frequency : 0 = 60 Hz, FF = 50 Hz

2600-2a40   BIOS8502     ; 8502 BIOS code (see map of 8502 mode below for
                                description)

2C00 to 2fff      VICSCREEN   ; 40 column video RAM, also appears in hardware
                                I/O area and Bank 2.  (Note this is separate
                                from normal C-128 40 col video RAM at $0400
                                which is unused in CP/M mode because it is
                                under the Z-80 ROM.)

3000  to 3cff     CCPBUFFER         ; CCP.COM hides here during TPA execution

3C00 BOOTPARM    ; various flags etc used during cold boot
3C02 LDBLKPTR
3C04 BLKUNLDPTR
3C06 BLOCKSIZE
3C07 BLOCKEND
3C09 BLOCKPTRS
3C29 INFOBUFFER
3C35 EXTNUM
3C36 RETRY
3C77 BOOTSTACK

3d00 BANK0FREE          ; BANK 0 work area can be used by transient programs
                          extends to 98FF on Dec/85 and May/87 CP/M versions
                                     9BFF on Aug/85 version
                          Primarily used by CP/M as directory and file
                          buffers.  Programmers can use, but beware of
                          implications.
```

Operating system areas:

Component	DEC	MAY	AUG
Bank BDOS	9900	9900	9C00
Resident BDOS	EA00	EA00	EE00
Bank BIOS	C700	C800	CA00
Resident BIOS	F000	F000	F400

MFM disk parameter table (DPT) is located at d876 to da75 in AUG version
d6bd to d8bc in DEC versions
and d860 to da5f in MAY version

The DPT structure is described in "Inside C-128 CP/M" (Transactor vol 8/4, pg 43.) It contains the basic information to allow access to foreign MFM disk types. The DPT can be found by the pointer at FD46 (all versions).

The other major component of accessing disk drives is the disk parameter header or DPH. This is the working area used by the BDOS for actual disk access. When a disk is logged in, the appropriate values are copied from the DPT to the DPH for general use. The drive table address which contains the vectors to the DPH for each logical drive can be found at BIOS.BASE+D7. This contains a series of address pointers to the DPH.BASE for each logical drive (A: to P:). If a drive is not supported (drives F: to L: and N: to P:), the drive table value is 0. For each drive the $37 byte long DPH has the following format:

```
DPH.BASE-A  pointer to the sector write routine for this drive
DPH.BASE-8  pointer to the sector read routine for this drive
DPH.BASE-6  pointer to the login routine for this drive
DPH.BASE-4  pointer to the initialization routine for this drive
DPH.BASE-2  physical drive assigned to the logical drive according to values
            listed below under VICDRV.
DPH.BASE-1  secondary disk type byte from byte 2 of DPT entry.
DPH.BASE    pointer to logical to physical sector skew table (0000 if none)
DPH.BASE+2  9 bytes of scratch pad for use by BDOS
DPH.BASE+B  media flag 0 if disk logged in, FF if disk has been changed
DPH.BASE+C  pointer to DPB values for this drive (contained later in entry)
DPH.BASE+E  pointer to CSV scratch pad area (used to detect changed disks)
DPH.BASE+10 pointer to ALV scratch pad area (used to keep track of drive
            storage capacity)
DPH.BASE+12 pointer to directory buffer control block (BCB)
DPH.BASE+14 pointer to data buffer control block
DPH.BASE+16 pointer to the directory hashing table (FFFF if not used)
DPH.BASE+17 bank for hash table
DPH.BASE+18 DPT entry for this drive
DPH.BASE+2A maximum sector number and MFM lock flag (bit 7 on if locked)
DPH.BASE+2B pointer to entry in master MFM DPT table

E000 Free memory in Common BANK 0 and 1.  Normally used by RSX (resident system
                                          extension) programs, as well as
                                          operating system extensions and SID,
                                          SUBMIT, SAVE, GET, PUT, etc
                                          Can be used by experienced
                                          programmers if you are aware of
                                          consequences, such as possible crash
                                          when using another program.

     Extends to E9FF in DEC and MAY versions and
                 EDFF in AUG version
```

System Control Block (SCB):

DEC and MAY Versions EF9C to EFFF
AUG Version F39C to F3FF

100 bytes containing basic system variables, located immediately before BIOS jump table, can be read or written with BDOS function 49. Basic parameters detailed in Appendix A of CP/M Plus Programmers Guide from Digital Research.

Byte Offset	Function
00 - 04	reserved for various system flags
05	BDOS version number
06 - 09	reserved for user determined flags (put your own stuff here)
0A - 0F	reserved for system use
10 - 11	16 bit program return code for passing data to chained programs
12 - 19	reserved for system use
1A	screen width - 1 (set to 79)
1B	current column position on screen (0 to 79)
1C	screen page length (24 lines per screen)
1D - 21	reserved for system use
22 - 2B	assignment vectors for CP/M's logical I/O devices. 16 bit value used as follows:

bit	device
f	KEYS (input only)
e	80COL (output only)
d	40COL (output only)
c	PRT1 (device 4 serial printer, output only)
b	PRT2 (device 5 serial printer, output only)

A CP/M Memory Map Continued

by Miklos Garamszeghy

```
a           6551   (not really supported, it was supposed to be on an
                    external card which was never produced)
9           RS232  (input or output)
8 to 0      not used
-------------------------------------------------------------------------
If a bit is on, then the physical device is assigned to that logical device.
Each logical device can have more than one physical device assigned to it and
each physical device can be assigned to more than 1 logical device.  Logical
devices and their assignment vectors are:

              22 - 23       CONIN
              24 - 25       CONOUT
              26 - 27       AUXIN
              28 - 29       AUXOUT
              2A - 2B       LSTOUT

2C                Page mode 0 = display 1 page of data at a time
                            1 = display continuously
                  (this is the annoying flag that causes CP/M TYPE
                   command to say "press return to continue" after
                   each screenful of data.)
2D                reserved for system use
2E                determines effect of CTRL-H.  0 = backspace and delete
                                                FF = delete and echo
2F                determines effect of delete   0 = delete and echo
                                                FF = backspace and delete
30 - 32           reserved for system use
33 - 34           16 bit console mode flag (default value = 0000):

                  bit         meaning
                  ---------------------------------
                   0          0 = return normal status
                                  for BDOS function 11
                              1 = CTRL - C only status
                   1          0 = enable CTRL-S CTRL-Q
                                  stop scroll/start scroll
                              1 = disable stop/start scroll
                   2          0 = normal console output
                              1 = raw console output,
                                  disables tab expansion,
                                  and CTRL-P printer echo
                   3          0 = enable CTRL-C program termination
                              1 = disable CTRL-C
                   8 - 9          used for RSX's

35 - 36            address of 128 byte scratch pad buffer
37                 output string delimiter (normally $)
38                 LIST output flag  0 = console output only
                                     1 = echo output to printer
39                 reserved for system use
3A - 3B            pointer to start of SCB
3C - 3D            current DMA (disk buffer) address
3E                 current drive (0 = A:, 1 = B:, etc.)
3F - 40            BDOS disk info flags
41                 FCB flag
43                 BDOS function where error occured
44                 current user number (0 to F)
45 - 49            reserved for system use
4A                 BDOS multi sector count for read/write
4B                 BDOS error mode: FF = system does not display
                                        error messages, returns
                                        to current program
                                    FE = display error messages,
                                        return to current program
                                  else  terminate current program on
                                        error and display message
4C - 4F           Drive search chain: up to 4 drives can be specified
                  0 = current default drive, 1 = A:, 2 = B:, etc
                  if program or file is not found on specified drive,
                  the search chain will be used and each drive in the
                  list will be tried in sequence until it is found.
50                Temporary file drive: 0 = default, 1 = A:, etc
51                Error drive: number of drive where last i/o error
                  was encountered (0 = default, 1 = A:, etc)
52 - 53           reserved for system use
54                BIOS flag to indicate disk changed
55 - 56           reserved for system use
57                BDOS flags: bit 7 set then system displays expanded
                  error messages (default is set)
58 - 59           date in days in binary since 1 jan 78
5A                hour in BCD
5B                minutes in BCD
5C                seconds in BCD
5D - 5E           start of common memory (E000)
5F - 61           reserved for system use
62 - 63           top of user TPA  (from vector at 0006 - 0007: entry point to
                                    BDOS)
-------------------------------------------------------------------------

BIOS Jump table:
----------------

(BIOS.BASE= F000 in DEC and MAY versions
           F400 in AUG version)

(Note: the first byte of each group is a Z80 jump instruction to the
       address contained in the next two bytes.)
                                                    BIOS Function number

BIOS.BASE+00 to +02  cold boot                                          0
BIOS.BASE+03 to +05  warm boot                                          1
BIOS.BASE+06 to +08  check CONSOLE input status                         2
BIOS.BASE+09 to +0b  read CONSOLE character                             3
BIOS.BASE+0c to +0e  write CONSOLE character                            4
BIOS.BASE+0f to +11  write LIST character                               5
BIOS.BASE+12 to +14  write AUXILIARY OUT character                      6
BIOS.BASE+15 to +17  read AUXILIARY INPUT character                     7
BIOS.BASE+18 to +1a  move to track 0 on selected disk                   8
BIOS.BASE+1b to +1d  select disk drive                                  9
BIOS.BASE+1e to +20  set track number                                  10
BIOS.BASE+21 to +23  set sector number                                 11
BIOS.BASE+24 to +26  set DMA address                                   12
BIOS.BASE+27 to +29  read specified sector                             13
BIOS.BASE+2a to +2c  write specified sector                            14
BIOS.BASE+2d to +2f  check LIST status                                 15
BIOS.BASE+30 to +32  translate logical to physical sector              16
BIOS.BASE+33 to +35  check CONSOLE output status                       17
BIOS.BASE+36 to +38  check AUXILIARY INPUT status                      18
BIOS.BASE+39 to +3b  check AUXILIARY OUTPUT status                     19
BIOS.BASE+3c to +3e  get address of character I/O table                20
BIOS.BASE+3f to +41  initialize character I/O devices                  21
BIOS.BASE+42 to +44  get address of disk drive table                   22
BIOS.BASE+45 to +47  set # of logical sectors to read/write            23
BIOS.BASE+48 to +4a  force I/O buffer flush                            24
BIOS.BASE+4b to +4d  memory move                                       25
BIOS.BASE+4e to +50  get or set time                                   26
BIOS.BASE+51 to +53  select memory bank                                27
BIOS.BASE+54 to +56  specify bank for DMA operation                    28
BIOS.BASE+57 to +59  set buffer bank                                   29
BIOS.BASE+5a to +5c  call user system functions                        30

BIOS.BASE+d3 to +d4  pointer to physical device table
                     (points to f7fd in AUG version and
                      f3e1 in DEC and MAY)

DEVTBL = Physical device table, each entry is 8 bytes long:

 device name (6 bytes, padded with spaces)
 mode byte   (1 byte)
 baud rate   (1 byte)

Mode byte component       device function
-----------------------------------------------
0000 0001                 device can do input
0000 0010                 device can do output
0000 0011                 device can do both
0000 0100                 supports software
                          selected baud rate
0000 1000                 supports serial i/o
0001 0000                 supports XON/XOFF
-----------------------------------------------

Baud rate byte          function
-----------------------------------------------
```

A CP/M Memory Map Continued

by Miklos Garamszeghy

```
   0                    no baud rate
   1                    50 baud
   2                    75
   3                    110
   4                    135
   5                    150
   6                    300
   7                    600
   8                    1200
   9                    1800   ; note: for all practical uses, the c-128 cp/m
   a                    2400          will not support anything over 1200 baud
   b                    3600          because of software overhead.
   c                    4800
   d                    7200
   e                    9600
   f                    19200
-------------------------------------------
C-128 devices are: KEYS, 80COL, 40COL, PRT1, PRT2, 6551, and RS232

BIOS.BASE+d7 to d8   pointer to drive table (List of DPH addresses).
                     points to FBD1 on DEC and MAY versions and FB78 on AUG.

FC00 to FD00  INTBLOCK    ; contains all FD's.  This is the pointer to the
                          ; interupt vector (FDFD). A hardware quirk of the
                          ; C-128 requires from FC00 to FD00 to be all FD's
                          ; else the interupt pointer address will not be
                          ; correctly specified on the Z-80 address bus.
                          ; If you use this seemingly unused area for your own
                          ; programs, it will crash on a random basis if an
                          ; interupt occurs.  BEWARE!!

PARMBLOCK                 ; BIOS parameter working storage for passing to 8502
FD01 VICCMD               ; BIOS 8502 command to execute
FD02 VICDRV               ; device number to execute on printer.
                            for disk i/o:

                          bit value    drive
                          ------------------------
                          0000 0001    unit 8, drive 0   (default drive A: and E:)
                          0000 0010    unit 9, drive 0   (default drive B:)
                          0000 0100    unit 10, drive 0  (default drive C:)
                          0000 1000    unit 11, drive 0  (default drive D:)
                          1000 0001    unit 8, drive 1
                          1000 0010    unit 9, drive 1
                          1000 0100    unit 10, drive 1
                          1000 1000    unit 11, drive 1

FD03 VICTRK               ; track number to execute disk operations on,
                            or secondary address for printer operations
FD04 VICSECT              ; sector number for disk operations.  for MFM
                            disks side 1, use $80 + sector #
FD05 VICCOUNT             ; number of sectors to read/write for disk i/o ;
                            number of characters to print on printer
                            or low byte of address to jump to for custom 8502
                            code.
FD06 VICDATA              ; various data/status items for disk and printer
                            or hi byte of address to jump to for custom 8502
                            code.
FD07 CURDRV               ; current logical drive using device A: (0 = drive A:
                                                                  4 = drive E:)
FD08 FAST                 ; drive type flags (updated after each drive i/o
                            operation), bit on=drive is fast (1571 or 1581)
                                        off=drive is slow (1541 type)

                            bit      drive
                            ------------------------
                            0        A:  (device 8)
                            1        B:        9
                            2        C:       10
                            3        D:       11
                            4-7      (not currently used)

FD09 KEYTBL               ; pointer to key scan definition table (1290)
FD0B FUNTBL               ; pointer to function key table (1000)
FD0D COLORTBLPTR          ; pointer to logical color table (13F0)
FD0F FUNOFFSET            ; offset to current function key (0 if none)

FD10 SOUND1               ; pointers for various clicks, beeps etc.
FD12 SOUND2
FD14 SOUND3

The following storage locations are used by the general BIOS and BDOS function
calls.  Values for BIOS 8502 storage (FD01 etc above) are taken from here:
FD16 @TRK                 ; track number used in BDOS and BIOS routines
FD18 @DMA                 ; address of disk data buffer
FD1A @SECT                ; sector number used in BDOS and BIOS routines
FD1C @CNT                 ; number of sectors to read/write
FD1D @CBNK                ; current bank  (0 or 1)
FD1E @DBNK                ; bank for disk data buffer (0 or 1)
FD1F @ADRV                ; absolute drive code (0 = A:, 1 = B:, etc)
FD20 @RDRV                ; relative drive code (same as for VICDRV
                            outlined above)

FD21 CCPCOUNT             ; # 128 byte records in CCP.COM file.

FD22 STATENABLE           ; general purpose status flags:

                            bit        meaning when set to 1
                            ------------------------------------
                            7          allow 8 bit key codes
                                       (key values above $80
                                       will not result in special
                                       functions defined in key
                                       scan table)
                            6          track cursor on 40 col screen
                                       when doing input on 80 col screen
                            5 to 1     (not used?)
                            0          display disk status on bottom line
                                       of screen.

FD23 EMULATIONADR         ; pointer to screen emulation code address (used to
                            switch between VT100 and ADM31 modes.  can be
                            pointed to custom code (in bank 0) for other
                            types of emulation.)

FD25 USARTADR             ; pointer to external 6551 address (not normally used)

FD27 INTHL                ; temporary storage for cpu registers during
FD3D INTSTACK             ; interupts, BDOS and BIOS calls, etc.
FD3D USERHLTEMP
FD3F HLTEMP
FD41 DETEMP
FD43 ATEMP

FD44 SOURCEBNK          ; source bank for inter-bank memory move  (0 or 1)
FD45 DESTBNK            ; destination bank for inter-bank move    (0 or 1)

FD46 MFMTBLPTR          ; pointer to MFM disk parameter table (DPT)

FD48 PRTCONV1           ; pointer to character conversion routine for PRT1
                          printer (normally ASCII to PETSCII)
FD4A PRTCONV2           ; pointer to character conversion routine for PRT2

FD4C KEYFXFUNCTION      ; pointer to dispatch table for executing extended
                          key codes functions F0 to FF.  This table can
                          be patched to add your memory resident special
                          functions.

FD4E XXDCONFIG          ; RS-232 configuration register
FD4F RS232STATUS        ; RS-232 status register

                            bit           meaning
                          ------------------------------------------
                            7           0 = ready to receive data
                            6           0 = idle   1 = busy
                            5           1 = data in buffer que
                            4           1 = parity error
                            3           1 = framing error
                            2           1 = other error
                            1           1 = receiving data
                            0           1 = ready to send data

FD50 XMITDATA           ; RS-232 sending data buffer
```

A CP/M Memory Map Continued

by Miklos Garamszeghy

```
FD73 RXDBUFCOUNT        ; number of characters in RS-232 buffer
FD74 RXDBUFPUT          ; temporary storage for RS-232 variables
FD75 RXDBUFGET
FD76 RXDBUFFER          ; RS-232 data receiving buffer (60 characters)

FDFD INTVECTOR          ; main entry point to interupt routine (jump to
                          routine elsewhere)

FE00 @BUFFER            ; 256 byte disk and general I/O buffer (you can put
                          your own code here if it is ok to overwrite when
                          not in use)

FF00 FORCEMAP           ; MMU configuration register
FF01 BANK0              ; force to MMU value of 3F
FF02 BANK1              ;                       7F
FF03 IO                 ;                       3E
FF03 IO0                ;                       3E
FF04 IO1                ;                       7E

FFD0 ENABLEZ80          ; 8502 code to switch to Z-80 mode
FFDC RETURNZ80

FFE0 ENABLE6502         ; Z-80 code to switch to 8502 mode
FFEE RETURN6502
```

****

BANK 1:

BANK 1 is the transient program bank. All transient programs operate in this bank. The MMU configuration value is $7F.

0000-0002 Jump to BIOS warm start entry BIOS.BASE + 3

0004 high nibble = current user number
low nibble = current default drive (0 to f, 0=A, etc.)

0005-0007 Jump to BDOS entry point (address stored in 0006-0007)
Note: this address also marks the end of the TPA.
It can be artificially lowered for use by resident programs.

0008-004f reserved for RST 1 to 7

(Note: Locations 0050 to 007b are set automatically by the CCP when a transient program is loaded. The transient program can then check this areas for parameters which may have been passed from the console in the form of a command tail.)

0050 drive from which latest transient program was loaded, 1=A, 2=B, ... 16=P

0051-0052 pointer to password of first operand in command tail (points to a location in the CCP buffer starting at 0080). It is set to 0 if no password specified.

0053 length of first password. Set to 0 if no password.

0054-0055 pointer to password of second operand in command tail. Set to 0 if no password specified.

0056 length of second password. Set to 0 if no password.

005c-007b default parsed file control block (FCB) area:
005c-006b initialized from first command tail operand
006c-007b initialized from second command tail operand

(Note: The FCB areas are normally 32 bytes long each. Thus, the second FCB area must be moved to an unused area before the first FCB area can be used or else it will be overwritten.)

007c current record position for default FCB 1.

007d-007f current random record position for default FCB 1.

0080-00ff default 128 byte disk buffer and CCP input buffer.

0100-e9ff 58 k transient program area (TPA) DEC and MAY versions
-edff 59 k TPA on AUG version

e000-ffff common with BANK 0 (top of TPA, also BDOS, BIOS, etc.)

****

Bank 2 : MMU value = $3E

This bank, which I have arbitrarily called BANK 2, is mostly the same as Bank 0 with the following exception:

1000 to 13FF VICCOLOR ; color map for 40 col screen

This bank must also be in context to access the MMU chip registers at D500 in the Z-80 I/O mapped area.

****

Z-80 I/O mapped area:

In addition to the normal memory mapping which includes RAM and ROM, the Z-80 has another addressing mode which is called I/O mapping. This is used to access the chip registers and is similar to memory mapping except the Z-80 IN and OUT instructions must be used instead of LD type instructions. The two most commonly used ones are:

IN A,(C) which will read the value of I/O addressed by .BC into the .A register

OUT (C),A which will write the value of .A to the I/O addressed by .BC

In both cases, .BC is a 16 bit address and .A is an 8 bit value.

The C-128 I/O chips appear at their normal address locations in the I/O area as outlined below and can be programmed directly by the experienced user. Note that you must be careful when playing with register values because CP/M expects most of them to be set in certain ways. Changing these settings may cause a system crash.

D000 VIC

D400 SID

D500 MMU

D600 8563 80 column chip

D800 VICCH VIC 40 col color map

DC00 CIA #1

DD00 CIA #2

DE00 external 6551 USART

```
DE00 TXD6551    (send and receive data register)
DE01 RESET6551  (write only)
DE01 STATUS6551 (when read)
DE02 COMMAND6551
DE03 CONTROL6551
```

DF00 RAM expander DMA controller chip

****

8502 mode

Surprisingly, the CP/M side of the C-128 also makes use of the 8502 side, including the KERNAL ROM's for some of the low level I/O operations. In 8502 mode, RAM from 0000 to 01FF is common between the banks. Note that this is different than normal C-128 mode which has common RAM from 0000 to 03FF. The most important implication of this that the KERNAL JSFAR, JMPFAR, INDFET, INDSTA, and INDCMP routines for bank manipulation cannot be used because they all rely on code which is no longer in common RAM (it is above 0200).

A CP/M Memory Map Continued

by Miklos Garamszeghy

Most of the memory map is similar to the Z-80 side with the exception of the low end. This low end area is very similar to that used in normal C-128 mode except as noted below.

```
Bank 0,   MMU value 3F

                ; BIOS 8502 working storage
000A            ; current printer device number
000B            ;         printer secondary address
000C            ;         disk drive data channel #
000D            ;         disk drive command channel #
000E            ;         disk drive device #
000F to 0012    ; temporary storage pointers

0013 to 008F    ; unused zp RAM where you can stash your own 8502 code
                ; (pointers normally used in C-128 mode, but not required
                ;  in CP/M mode)

0090 to 00CA    ; KERNAL pointers required for disk and printer I/O
                ; same as C-128 mode

00CC to 00FF    ; unused zp RAM

0100 to 0200    ; various KERNAL pointers, 8502 stack, etc.  some
                  unused locations but too chopped up to put code here.

0201 to 02FF    ; basically unused, you can put some code here if you wish

0300 to 0333    ; system vectors

0334 to 0361    ; unused?

0362 to 037F    ; logical file tables

0380 to 09FF    ; unused?

0A03            ; FF=50 Hz, PAL; 0=60 Hz, NTSC

0A1C            ; serial bus fast flag
```

Other areas up to 0FFF are used sporadically. Large open spaces in table buffer (0B00 to 0BFF) and C-128 mode RS-232 buffers (0C00 to 0DFF).

```
1000 and up     ; common with Z-80 side
```

The following area is used when the 8502 is switched in during CP/M operations:

```
2600-2a40   BIOS8502      ; 8502 BIOS code (all code here is in standard 8502
                                            machine language)
```

BIOS8502 function		Jump Table Entry		Points to code at	
		AUG	MAY/DEC	AUG	MAY/DEC
-1	reset	260f	2614	2625	262e
0	initialize	2611	2616	2654	265d
1	read 1541	2613	2618	2682	2690
2	write 1541	2615	261a	26b0	26be
3	read 1571/81	2617	261c	26f9	2707
4	wrt 1571/81	2619	261e	26f6	2704
5	inquire disk	261b	2620	270e	271c
6	query disk	261d	2622	2740	274e
7	printer	261f	2624	2776	2784
8	format dsk	2621	2626	27ac	27e2
9	user code	2623	2628	262b*	2634
a	1750 read	n/a	262a	n/a	2820
b	1750 write	n/a	262c	n/a	2823

* NOTE: there is an error in the code at 262b of the AUG version. The byte value is $c3 and it should be $6c for a JMP (xxxx). This means that 8502 BIOS subfunction 9 (jump to custom user 8502 code) does not work on the AUG version unless you first correct this bug!!

```
BIOS8502   IRQ, BRK and NMI vectors point to 29ae on the AUG version
                                          and 29f0 on the MAY and DEC ver.

Other I/O vectors are unchanged.

Bank 1 (MMU value 7F):
----------------------

0000 to 01FF        ; common with BANK 0

0200 to FFFF        ; common with Z-80 memory map in BANK 1
```

BASIC 8 PATTERNS

by Loren Lovhaug

Over the past five issues of Twin Cities 128 we have explored a variety of different *BASIC 8* programming techniques. With these techniques we have been able to do some pretty impressive things graphically. This issue's *BASIC 8* programming topic is patterns.

Unlike the C-128's built-in BASIC 7.0 paint facility, *BASIC 8* has a very powerful facility for creating and using fill patterns. With *BASIC 8*, fill patterns are more than simple solid or repetitive dot patterns of fixed sizes. *BASIC 8* fill patterns can be made up of any conceivable pixel combination with variable size and width and can include color data as well. These patterns can be as small as 8 x 1 pixels or as large as 2040 x 255 pixels, meaning that fill pattern data can be as small as a one byte line segment or big as a full screen image (and beyond). There are two ways in which BASIC 8 patterns can be created. The first method is by mechanically building a pattern structure data byte by byte. This technique is tedious at best and in my opinion not recommended, however if you want to learn more about this method, I defer you to the example on page 35 of the *BASIC 8* manual. The second, and in my opinion best way for creating pattern data is accomplished by converting *BASIC 8* picture scraps (called brushes) to pattern structures. You can create these brushes either by using BASICpaint or through the use of the wide variety of the plotting commands available in *BASIC 8*. The following program illustrates the technique for converting a brush to to pattern data and shows a variety of ways *BASIC 8* patterns can be utilized.

```
97 REM ** FUN WITH BASIC 8 FILL PATTERNS **
98 REM ** BY LOREN LOVHAUG - JULY 1988   **
100 FAST
110 DEF FNR(X)=INT(RND(1)*X)+1
120 @WALRUS,0
130 @MODE,0
140 @SCREEN,2:@COLOR,2,13,0:@CLEAR,0
147 REM ** CREATE SCREEN DATA **
150 @CIRCLE,20,20,0,18,1
160 @CIRCLE,15,15,0,5,1
170 @CIRCLE,25,15,0,5,1
180 @ARC,20,20,0,10,10,220,320,8,1,0
190 @CHAR,254,5,0,2,1,2,"TWIN CITIES 128"
200 @CHAR,254,6,16,1,1,2,"THE BEST C-128"
210 @CHAR,254,5,24,1,1,2,"COVERAGE AROUND"
220 @LINE,40,34,0,160,34,0,1
230 @LINE,40,36,0,160,36,0,1
236 REM ** PUT SCREEN DATA INTO STRUCTURE **
237 REM ** AND CONVERT TO PATTERN DATA **
240 @BUFFER,0,48896,16384
250 @STASH,0,0,0,0,0,160,40,0
260 AD=@STASH,0,0,0,0,0,160,40,0
270 @BRUSHPATRN,0,1,0,AD
273 @DRWMODA,0,0,0,0,1,0,0
275 @PATTERN,1
276 REM ** DEMO 1 - FILL **
280 @COLOR,0,13,0:@CLEAR,0
290 @ARC,320,88,0,120,60,0,360,8,1,0
300 @ARC,320,88,0,180,80,0,360,8,1,0
330 @PAINT,321,10,0,31000,16384
336 REM ** DEMO 2 - RADAR REVEAL **
340 @COLOR,0,4,0:@CLEAR,0
350 FOR I=79 TO 0 STEP-1:@LINE,160,79,0,0,I,0,1:NEXT I
360 FOR I=0 TO 320:@LINE,160,79,0,I,0,0,1:NEXT I
370 FOR I=0 TO 79:@LINE,160,79,0,320,I,0,1:NEXT I
376 REM ** DEMO 3 - PIXELIZE **
380 @COLOR,0,15,0:@CLEAR,0
390 FOR I=1 TO 10000
395 @DOT,FNR(160),FNR(39),0
397 IF I/500=INT(I/500) THEN @CHAR,254,25,24,1,1,2,STR$(I)
400 NEXT
410 SLEEP 5:@TEXT
```

The first part of this program, lines 100 - 230, simply sets up the and program draws a small picture (160 x 40 pixels) in the upper left portion of the screen. Then in line 240 I define a 16K area of storage in bank 0 using the @buffer command (page 69 - *BASIC 8* manual). Note that bank 0 is also where your BASIC code is stored so ordinarily you should adjust the start of BASIC text pointer (locations 45 and 46) or the end of BASIC text pointer (locations 4624 and 4625) to insure that your program does not conflict with your storage area. In this case I knew that this demo program was going to be rather short so I did not bother to do so, but keep in mind it is a good idea to do so. In lines 250 and 260 I copy my picture as a brush structure using the @stash command (page 110 - *BASIC 8* manual) into the graphics buffer and find out where the next byte of free space is in my graphics buffer. This location is stored in the variable AD and is needed so that we can tell *BASIC 8* where to store our pattern data when we convert the brush structure data in line 270. The @brushpatrn (page 69 - *BASIC 8* manual) is quite straightforward and automatically converts your brush data to a pattern structure.

Once we have actually created the pattern structure from our picture, we must tell *BASIC 8* that we want to use a user defined fill pattern and which pattern we want to use. To tell *BASIC 8* that we want to use our own fill pattern as opposed to the solid default pattern we must utilize the @drwmoda command (page 81 - *BASIC 8* manual). The fifth parameter of this command controls whether or not a user defined fill pattern is being used. Setting this parameter to 1 will tell *BASIC 8* to use a user defined fill pattern, see line 273. The @pattern command (page 95 - *BASIC 8* manual) found in line 275 lets *BASIC 8* know which structure is going to be used as our user defined fill pattern.

My first demo (lines 280 - 330) simply draws two circles using the @arc command (page 66 - *BASIC 8* manual) and then fills the area between those circles with our newly created fill pattern using the @paint command (page 94 - *BASIC 8* manual). Keep in mind when using the @paint command, you should be sure to specify an area in memory that the command can use a "stack area" so that the paint command can function correctly when filling around complex objects such as curves. As above, you might have to adjust the BASIC text pointers to insure this stack area does not conflict with your BASIC program if you place the stack in Bank 0.

My second demo (lines 340 - 370) utilizes a little known feature of *BASIC 8*. These feature involves the fact that when the user defined fill pattern flag in the @drwmoda command is set to 1, all commands which draw on the screen, do so using the currently defined fill pattern. This means that *BASIC 8* drawing commands such as @line and @dot do not necessarily render solid lines or points on screen when a user defined fill pattern is being utilized. Instead these commands "turn on" only those pixels that are on within the fill pattern. This feature can be used to produce a variety of effects, including my "radar reveal" which uses the @line command (page 87 - *BASIC 8* manual) in three for-next loops to uniquely display our original picture four times.

My final demo, (lines 380 - 400) uses this same technique in a different way to "pixelize" our original picture, by plotting ten thousand randomly selected dots within the original 160 x 40 area. As you watch the dots being plotted a counter at the right of the picture will tell you how many dots have been plotted (updated every 500 dot plots). Note that only those dots that are part of the picture are being turned "on". Another thing to keep in mind as this demo runs that it is likely many dots are being "plotted" more than once.

Well that's all for this issue, I hope that this column will inspire you to utilize *BASIC 8's* pattern capabilities within your programs.

WILY WINDOW WORK

by Loren Lovhaug

Most people write off BASIC 7.0's window command as the C-128's "command that could have been". After all, BASIC 7.0's windowing facility as sophisticated as those on the Macintosh, Amiga, and Atari ST. But as usual, the C-128 once again proved that it has a great deal of power and sophistication under its hood.

Although the C-128 is not blessed with built-in non destructive windowing, (it can be added, see John Kress' article in Twin Cities 128 Issue #15 and on pages 128-130 of our compendium book #1) this does not diminish the C-128's windowing feature, especially as a text formatting and display tool. To illustrate this point I wrote the following BASIC 7.0 program I call "scroll toy". Scroll toy's mission is quite simple, all it does is allow you to scroll through some information using the up and down cursor keys. But I think the way in which is does so is pretty elegant as useful for programmer's looking to add that special touch of class to your masterpiece.

Scroll toy relies on the window scrolling feature Fred Bowen built into the C-128's screen editor. Ordinarily people become aware of this feature when they experiment with C-128's ESC sequences. ESC v and ESC w will scroll the currently defined window up or down respectively. When you want to use this feature in your programs, you can access the scrolling feature by actually printing the ESC sequence (i.e. print chr$(27);"v") or calling the screen editor routine which accomplishes the task directly. In my example below I chose to do the latter, but since Fred always says, "Don't mess with my code, unless you know what you are doing, and even then...", please don't tell him that I showed you how to call these routines directly!

Another important concept scroll toy relies on is the C-128's ability to "clear any window definition". Ordinarily this is accomplished by pressing the home key twice or printing CHR$(19) twice consecutively, resulting in the window returning to its default (full) size. This concept is important because the window command always works relative to the currently defined window. This makes consecutive window definitions tedious because you always have concern yourself with what your new definition must be relative to the previous one. By returning to the default window, things become much simpler.

One last note, when doing a lot of tricky formatting with the multiple window definitions is often helpful to turn the C-128's automatic screen scroll feature off. This will prevent your data from automatically scrolling upwards when you print data on the last line of the currently defined window. Printing ESC m or setting bit 7 of location 248 will disable auto-scrolling. Printing ESC l disables or turning off bit 7 of location 248 disables it. Enjoy!

```
 98 REM *** WILY WINDOW WORK SCROLL TOY *** JULY 1988
 99 REM *** BY LOREN J. LOVHAUG *** FOR ISSUE #21 OF TWIN CITIES 128
100 CW=51748:REM                                          * SYS CW: CLEAR CURRENT WINDOW DEFINITION
110 UP=51900:REM                                          * SYS UP: SCROLL TEXT UP WITHIN WINDOW
120 DN=51914:REM                                          * SYS DN: SCROLL TEXT DOWN WITHIN WINDOW
130 :
140 L$=CHR$(192):S$=CHR$(32):REM                          * LINES 140-180 DEFINE THE STRING VARIABLES
150 FOR I=1 TO 7:L$=L$+L$:S$=S$+S$:NEXT:REM               * THAT ARE USED WHEN THE BOX IS PRINTED.
160 T$=CHR$(176)+LEFT$(L$,78)+CHR$(174):B$=T$:REM         * L$: HORIZONTAL BAR T$: TOP OF BOX
170 M$=CHR$(221)+LEFT$(S$,78)+CHR$(221):REM               * B$: BOTTOM OF BOX  M$: SIDES   S$: SPACES
180 MID$(B$,1,1)=CHR$(173):MID$(B$,80,1)=CHR$(189):REM
190 CH$=CHR$(145)+CHR$(17)+CHR$(13):REM                   * CH$: LEGAL KEYPRESSES UP/DOWN/RETURN
200 :
210 FOR I= 1 TO 10:READ N$(I),A$(I),C$(I),S$(I),Z$(I):NEXT I:READ FM$:REM   * READ DATA AND PRINT USING FORMAT TEMPLATE
220 :
230 SCNCLR:POKE 248,128:WINDOW 0,1,79,24:COLOR 5,6:REM    * SCNCLR, TURN SCROLL OFF, SET WINDOW/COLOR
240 PRINT"PRESS THE UP/DOWN ARROWS TO SCROLL.  RETURN ENDS."
250 COLOR 5,3:PRINT T$:FOR I=1 TO 3:PRINT M$:NEXT I:REM   * CHANGE COLOR, DISPLAY BOX
260 PRINT B$:COLOR 5,6:SYS CW:T=1:B=3:REM                 * T: TOP ITEM IN BOX  B: BOTTOM ITEM IN BOX
270 WINDOW 2,3,77,6,0:FOR I=1 TO 3:GOSUB 340:NEXT I:SYS CW:REM   * DISPLAY INITIAL DATA (ITEMS 1 - 3)
280 :
290 DO:GETKEY A$:C=INSTR(CH$,A$):IF C=3 THEN EXIT:REM     * LINES 290-310 FORM THE PROGRAMS MAIN LOOP
300 ON C GOSUB 360,400:REM                                * KEEP FETCHING KEY PRESSES UNTIL RETURN IS
310 LOOP:REM                                              * PRESSED; IF THE KEY IS UP - UP SUBROUTINE
320 POKE 248,0:SYS CW:SCNCLR:END:REM                      * IF KEY IS DOWN - DOWN SUBROUTINE
330 :
340 PRINT USING FM$;N$(I),A$(I),C$(I),S$(I),Z$(I):RETURN:REM   * PRINT FORMATTED DATA SUBROUTINE
350 :
360 IF T=1 THEN SOUND 1,500,4:WINDOW 70,1,79,1,0:PRINT "TOP":RETURN:ELSE T=T-1:B=B-1:WINDOW 70,7,79,7,1
370 SYS CW:REM                                            * LINES 360-380 FORM THE UP SUBROUTINE
380 WINDOW 2,3,77,5,0:SYS DN:SYS CW:WINDOW 2,3,77,3,1:I=T:GOSUB 340:SYS CW:RETURN
390 :
400 IF B=10 THEN SOUND 1,500,4:WINDOW 70,7,79,7,0:PRINT"BOTTOM":RETURN:ELSE T=T+1:B=B+1:WINDOW 70,1,79,1,1
410 SYS CW:REM                                            * LINES 400-420 FORM THE DOWN SUBROUTINE
420 WINDOW 2,3,77,5,0:SYS UP:SYS CW:WINDOW 2,5,77,5,1:I=B:GOSUB 340:SYS CW:RETURN
430 :
440 DATA "COMMODORE","1200 WILSON DRIVE","WEST CHESTER","PA","19380":REM    * PROGRAM DATA AND PRINT USING FORMAT
450 DATA "QUANTUM LINK","8620 WESTWOOD CENTER DRIVE","VIENNA","VA","22180"
460 DATA "RUN MAGAZINE","80 ELM STREET","PETERBOROUGH","NH","03458"
470 DATA "BERKELEY SOFTWORKS","2150 SHATTUCK AVE.","BERKELEY","CA","94704"
480 DATA "TWIN CITIES 128","P.O. BOX 4625","ST. PAUL","MN","55104"
490 DATA "FREE SPIRIT SOFTWARE","905 W. HILLGROVE #6","LA GRANGE","IL","60525"
500 DATA "XETEC","2804 ARNOLD ROAD","SALINA","KS","67401"
510 DATA "ABACUS SOFTWARE","P.O. BOX 7219","GRAND RAPIDS","MI","49510"
520 DATA "S.O.G.W.A.P. SOFTWARE","115 BELMONT ROAD","DECATUR","IL","46733"
530 DATA "BUSY BEE SOFTWARE","P.O. BOX 2959","LOMPOC","CA","93438"
540 DATA "##########################  ####################  ##############  ##  #####"
```

CHECK US OUT! LET US EARN YOUR BUSINESS!!

☑ SERVICE ☑ SUPPORT ☑ SELECTION ☑ PRICES ☑ GUARANTEE

HOME/BUSINESS

Bankstreet Writer		$35
BEST Business Series 64	EA	45
Business Form Shop		30
CMS Accountant 128		129
CMS Inventory 128		53
Chartpak 128 OR 64	EA	29
Datamanager 128		39
Datamanager 2		17
Fleet Filer 64/128		29
Fleet System 2 +		39
Fleet System 4		52
Fontmaster 128 w/Spell		39
Fontmaster II		35
KFS Accountant 128		119
Add-ons - call us		52
Math Function Evaluator		27
Microlawyer 64		42
PPM 128		42
PPM 64		29
Page Builder 128		39
Paperclip Publisher 64		39
Paperclip 3		39
Partner 128		39
Partner 64		30
Pocket Dictionary		10
Pocket Superpak 2		67
Pocket Write/File or Plan 2	EA	39
Securities Analyist 128		35
Superbase 128		49
Superbase 64		44
Superbase 128/1581 drive		**call
Swiftcalc 128 w/Sideways		39
Swiftcalc 64		17
TAS 128		42
TAS 64		29
Timeworks Business Series	EA	39
Vizistar 128		60
Viziwrite Classic 128		60
Wordwriter 128		39
Wordwriter 3		30

EDUCATIONAL

Body Transparent		22
Early Learning Friends/PRE		6
Facemaker/PRE		18
Galactic Frontier		22
Kindercomp/PRE		18
Stickybear ABC's/PRE		22
Stickybear Numbers/PRE		22
Stickybear Shapes/PRE		22
Alphabet Zoo/5-8		18

EDUCATIONAL

Easy Sign/5 +		$17
Grandma's House/5-9		6
Kids on Keys/5-8		18
Linkword Foreign Languages	EA	19
Little Computer People		$1
Peter Rabbit (Reading 1) / 5-8		19
Reader Rabbit/4-7		29
Rocky's Boots/9 +		29
Stickybear Math 1/5-8		22
Stickybear Reading/5-8		22
Stickybear Typing/5-8		22
Jungle Book (Reading 2) / 6-9		19
Kidwriter/6-9		22
Mathbusters/6-9		9
Stickybear Math 2/6-9		22
Stickybear Spellgrabber/6-9		22
Stickybear Townbuilder/6-9		22
First Men on Moon Math/9 +		19
Stickybear Read Compreh./9 +		22
Evelyn Wood Dynamic Read/11 +		17
Pro Tutor Accounting/14 +		69
R.S.V.P. / 12 +		21
Ticket to London, Paris, Span OR Washington/12 +	EA	21
Where USA Carmen		35
Where World Carmen		35
Widham Clasics / 10 + Alice In Wonderland, Swiss Family Robinson, Below The Root, Treasure Island OR Wizard of OZ	EA	12

PERSONAL

Bodylog Cardio Exercise		75
Bodylog Enhanced Stress		229
Bodylog Heartlab		49
Bodylog Muscle Development		54
Bodylog Standard Stress		89
Boston Bartender's Guide		12
Bridgemaster		12
Build a Book About You		19
CDA Cooking Titles		call
Crossword		12
Dr. Ruth's Sex		22
Dream Machine		19
Family Tree 128 OR 64	EA	39
Paul Whitehead Chess		25
RSVP		21
Sexual Edge		19
Tempo Typing		12

GEOS

Becker Basic		$35
Fontpac Plus 64/128		35
GEOS 128		45
GEOS 64		39
GEOS Companion Number One		24
GeoFile 64		35
GeoProgrammer		45
GeoPublish 64		45
GeoSpell		19
Geocalc 128		45
Geocalc 64		35
Geofile 128		45
Geowrite Workshop 128		45
Geowrite Workshop 64		35
Neofont		19
Word Publisher (GEOS)		28

GRAPHICS/CAD

3D Perspective 2		39
Advanced OCP Art Studio		25
Animation Station		65
BASIC 3		25
Billboard Maker		29
Blazing Paddles		25
Bumper Sticker Maker		35
Business Card Maker		35
CAD 3D		39
Cadpak 128		42
Cadpak 64		29
Certificate Maker		17
Colorez 128		12
Doodle!		29
Flexidraw 5.5		29
Geopublish 64		45
Graphic Art Disks 1 thru 10	EA	13
Graphic Label Wizard		21
Graphics Galleria Pak 1 OR 2	EA	24
Graphics Integrator 2		24
Graphics Transformer		25
Home Designer 128		45
ICON Factory		25
Moving Pictures		20
Newsroom		17
PCB Circuit Board Maker		79
Page Illustrator 128		29
Paperclip Publisher 64		39
Photo Finish		19
PrintMaster Plus		25
Printshop		35
Screen F/X		25
Slideshow Creator		13
The Toy Shop		22
Video Title Shop		25

AIDS/UTILITIES

1541/1571 Drive Alignment		$25
BASIC 8		30
BASIC Compiler 128		42
BASIC Compiler 64		29
Big Blue Reader 64/128		32
Bobsterm Pro 128		42
Bobsterm Pro 64		35
CPM Kit 128		22
COBOL 128 OR COBOL 64	EA	29
Final Cartridge 3		49
Gnome Kit 64/128		29
Mach 128		35
Merlin 128		45
Merlin 64		35
Physical Exam 1541 OR 1571		29
Power Assembler 64/128		30
Power C 64/128		30
Proto Linc BBS 128		45
RAM DOS 128		29
Renegade		29
Super 81 Utilities 128 or 64		29
Super Aide		25
Super C 128 OR 64		45
Super Chips 64		25
Super Chips 128 or 128D		45
Super Disk Utilities		25
Super Librarian 128		19
Super Pascal 128 OR 64	EA	45
Syntech BBS Construction Set		43
Syntech BBS Games Module		18
Sysres Enhanced		30
Warp Speed 128		35
Kracker Jax Volumes 1-4	EA	10
Kracker Jax Volumes 5-7	EA	17
Loaded Gun 2		17
Super Snapshot V3		49
Super Snapshot V3 w/C128 Disk		54

BOOKS

Abacus Books		call
Big Tip Book for Commodore		15
C128 Programmer Ref. Guide		22
C128 Subroutine Library		15
GEOS Prog Ref Guide 2		17
K Jax Revealed Book 1		23
K Jax Revealed Book 2		23
Superbase the Book		15
Troubleshoot & Repair C64		18

MISCELLANEOUS

2-for-1 Serial Box		29
4-for-1 Serial Box		39

MISCELLANEOUS

C64 OR 1541 Cover	EA	7
C128 OR 1571 Cover	EA	9
40/80 Column Switch Cable		18
AB Printer Switch Box		35
Aprospan Cartridge Holder		$27
Drive Box		27
Hotshot Plus Printer Interf.		69
Lightpen 170C		69
Lightpen 184C		49
M3 Proportional Mouse		40
Mousepad		8
Partner 128		39
Partner 64		29
Print & Wear Transfer Paper		17
Quick Brown Box - 32K		79
Quick Brown Box - 64K		99
RGB to TV Cable		9
RS232 Interface Cable		39
Printer Paper		**call
Printer Ribbons		**call
Serial Cable - 6 Foot		9
Super Graphix Gold		89

BRiWALL

Solid Products • Solid Support

P.O. Box 129, 56 Noble St.
Kutztown, PA 19530

TOLL-FREE 24 HOURS
1-800-638-5757

OUR PROMISE

WE GUARANTEE YOUR SATISFACTION

If, for any reason, you are not satisfied with your selection within 15 days of your receipt, simply return the product to us. We will either issue you full credit for exchange on another selection, or refund your purchase price, less $5.00 for restocking and handling. Defective items are replaced free of charge!

Great Products By Free Spirit Software Inc.

1541/1571 Drive Alignment

1541/1571 Drive Alignment reports the alignment condition of the disk drive as you perform adjustments. On screen help is available while the program is running. Includes features for speed adjustment and stop adjustment. Complete instruction manual on aligning both 1541 and 1571 drives. Even includes instructions on how to load alignment program when nothing else will load! Works on the C64, SX64, C128 in either 64 or 128 mode, 1541, 1571 in either 1541 or 1571 mode! Autoboots to all modes. Second drive fully supported. Program disk, calibration disk and instruction manual. only **$25.00**

Super 81 Utilities now available for the C64!

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

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

SUPER 81 UTILITIES

Super 81 Utilities uses an option window to display all choices available at any given time. A full featured disk utilities system for the 1581. only **$29.00**

RAMDOS RAMDOS is a complete RAM based "Disk" Operating System for the Commodore 1700 and 1750 RAM expansion modules which turns all or part of the expansion memory into a lighting fast RAM-DISK. RAMDOS behaves similar to a much faster 1541 or 1571 floppy disk except that the data is held in expansion RAM and not on disk. Under RAMDOS, a 50K program can be loaded in ½ second. Programs and files can be transferred to and from disk with a single command. RAMDOS is available for only **$29.00**

PROFESSIONAL CAD FOR YOUR COMMODORE 128! HOME DESIGNER

This is the most powerful two-dimensional CAD program available for your C128! As an object-based package (this is what the professionals use) your drawings are created and printed with the **greatest accuracy and detail possible.** This object-based design provides extreme flexibility and control over your drawings because each line, circle, text-string, etc. can be manipulated (rotated, enlarged, erased) as a separate entity. **With over 50 powerful commands,** you can translate, rotate and mirror objects. Stretch or shorten lines as needed. Easly create parallel lines. Draw or erase arcs, circles, ellipses and boxes at will. Insert library figures anywhere into drawings at any scale and angle. Scale text to any height and width. Work in either an absolute scale or feet/inches. Your drawing screen (a 16" by 22" page) includes **5 drawing layers** that can be selected/included/excluded/erased at will. HOME DESIGNER provides **accurate, lazer-quality output, at ANY scale, on your dot matrix printer or plotter.** Use a joystick or the 1351 mouse as input.

WHEN THE DETAILS ARE IMPORTANT! **Still only $45.00**

THE WRITE STUFF WRITER & TALKER

This WORDPROCESSOR is by far one of the BEST that we have seen! Let's get right to the specs: 21 function calculator - line up numbers with the decimal tab; 60 customizable help screens - 86K of disk doc's; Double-column printing in one pass; Encyrpt/decrypt text; Sort up to ten columns (great for lists); Merge/Append text files; Link files; Word/paragraph counter; File translator for 15 other W/P's; Dvorak [- -] QWERTY toggle; True 80 column Hi-Res preview; Key-click for typewriter sound; Mail Merge; Automatic Case; On-line clock; 1764 RAM Expander support; MACRO TYPING cuts your typing time by over 25% and more!

The WRITE STUFF even comes in a TALKING version, which includes SAM, a program that produces speech from your computer's sound capabilities. This BB TALKER recites any portion of your text, can say each letter and command as it is typed, and translates to phonetic script for additional editing. **Great for the sight-impaired and the youngsters!!**

C64 version - $19.00 **C128 version - $24.00** **C64 w/Talker - $24.00**

FREE CATALOG

40 pages of detailed descriptions for over 300 fine products for your Commodore computer

CALL OR WRITE FOR YOUR COPY TODAY!

$$$$$$$$

COLLECT BONUS DOLLARS

One bonus dollar, good towards your next purchase, for every $50.00 ordered!

COLLECT THRU 8/31/88

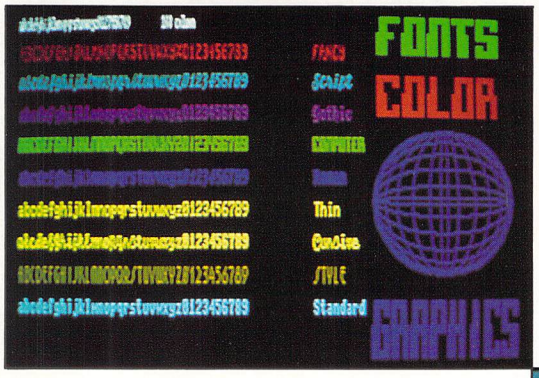
USE THRU 12/31/88

CALL FOR LATEST PRODUCTS, PRICES AND COMBINATION SPECIALS!

ORDERING INSTRUCTIONS: VISA, MasterCard, Personal Check, Money Order or COD are welcome. Shipping USA: $3.50/ground, $5.50/2nd Day, $14.00/overnight. C.O.D. add $3.00. Canada/Mexico: $6.00 Air Mail. Foreign: $12.00 (for Air Mail use charge & specify maximum). PA residents add 6% sales tax. Include phone number and computer/drive model with order. All items will be shipped immediately, (if we are out-of-stock, we will let you know.) **CHARGE IS PER ORDER.**

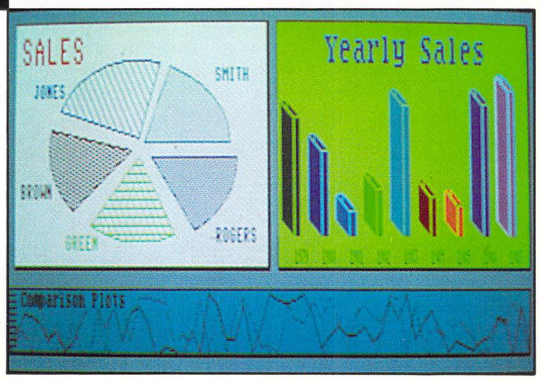
For CUSTOMER SERVICE please call us:
Monday thru Friday, 9:30 a.m. - 4 p.m. EST
1 - (215) - 683-5699

VISA MasterCard NO SURCHARGE



Introducing BASIC 8

By Lou Wallace & David Darus



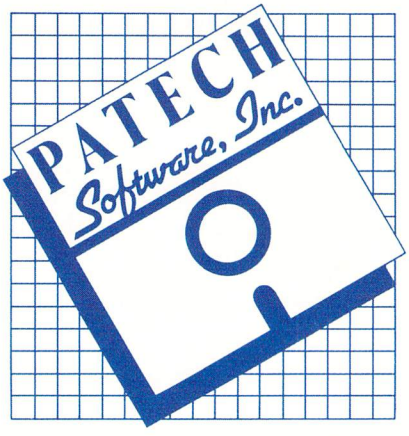
At last, you can unleash the graphics potential of your Commodore 128 to achieve performance which rivals that of 16-bit micros! Imagine your 128 (or 128-D) producing resolution of 640 x 200 in monochrome and 640 x 192 in 16 colors without any additional hardware. Sound impossible? Not with **Basic 8**, the new graphics language extension.

Basic 8 adds over 50 new graphics commands to standard C-128 Basic. Just select one of many graphics modes and draw 3-D lines, boxes, circles and a multitude of solid shapes with a single command. We've even added commands for windows, fonts, patterns and brushes.

To demonstrate the power and versatility of this new graphics language, we have created **Basic Paint**, a flexible icon-based drawing application. Written in **Basic 8**, **Basic Paint** supports an expanded Video RAM (64K), RAM Expanders, Joystick and the New 1351 Proportional Mouse.

Also included is an icon-based desk-top utility which provides quick and convenient access to each of your very own **Basic 8** creations.

All this graphics potential is yours at the special introductory price of $39.95. The package includes **Basic 8**, **Basic Paint**, the desk-top utility, a 180-page manual and a run time module. (80-Column RGB Monitor Required)



Complete Package

$39.95

*Details inside package

TWO CONSTRUCTIVE PROGRAMS
From Patech Software, Inc.

You don't need a new computer to join the desktop publishing revolution! With ***PAGE BUILDER*** and ***PAGE ILLUSTRATOR*** from PATECH Software, your C-128 (or C-128D) can compose professional-looking pages, using your own dot-matrix printer!

PAGE BUILDER and ***PAGE ILLUSTRATOR*** are stand-alone programs; each with a specific function to perform. Whether used singly or as an unbeatable team, these Two Constructive Ideas are designed to make maximum use of the C-128's native power. They were created by publishing professionals who put that power where you need it most — into real performance and utility, not bells and whistles.

PAGE BUILDER *Suggested Retail Price* $49.95

PAGE ILLUSTRATOR *Suggested Retail Price* $39.95

Dealer & distributor inquiries
201-545-1571

For ordering and information
201-238-5959

PATECH Software, Inc.

VISA MasterCard

Add $3.50 for Shipping and Handling