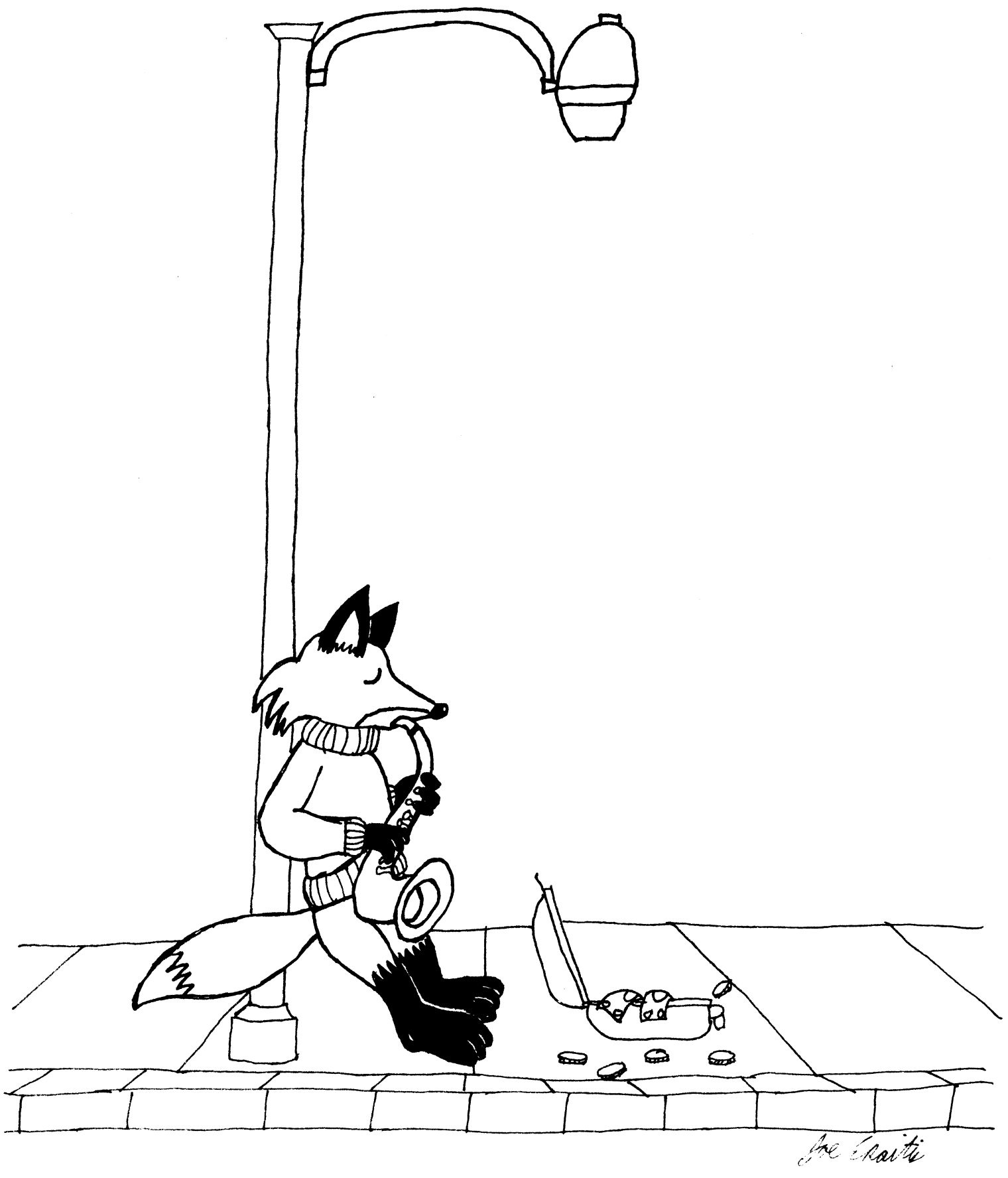
TC-128/64



Issue #34 - August 1993
$4.95 magazine - $9.95 magazine and disk

LEGAL NOTICE

Twin Cities 128/64, the magazine, or companion disk may NOT be copied in whole or in part for ANY reason. TC-128/64's companion disk is commercial software and is only for individual use. It may NOT be put into a User Group s "commercial disk collection" where it can be copied freely, loaned or rented. If it is, the group and all members will be held liable for all attorney fees and damages. Owning a copy of the magazine does NOT entitle you to a copy of the companion disk unless you receive it through your paid subscription.

SOFTWARE NOTICE RIGHT TO USE

The software, hardware, and routines published in this magazine can be used free of charge only if ALL of the following conditions are meet:

1)The program is copyrighted but freely distributable and you were a were a subscriber when the issue was published.

2)You have to give a written notice on your first screen or title screen, where this type of phrase can be clearly noted by the user (an example) "Sound Routines from Twin Cities 128/64 - issue #32"

3)You have to send us a copy of the program on a disk or upload the program to our library on GEnie. Do not send it by e-mail!

4)If there are any kind of charges for the program, either as commercial or shareware software, or if it is a "demo" for a company contact us FIRST before releasing the software/hardware so we can talk about the licensing fee. This usually will be something small, such as copy of the finished product. If we find out after the fact it will cost you *MUCH* more. Only written releases from Parsec through the U.S. mail with our company stamp imprinted on the contract will be considered valid.

5)These routines may not be uploaded to any network.

6)These routines may NOT put into ANY disk library collection - individual use only - no exceptions!

SUBMISSIONS:

Submissions greedily accepted. Average author rate is $25-$100 plus a free six issue extension with the disk (code 4). We gladly consider any program, article, software, or hardware for publication. If you have an idea, write to us first to see if we can use it and so we can offer suggestions on what we need.

REVIEWS

Reviews of software and hardware are done by established staff members that have already had articles published and are GEnie members

ADVERTISEMENT RATES

Full page with (2) colors $200 (white paper, black ink, and one other color)
1/2 page with (2) colors $100 (white paper, black ink, and one other color)
1/8 page B&W $60
CLASS(Y) ADS - Please see the classified section.

Payment must accompany the ad. It is best to write or leave e-mail on GEnie to guarantee a spot in the issue. We can typeset ads on our laser equipment. Any ads typeset become copyrighted by Parsec and may not be republished without a release. The typesetting charge depends on the amount of work needed.

For sellers of hardware, a copy of the product being advertised must be submitted for review. If it is expensive, it will be returned. For sellers of software two copies of the product (one for the editor and one for the reviewer) must be submitted for review. They will not be returned. Payments must accompany the ads.

DISTRIBUTORS

Twin Cities 128/64 is sold only through Parsec, Inc. Twin Cities 128/64 is distributed by Parsec, Inc. RIO computers, Matt Mattings (Germany), and Sandinge's Import and Data (Sweden).

THE COMMODORE 128 & 64 COMPUTER JOURNAL TC-128/64 - ISSUE #34 AUGUST 1993

Publisher: Parsec, Inc.
Editor : John W. Brown

ADVERTISERS:

Published, distributed, and copyrighted 1993 with all rights reserved by Parsec, Inc. PO Box 111 Salem, MA 01970-0111 USA

ADDRESS CORRECTIONS SHOULD BE SENT TO THE ABOVE ADDRESS.

Parsec is not responsible for any advertiser's claims, typos, and misprints. Ads typeset with errors by Parsec will be rerun at Parsec s option and/or a correction will be run in one of the following issues. If readers have unsolved problems with an advertiser they should contact Parsec. The only way to contact us is on GEnie @ C128.JBEE or by US mail. Phone calls will not be returned. Twin Cities 128/64 is produced completely on a C-128D system with an Epson Action Laser II printer using Paperclip III and GEOS.

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

EDITORIAL LICENSE

BY JOHN W. BROWN

C-128.JBEE on GEnie

Well, it is that time of year again, summer time. This column at this time of the year has been frequently used to remind people to "get a life" and to enjoy summertime. To be with the wife (spouse or S.O.) and kids, and to enjoy yourself out in the summer air. To take a break from the computer. So in the middle of putting this issue together I decided to listen to that advice and went to see a show at the local Performing Arts Theatre with my wife. We were not disappointed!

We saw Jim Nabors in concert which is something I have always wanted to do! Though known mostly for the TV show Gomer Pyle, he is also known the world over for his deep rich singing voice and a wide range of entertainment talents. Though the story telling was a bit corny, it fit with the show and story clips on the projector screens. The tickets were a bit over priced and it would have been nice to meet him after the show. But, I still felt the tickets were well worth it and I would recommend seeing the show if you had a chance.

Which brings me to

"SURPRISE, SURPRISE, SURPRISE"
We changed the format of the magazine and it was not easy! This tabloid page is roughly 50% larger than the previous sized page and has made layout of program listings much easier. The older page size had a printed page coverage of roughly 4704 inches (08x10.5x56 pages) Our new magazine will have roughly 7280 inches (10x13x56 pages) which is a 50%+ increase in page coverage!!! This will be at no additional cost or increase in subscription fees now or in the future. If everything works out we may even up the page count dramatically. We will be increasing the page count incrementally. There will also be much more color and a wider variety of articles now that we have the room. Look for a lot more reviews in the next issue!

We finally found suitable packaging that is cheaper than the plastic bags we have used in the past. We used PVC plastic bags because cheaper packaging was not available. I had tried in vain to buy the biodegradable plastic bags made from corn starch but at $0.12+ a piece in large quantities it just was not feasible when plastic bags cost four times.

Most of the free issues of #33 were shipped in specially purchased paper bags that we were trying out before committing to this new format and packaging. I believe DuPont, of all companies, makes these new paper bags. Besides being biodegradable and saving plastic trees :) this new packaging means we no longer have to have a "mailing page" and can recover that page for expanded editorial use. We used it for the new C-64 checksum program on page 56 Mike Gilsdorf rushed to finish for me. Thank you Mike.

Companion Disk issues and Overseas issues will still use the plastic bags or special paper envelopes because we have to provide those issues with more than adequate protection that is not needed for the bulk issues mailed in the US.

I figured this move cuts down on the use of at LEAST 1500 lbs of plastic bags that we use a year, which are not biodegradable, and are thrown away after the magazine is opened. Unless some of you have been using them to store your frozen vegetables :-) As some of you probably have noticed, I like to reuse floppy mailers, cardboard, obsolete W-2 envelopes or anything else other companies and people throw away or dump as trash. I am sure it is a relatively small amount in the scheme of things but eliminating this plastic bag usage was something I have wanted to do for the longest time. Speaking of paper problems ....

ITS TOO BIG and TOO MUCH!
My local dealer wanted $100 for a fourteen inch paper tray, being a frugal Commodore owner, I was not about to pay that much for a paper tray! After all, how much does it cost to produce and market a relatively low tech piece of plastic and metal like a paper tray.

Luckily, as I suspected, fourteen inch paper works fine in a 11 inch tray, as long as you do not mind the paper flopping over the end of the tray. For $100 I do not mind that much. What I really need is a 11x17 duplex Hewlett Packard laser printer that costs more than my van and car put together. The piecing together of the pages, since the printed image is wider than 8.5 images, drove me crazy. Six rolls of tape for this issue. I hope the camera shots look acceptable. I honestly do not know how good it will look spliced and diced with all this tape. Live brave and pray a lot!

Well, I hope this issue finds you in good health (even our critics) and that you enjoy the summertime and warm weather.

JBEE
(:

IN THE NEWS

NEW GENIE RATES
(Editor's note:this press release was edited to fit the page space available) May 24, 1993 -- GEnie today announced a new pricing stucture, for the U.S. and Canada, that reduces hourly connect rates by 50%. Effective July 1, GEnie s standard hourly connect rate drops to $3.00 per hour ($4.00 Canadian) The monthly fee has been restructured, and moves from $4.95 to $8.95 (from $5.95 to $10.95 in Canada), which will include up to four hours of standard connect time access to most GEnie services, such as software downloads, bulletin boards, email, an Internet gateway, multi-player games and chat lines. This change also eliminates the GEnie*Basic package.

GENIE ANNOUNCES NEW PRICING SIMPLER, MORE AFFORDABLE AND, AS ALWAYS, THE BEST VALUE ONLINE

An Open Letter to GEnie Subscribers From John Barber, General Manager of GEnie:

On July 1st, a new pricing structure goes into effect at GEnie. It's a big day for us -- the result of months of planning, number-crunching, monitoring and maneuvering around the competition. And once again, we're looking forward to showing the online community that no one offers a better value than GEnie.

GEnie has always offered its subscribers the best combination of sophisticated services, information, entertainment -- and affordability. And we've done it once again. Starting July 1st, GEnie will offer you the lowest hourly connect time of all the major online services; combined with a highly-competitive monthly subscription fee; and credit each month for up to four hours online.

Effective July 1st, this is GEnie's new U.S. pricing structure:

- Our monthly subscription fee becomes $8.95 a month.
- Our standard hourly connect rate drops to $3.00 an hour.
- And every month, you'll get a credit for up to 4 hours of standard $3.00 connect time.

It's about that simple. For our Canadian and international PDN customers, a complete price chart is available.

SIMPLY BETTER

You might notice one more important benefit to our plan. We kept it simple. It's easy to understand, easy to live with. Multi-player games, downloading, computing bulletin boards, real-time conferences -- they're all just $3.00 an hour.

OK, NOW WHAT DOES IT MEAN TO MY MONTHLY BILL?

Let's cut to the chase. This is what everyone really wants to know. For the people who like to keep to a strict budget each month, this plan offers a lot of flexibility and a broader range of services than before. You have a set subscription fee. And for that fee, you get a credit for four hours of standard connect time to spend just about anywhere on GEnie.

There's no asterisk-chasing to make sure you haven't strayed into Value services. Less worry of credit card shock. You can even explore areas of GEnie you thought you couldn't afford before. For the active users who spend significant hours a month on GEnie, especially in the former GEnie Value services, you should see a tremendous savings. The hourly rate is half what it was before -- and the best in the business!

SOME THINGS NEVER CHANGE

GEnie has always been the very best service for people who really enjoy being online, and who expect great products and the best value for the time they spend with us. That's not going to change. In fact, this new pricing structure should make it even easier for you to enjoy everything we have to offer. We're looking forward to seeing you online.

Sincerely, John Barber, General Manager, GEnie

G E n i e P r i c i n g

(effective July 1, 1993)

GEnie Services	U.S. (U.S.$) [5]	CANADA (CAN $)
Monthly Subscription Fee	$8.95/month	$10.95/month
Hours Credited Per Month [1]	Up to 4 hours	Up to 4 hours
Hourly Connect Charge	$3.00/hour	$4.00/hour
GEnie Premium Services	Prices vary per individual service. These include: Charles Schwab Brokerage Services (not available in Canada), Dow Jones News/Retrieval (R), The Official Airline Guides Electronic Edition (R) Travel Service, QuikNews clipping service, Telebase Investment ANALYST (SM),ARTIST (R) gateway.	
Additional Charges (where applicable) [4]		
Prime Time Surcharge [2]	$9.50/hour	$12.00/hour
9600 Baud Surcharge	$6.00/hour	$8.00/hour
Communications Surcharge:		
800 Service [3]	$6.00/hour	--
Extended Network	$2.00/hour	--
SprintNet	$2.00/hour	--
Datapac	--	$6.00/hour

[1] Credit for up to 4 hours of standard $3.00 U.S. ($4.00 CANS) connect time. Hours credited apply to current month only.

[2] Prime-time: 8 a.m. - 6 p.m. local time on weekdays only. The prime-time surcharge is in addition to $3/hour charge. Prime-time surcharge is waived for selected holidays. Residents of Hawaii, Alaska, Indiana, Arizona and Puerto Rico, please verify hours with GEnie Client Services.

[3] "800" Service surcharge waived at 9600 baud.

[4] State taxes will apply in some areas.

[5] International PDN subscribers billed in U.S.$ at U.S. rates

COMMODORE LOSES $177.6 MILLION IN THIRD QUARTER

Commodore International Limited reported a net loss of $177.6 million on sales of $120.9 million for the third fiscal quarter ended March 31, 1993. This compares with earnings of $4.1 million on sales of $194.6 million in the year ago quarter.

Overall the sales decline of almost 40 percent for the quarter was primarily due to prevailing economic softness in all of the Company's major markets, especially Germany. There was also significant pricing erosion for the Company's older Amiga models and PC products. Unit volume of Amiga products declined 25 percent while Amiga revenues declined over 45 percent.

INEXPENSIVE 4 MEG FLOPPIES

If you need inexpensive EHD floppies for your FD-4000, this is the place. Parsec bought Maxell 3.5 DS/ED 4MB disks, 10 to a pack, for $39. They look pretty cool with the matte black finish, black shutter, gold lettering, and gray labels. B.C.S. also offers bulk 4.0 MB floppies for only $1.99 per disk!

(continued on page 29)

GEOS Machine Language Programming: Icons & Menus

BY ROBERT A. KNOP JR.

I. INTRODUCTION

All GEOS users are familiar with the appearance of icons and pulldown menus. Briefly, icons are little pictures on the screen; when you move the pointer over one such picture and click the button on your mouse or joystick, the program performs some action. Pulldown menus appear in a strip across the top of the screen; selecting one item from a menu may bring up a submenu, or may send the program to perform some action.

Including icons and menus in your own GEOS programs is easy. You merely need to supply the GEOS Kernal with a couple of tables that define how many icons and menus you want, where you want them, what they look like, and the "service routines" to call when they are selected. GEOS will then take care of detecting when an icon or menu has been selected, and all you need to do is write the service routines.

II. ICONS

THE ICON DEFINITION TABLE

You specify what icons will appear in your application in the Icon Definition Table. At the top of this table is a header:

```
.byte     Number of Icons
.word     Initial X Position
.byte     Initial Y Position
```

The initial positions specify where the pointer will appear when the icon table has been set up. This lets you start the user out with the pointer over a given icon, or anywhere else on the screen. If you pass all 0's for the initial positions, GEOS does not place the pointer anywhere, but leaves it where it was before the icon table is drawn.

Following the four byte header is one eight byte entry for each icon. There must be at least the number of eight byte entries as the number of icons you specified in the table header. Each icon entry follows the format:

```
.word     Pointer to Icon Picture
.byte     X Position (in cards)
.byte     Y Position (in scanlines)
.byte     Width (in cards)
.byte     Height (in scanlines)
.word     Service Routine
```

The first word should hold the address in memory of the picture for the icon. This picture should be in the GEOS compressed bitmap format - but geoProgrammer owners need not worry about that. for all they need to do is paste a photo scrap into the geoWrite source file. (Note that you do NOT paste the picture in within the icon definition table; rather, put it on another page with a label like "Icon0Pic", and then put that label as the first word in the icon's entry in the icon definition table.) If the pointer to the icon picture is $0000, this icon is disabled. You can use this to turn icons on and off during the execution of your program.

The x and y positions are measured from the top left corner of the screen. For icons, all X values are given in cards. This means that icons must be an integer multiple of 8 pixels wide, and must appear a multiple of 8 pixels from the left of the screen. You have complete freedom for the height and Y placement of an icon. In order to use the same icons in both 40 and 80 columns under GEOS128, you can set the "bitmap doubling bit" in the X-position and width entries in the table. Unfortunately, GEOS 64 does not recognize and ignores this bitmap doubling bit as it should. If it is present, it will produce unpredictable results with GEOS64. When the high bit is set in either the X-position or the width byte in an icon's entry, GEOS will automatically double those values when the icon is drawn on the 80 column screen. It will even take care of stretching the icon picture, so you need only supply one picture. In practice, you can set this bitmap doubling bit by ORing the value with $80, e.g.:

```
.byte     $80  IconWidth
```

Bitmap doubling will be discussed at greater length in the next article in this series.

The last word in the icon's entry is the address of the service routine to be called when the user clicks on the icon in question. Before calling the routine, the GEOS will flash or invert the icon depending on the value of iconSelFlag ($84b5). STFLASH ($80) makes GEOS flash selected icons, STINVERT ($40) makes GEOS invert them permanently, while $00 indicates that GEOS should just directly call the service routine without doing anything to the screen image of an icon. You can configure the speed of this flash (if STFLASH is set) by changing the value in selectionFlash ($84b3). Higher values mean a slower (longer) flash. The default value is SELECTIONDELAY (10).

To install an Icon table in an application, you use the GEOS Kernal routine DoIcons ($c15a). Load the zero page word length pseudoregister r0 ($02) with the address of your icon table, and call the routine. GEOS will draw the icons, and begin detecting icon events with the icons in this table. You can disable individual icons at any time by setting their picture pointer word to $0000, and re-enable them by setting the word to anything other than zero; no call to DoIcons is necessary. To replace the entire set of current icons with another, you can at any time load r0 with the address of a different icon table and make another call to DoIcons. You will need to erase the old icons yourself, however, as DoIcons will not erase old icons from the screen, but will only draw the new ones on the screen. If you wish to use no icons, or you want to disable altogether a current set of icons, you can pass a "null" icon table to GEOS:

```
.byte     1          ;One icon
.word     0
.byte     0
.word     $0000      ;0 picture means disabled icon
.byte     0,0,0,0    ;Positions, sizes all zero
.word     $0000      ;null service routine
```

Why not just pass it a table with the number of icons as zero? GEOS expects that every application will have at least one icon at all times. So, if you do not want to use icons, you need to call DoIcons with a table that specifies one disabled icon as is shown in this example.

Since GEOS needs the icon table as long as it is active, VLIR applications (discussed in the previous article in this series) must keep the table in the resident section of their code. If you do place it in an overlay module, you must be sure that this overlay module will be in memory as long as this icon table is active. The images for a set of icons, on the other hand, are only needed at the time of the call to DoIcons, so it would be feasible to have an initialization module that contains the icon pictures, which gets swapped out following the call to DoIcons.

ICON SERVICE ROUTINES

The service routine called when a user clicks on an icon can do most anything - even replace the current list of icons with another through a call to DoIcons. The service routines for icons must either be in the resident module of your code, or you must be careful to disable relevant icons before swapping out an overlay module with icon service routines. In practice, this is not a severe limitation because you can write a three line resident "stub" service routine for each icon which

(1) loads the accumulator with the number of the overlay which contains the meat of the routine,
(2) calls a routine that checks if the accumulator-specified overlay module is in memory, and if not, loads it, and
(3) jumps to the meat of the routine in the overlay module.

When the service routine starts up, GEOS passes a couple pieces of information to it. r0L will have the number of the icon which the user selected. This can be useful if you want to use one routine for a number of different icons, with a slightly different effect for each icon. Consider, for example, a calculator program which really only needs one routine for all of the "number" icons.

r0H holds the double-click flag; TRUE ($ff) indicates a double click, while FALSE ($00) indicates a single click. This is fine if you want the service routine to run with only a double click; utilizing this flag becomes more challenging if you have two separate behaviors for single clicks and double clicks, as your routine will actually be called twice during a double click, once after the first click (with r0H set to FALSE) and one after the second click (with r0H set to TRUE). There are a couple of ways to deal with this. First, during the first call you can watch the mouseData ($8505) variable (the high bit of which is 0 when the mouse button is down, 1 when it is up) to see if the user performs a release and a second click within a given time limit. Second, you can have the double click action be a superset of the single click action. The latter is what the deskTop does with file icons. One click, be it the first of a double click or a solitary click, selects a file's icon. The second click of a double click opens the file.

Upon a call to DoIcons, icons are drawn from lower number to higher number. This means that if two icons overlap each other, the higher numbered icon will appear "on top." However, when the GEOS mainloop decides which icon service routine to call, lower icon numbers take precedence. Because of this discrepancy, it is best to keep your icons from overlapping. If you must overlap icons, to avoid user confusion you may need to redraw some of the icons manually. (The graphic routines you would use to do this will be discussed in the next article in this series.)

III. MENUS

Menus, like icons, are defined with a table. The menu table is somewhat more complex, allowing for submenus. Unlike icons, applications do not need to have menus, so an application which wishes not to use them need not install a trivial menu table as is the case with icons.

Like an icon definition table, a menu definition table begins with a short header:

```
.byte     Top of Menu (pixels)
.byte     Bottom of Menu (pixels)
.word     Left of Menu in (pixels)
.word     Right of Menu in (pixels)
.byte     Number of Items  Menu type
```

Note that unlike icons, you can specify X menu positions to the pixel. While bitmap doubling is again available for the X values (by setting the high bit of the word), it is less useful for menus because the BSW128 80 column system font is not exactly twice as wide as the BSW 40 column system font. For GEOS128 applications that support both 40 and 80 columns, it is probably easiest to just have two separate menu tables.

For vertical position and height values, you can figure that each line in a menu needs 14 pixels. Horizontally, you are on your own. One tried and true method is to experiment, and tweak the x position and width values until they are right. (GeoDebugger can be very useful for this.) Prior to version 2.0, GEOS did not deal well with menus that went further right than 255 pixels. While this is not usually a problem in 40 columns, it severly limits the potential width of 80 column menus. This shortcoming has been fixed in GEOS128 2.0.

The low 6 bits of the last byte in the menu table header specify the number of items in the menu. The 6th bit specifies whether this menu is "constrained" (bit6=1) or "unconstrained" (bit6=0). When a constrained sub-menu is open, the user will only be able to move the pointer off of it in the direction of the parent menu to this submenu. You do NOT want to make your main top-level menu constrained. The 7th bit of the final byte in a menu table header specifies whether the menu is horizontal (0) or vertical (1).

Your main menu, to fit the pattern long established by existing GEOS applications, should be horizontal. To set these two bits, you can use the geoAssembler OR operator and the geoProgrammer constants "horizontal", "vertical", "constrained", and "unconstrained".

Following the menu header is a number of menu entries equal to the number in bits 0-5 of the last byte of the header. Each menu entry has the form:

```
.word     Pointer to Menu Text
.byte     Menu Item Type
.word     Pointer to Routine or Submenu
```

The menu text is simply a pointer to a null terminated text string. One common trick with this text is to leave two blank spaces at the beginning of the text for "toggle" menu items. When this menu is selected, you can put an asterisk in place of the first space to indicate the selection. GeoWrite uses this for its font and style submenus.

The menu item type is one of MENUACTION ($00), SUBMENU ($80), or DYNSUBMENU ($40). MENUACTION menu items are much like Icons. The last word in the item's menu table entry specifies a routine to call when this menu item is

selected. Before calling the routine, GEOS will flash the menu item on the screen. The rate of this flash is controlled by the same variable (selectionFlash) that controls the rate of icon flashing.

The final word of menu table entries for SUBMENU type menu items is a pointer to another menu structure. When the SUBMENU menu item is selected, GEOS will open and display the menu specified by the submenu table. The submenu table has exactly the same format as the main menu table. beginning with a menu table header.

A "dynamic submenu," a menu item of the type DYNSUBMENU, is a cross of the other two. It allows you to call a routine before opening a submenu. This can be useful if you need to do some system cleanup whenever a submenu is opened. The final word in the item's entry in the menu table specifies the routine to be called. At the end of that routine, load the word r0 ($02) with a pointer to the submenu table. and upon returning from this routine GEOS will open the submenu. Return $0000 in r0 to open no submenu.

As with icons, you submit your menu table (and all of its daughter submenu tables) to GEOS by loading r0 with a pointer to the top level menu table and calling DoMenu ($c151). The entire menu table, as well as all of the text for all of the menu items. must be in the resident portion of VLIR applications, as GEOS refers to it when determining which menu items have been selected and when opening submenus.

MENU SERVICE ROUTINES
Like icon service routines, menu service routines must be resident. Again, you can use resident "stub" routines to call routines in overlay modules. Menu service routines have one additional requirement: before doing anything else, they must take care of the opened menus. Fortunately, this is easy. Most menu service routines will simply call the routine GotoFirstMenu ($c1bd), which retracts (erases) all opened submenus.

The routines called by DYNSUBMENU items have additional data available to them. The variable menuNumber ($84b7) holds the current submenu depth; when the current menu is the top level menu, menuNumber will hold the value 0. The accumulator holds the item number within the current submenu, where the first item in the menu is item number 0. Additionally, dynamic submenu handler routines may wish to make use of two additional routines: DoPreviousMenu ($c190) retracts the current submenu and activates its parent menu. ReDoMenu ($c193) reactivates the current submenu.

IV. SELECTIONS ELSEWHERE
GEOS takes care of calling the appropriate routine when the user clicks the mouse or joystick button with the pointer over an icon or a menu item. What about other clicks anywhere else on the screen? To intercept these mouse events, load the "vector" otherPressVec ($84a9) with the address of a routine to be called whenever the user clicks on a position other than an icon or a menu. (A "vector" is just a word that specifies an address of a routine that is called under certain circumstances.) Load otherPressVec with $0000 if you want to ignore these events.

While the routine in otherPressVec is only called for those presses with the pointer not on a menu or an icon, it is called every time the user releases the button, regardless of whether or not the original press was over an icon or menu. If you wish to ignore releases, simply check the high bit of the variable mouseData ($8505). If the high bit of this is 0, the routine was called as a result of a mouse button press. If it is 1, the routine as called as a result of a mouse button release. If you do want to detect releases, you will have to develop your own code for detecting whether the original press was over an icon or menu. An easy example would be to set some application flag every time the otherPressVec routine is called due to a press. When it is called due to a release, it can check this flag to find out if the mouse had been somewhere other than an icon or a menu when the button was pressed.

This wraps up the article on icons and menus. There are a few details which have been left out. some of these will work themselves into later articles.
Next in the series: all about the plethora of GEOS Kernal graphic routines, and using graphics in GEOS, including a discussion of bitmap doubling on the GEOS128 80 column screen.

CLASS(Y) ADS

Starting with issue #35 we will have a new classified section with a low per line ad charge. All ads have to contain either a phone number OR a mailing address. People with POBs have to submit a street address to Parsec with a matching night time telephone number that can be verified with directory assistance. We will not release the street address to anyone UNLESS there is an UNRESOLVED problem with your ad. We will notify you first.

CLASS(Y) ads can be submitted on either 1541 or 1581 disks as either PetAscii or straight Ascii sequential text files. You can also submit them printed by hand or printed out on a printer. Parsec is not responsible for omissions or typographical errors. If you can send matching hardcopy with your ad it will be appreciated.

BUYER & SELLER BEWARE!
The guidelines to buying through the mail are unless you know the person well:

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

RATES and RULES

$0.50 a line with a minimum charge of $10 per ad.

Each line is 35 characters wide.

Only alphanumerical characters allowed plus puncuation. no special characters.

The Seller must be a TC-128/64 subscriber.

Commercial advertisers have to submit a copy of their product for review.

Your ad does not have to be computer related.

We reserve the right to cancel any ad for any reason. Canceled ads will have their money returned.

If there is not a category already created for your Class(y) Ad. then we will create one.

GateWay 2.5 review

by Robert A. Knop Jr.

INTRODUCTION
In issue #31 of Twin Cities 128, Dick Estel reviewed CMD's replacement desktop for GEOS, known as The GateWay. So why is it being reviewed again here? Well, CMD has subsequently released version 2.5 of GateWay, which I will be addressing in this review.

First, for those of you who have not seen the review in issue #31 and who are not familiar with the GateWay, I shall briefly summarize what it is. The GateWay is a program you can use to interact with GEOS in place of the deskTop, which is supplied with GEOS. Versions of GateWay are available for both GEOS64 v2.0 and GEOS128 v2.0. The GateWay was originally introduced in order to allow GEOS to be compatible with CMD's memory storage hardware such as RAMLink. Since then, CMD has released a number of patches which can be made to the normal deskTop GEOS system in order to use these devices with GEOS, but GateWay remains available as an alternate user interface to GEOS. Additionally, despite the patches for the deskTop, the GateWay still has more complete support for CMD memory devices such as RAMLink.

WHAT'S NEW?
What's new with version 2.5 of GateWay? The feel of the program, and the basic functionality, is the same as it was in previous versions of the program. One change is the name in the author box. GateWay was originally written by Paul Bosaski. Jim Collette has now taken over the project. For those of you who do not know of Jim, he is probably one of the most prolific and talented GEOS programmers out there, and one who knows GEOS better than almost anybody else. The other evident changes to the GateWay are numerous bug fixes and performance improvements. One no longer has to worry nearly as much about surprise crashes and the general operation of the program is smoother.

DESCRIPTION AND CHANGES
GateWay's basic operation was described in the review in issue #31, so I will only briefly summarize it here. Basically, in place of the deskTop's icon-laden disk notepad, there is a window which lists the names of the files on the current disk. To the left of the name is a small "mini-icon" which indicates the type of the file (e.g. System, Application, Application Data, etc.). The notepad can be made either narrow, just showing the filename and updating the screen faster, or wide, showing filename, file type (in words), and file size, at the cost of slower screen updates. You select files by clicking on the filenames.

As with the desktop, multiple files can be selected by holding down the C= key. Below the window with the filenames is the trash can and three icons for disks A, B, and C. Unlike the DeskTop, GateWay directly supports all three disk drives without having to swap drives. However, swapping is still available, for backward compatibility with BSW applications.

There is a full suite of menu options, roughly equivalent in functionality to that of the deskTop. One notable difference is the "geos" menu, which instead of Desk Accessories features "GateWay Documents." GateWay documents may be installed into the GateWay program file itself using the GWMover utility. This allows you to expand the functionality of the GateWay at the cost of a larger GateWay file.

As with previous versions of GateWay the full functionality of the program is not available unless you have some form of RAM expansion, i.e. either a RAMLink, RAMDrive, or REU.

Now that version 2.5 of GateWay has been released, how has it improved from previous versions? The fundamental and most important change is that it is much more stable and less buggy than previous versions. I have an older copy of GateWay, and briefly used it, but before long I shelved it and went back to the deskTop because unexplained system errors were far too frequent. This is no longer the case with version 2.5, which has performed consistently and reliably with my system.

This greater stability extends to the Switcher too. Briefly, the Switcher is a task switching utility for GEOS, similar to though easier to use than geoWizard. For example, from within the GateWay, you can activate the Switcher, and then run geoWrite. Hit escape, and you are out of geoWrite, and back at the GateWay. Run geoPaint, and thereafter, hitting the ESC key swaps you between geoWrite and geoPaint. Basically, you can have two separate GEOS sessions going, and swap between them at will. With previous versions of GateWay, you were most likely to crash when using the Switcher. Now, however, the Switcher has been completely rewritten, and is rock-solid. And, on the C-128, all 128K of the 128's RAM is saved, unlike the previous version which only saved the 64K of GEOS "FrontRAM." The switcher is a little slower than the older version, it takes 3 or 4 seconds to swap between processes, but this is very minor and well worth the added stability of the switcher.

One important difference in functionality is how GateWay deals with the trash. With the deskTop and older versions of the GateWay, you could only sometimes recover the last file deleted. With the current version of GateWay, every file you delete is stored in the trash, until you explicitly select "empty Trash" from the "special" menu. You can recover any of these files by clicking on the trash can and double clicking on the filenames you wish to recover.

COMPARISONS WITH OTHER DESKTOPS
So, which should you use, deskTop, or GateWay? This remains very much a matter of personal preference, the most important question still being, which do you prefer, more intuitive icons or more flexible filename lists? While the GateWay is more modern that the deskTop and, I would venture, somewhat more powerful, the difference is not enough to overcome personal preference of icons vs. no icons. If one single reason were to convince me to use the GateWay over the deskTop, it would be GateWay's ability to copy new versions of multiple files from one disk to another without having to verify overwriting each and every single file. If one reason were to convince me to use the deskTop over the GateWay, it would be the deskTop's ability to list files by date.

A more relevant comparison might be GateWay and Dual Top. Why? Both operate on the same idiom i.e. lists of filenames. Basically the difference in functionality between these two programs is how they use the width of the screen. Dual Top uses it to list filenames from two disks. GateWay uses it to make

available at a glance more information about each file. Each of these has its advantages and disadvantages. In terms of the feel of the two programs, the operation of Dual Top is generally faster than that of the GateWay, especially in 80 columns on the 128. This is most evident when scrolling through the filename list, which flies on Dual Top but is still somewhat slow in GateWay. Also, Dual Top at least has some support for a fourth disk drive, something GateWay doesn't even try to do.

COMPLAINTS

My first complaint about the GateWay is that it will not list files by date. When you list files with full information, the information you get is filename, file type, and size. Size is nice, but why do we need to see the file type? That information is there already, in the form of the mini-icon to the left of the filename. So listing the type type is redundant. This space could have been used much better by listing the date and time of creation for the files. Especially when you consider almost every piece of CMD hardware comes with a clock or has it as an option. The only way in GateWay to see the date of the file is to pull up that file's "info" box. Dated files are one of the great conveniences of GEOS. I think that Commodore programmers have lost track of how useful this can be by living for 10+ years with a filing system that does not store nor show the creation times of files.

My second complaint has to do with the support of CMD hardware devices. Before making this complaint, I want to temper it by noting that GateWay unsurprisingly has better support of CMD devices than any other deskTop in my experience.

Thanks to Jim Collette's patches (available from CMD, and included with RAMLink and other CMD hardware), GEOS and the deskTop can be used with CMD hardware devices, such as RAMLink, using 1581 partitions. With another program from CMD, geoMakeBoot, you can even boot GEOS from CMD devices. All of this functionality and more is included with the GateWay. The GateWay works naturally and easily with my RAMLink. Additionally, with the GateWay you can make any GateWay supported disk a GEOS boot disk. Booting GEOS/GateWay off of my RAMLink takes less than 20 seconds- after which GEOS is ready to go, impressive!

The other thing GateWay supports that non-GateWay GEOS does not is CMD native mode partitions. (Note: thanks to a tipoff from Steve Vanderark, I found that the deskTop and Dual Top, when run from within GateWay, do in fact work with these native mode partitions.) Supplied with GateWay is CMD MOVE, a program that lets you copy files between CMD partitions. Though, for an example, on a RAMLink, if you are running in a native partition, you can only copy to other native partitions, and if you are running in a 1581 partition, you can only copy to other 1581 partitions.)

What is sadly lacking, however, is complete subdirectory support! One of the biggest advantages of CMD native mode partitions is the ability to make subdirectories in those partitions, so you can organize the files in a huge partition logically. It would be nice if this were completely supported in GEOS.

Some subdirectory support is present in GateWay. With the GateWay document "MakeDir," you can make a new subdirectory. Opening that subdirectory is as simple as double-clicking on its name. This makes that subdirectory effectively your current disk. Clicking on the close icon at the upper left of the filename window puts you into the directory just above your current subdirectory, or lists all possible partitions you could select if you are already at the top directory.

This is all well and good, but not enough. In order to open a document in a subdirectory, a copy of the document's application must be present either in the same subdirectory, or on another disk. Both of these present problems: if you have copies of all applications in all subdirectories where there are documents for these applications, you are wasting a lot of space by keeping multiple copies of the same applications in many places on the same disk. Choosing to keep the application on another disk usually means a physical disk such as a 1541 or 1581 and doing this you lose the speed advantage of having your application on a RAM disk.

The barely acceptable hack "Bordercross" is a GateWay document which presents a partial solution to these problems. When you select a file and then choose the "Bordercross" option under the GEOS menu, the file is moved to what would be the deskTop border. These border files are visible at the end of the directory listing for ALL subdirectories in a partition. This allows you to have just a single copy of up to eight applications (and you will want the GateWay to be one of these eight) and have documents for those applications spread throughout directories in the current partition.

While this is nice, it is also flawed. First, there is no way within the GateWay of telling which files are actually on the border. They are at the end of the directory listing, but there is nothing other than that to set them apart from the normal files in the current directory.

Second, there is no way within the GateWay to subsequently get files off of the border. If you try to Bordercross a file already on the border, that file simply disappears . What is needed is the ability to set a "search path" in the same way one can using CP/M. In other words, when an application is not found in the current directory, GateWay should search a user-configurable set of devices and directories for that application.

A second complaint about subdirectories is that there is no easy way to copy files between subdirectories. CMD MOVE will copy files between partitions, but, perversely, not between subdirectories in the same native mode partition. Basically, the only way to copy a file between subdirectories is to copy it off of the disk onto another disk, change directories on the CMD device, and copy the file back to the CMD device. For the person with a single 5.25 disk drive this can be a hassle and they lose the speed and ease of use advantages that prompted them to buy the CMD mass storage device in the first place.

DOCUMENTATION

The manual included with GateWay is the same as that with previous versions. There is a small four page insert that describes the changes and new features of version 2.5. The manual is in general well written, although occasionally verbose, and is nicely printed as are other CMD manuals. One thing to be aware of: pay very close attention to the section on configuring your system and first installing GateWay. If you have a RAMLink, remember how you had everything set up when

you make your boot disk. If you make even a change as small as switching the REU in a RAMLink from "Normal" to "Direct," GateWay will completely fail to boot Fortunately, you always have the option of starting over from scratch.

CONCLUSIONS
GateWay version 2.5 is a solid product. If you use GEOS with any CMD devices or find yourself using the "show by name" option of deskTop frequently, or desire the power and convenience of the Switcher, then GateWay is probably for you. While the lack of date support is maddening, the subdirectory support in GateWay, though not "enough," is better than is present in any of the other alternative deskTops.

GRADE: B+
COST: $29.95

Creative Micro Designs, Inc.
PO Box 646
E Longmeadow MA 01028
800-638-3263 Orders
413-525-0023 Information

THE GEOPUBLISH COMPENDIUM A REVIEW

BY JOHN W BROWN

The line "A guide to using geoPublish with a laser printer, even if you don't own one!" on the cover of "The geoPublish Compendium" pretty much sums up what this small twelve page booklet is all about.

The paper masters are printed on a Postscript laser printer a at print shop by the author. David B Ferguson, using his Commodore computer and geoPublish to create the data files.

This booklet contains a brief tutorial on what Postscript is, how to modify Postscript files produced by geoPublish to get them ready for the printer, (briefly) how to transfer them to an IBM PC disk, (briefly) how to print them on a IBM PC at the printers, and how the booklet was actually produced.

The overall look and value of the booklet is good, especially considering that the shipping price is included in the price of the product. My copy of the booklet came with a few additional example sheets.

If you belong to a major telecommunications network you could skip this booklet and find all your answers online. National networks have dedicated Desk Top Publishing and Postscript RoundTables where you can pick up all the Postscript and Commodore information you need.

Which brings me to the reason why I graded this booklet "only" a B. The names and addresses of where to find the files that he mentions are left as "Q-link". I think this is a serious omission. The majority of people who could really use this booklet, like our readers, do not own modems, so they can not easily get access to the files mentioned unless they write to a public domain software company (such as Parsec) and directly ask for the needed files.

I am sure most of the sources he did list, including himself, would gladly send someone a disk with the needed drivers and programs for a buck or two. Though I am sure most people would be at a loss on how to contact Adobe for their "bible" on Postscript and without the ISBN numbers for the books he did mention, tracking them down will be a little more work.

The other negatives were the obvious printing mistakes using Postscript geoPublish that would have been fairly easy to fix with a little more effort. Since the copyright on the booklet is 1991, I would have hoped in two years the author would have found time to correct the mistakes. If you are going to espouse the benefits of using a product and procedure to make a printed product, make it visually correct!

On the plus side, the above words are probably a bit too harsh for a $4 product that includes the shipping costs. It is a good cost leader for an author that wants to get his foot in the door for other products he wants to sell you. For $4, you can not go wrong by purchasing it.

Having all the information you need to get started in one easy to read booklet will appeal to many people that do not feel like looking for it online. Or for those that do not want to search through magazines that carries columns on the same subject, such as Don Lancaster's in Computer Shopper.

If it were not for these few persnickety details, the product would rate a solid "A", but, I just could not give it that since the booklet is about what an excellent job you can do using geoPublish on a Postscript printer.

Grade: solid B

Price: $4

U.S. funds, CA residents add sales tax, overseas add 15% for S&H

GEnie: DiBieF"
209-883-2550
Quincy Softworks
9479 E Whitmore Ave
Hughson CA 95326-9745

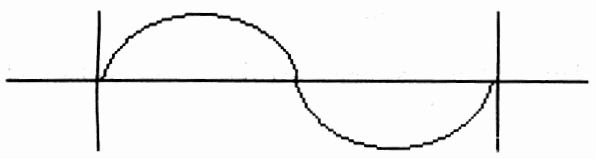
DIGI-BLASTER 64

BY STEVE GOLDSMITH

PLAY BACK DIGITIZED SOUND ON
YOUR C64 THROUGH SOFTWARE!
Have you ever wondered how digital sound is created on computers or compact disc players for that matter? There has not been much information published on digital sound play back, using software, on any computer platforms. Most of the available information I have found takes a Rocket Scientist to figure out. What I am going to do here is cover play back of digitized sound on a C64 in terms that most programmers can understand and use in a wide variety of applications. Even though this sound driver is written for the C64, the information provided here is not limited to that platform. I have written drivers for the C64, C128 in 2MHz 128 mode and IBM PC DOS. I plan to do a Z80 driver for C128 CP/M mode too.

BASICS
I am sure most of you know that sound is the movement of air molecules caused by objects that vibrate. A good visual example of this would be to take the grill off one of your stereo speakers and turn up the music. If you watch the bass speaker carefully you can see it vibrating. The speaker causes the air molecules to vibrate your auditory nerves that let you hear (feel) the music.

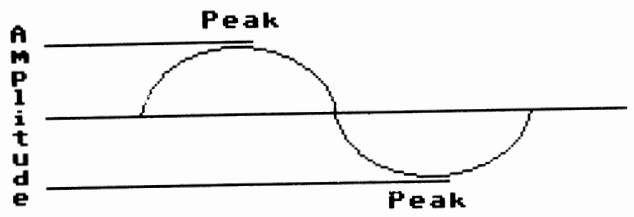
Sound can be represented by sine waves. The two components of a sound sine wave are frequency and amplitude. Frequency is the number of cycles per second



Wavelength

One Cycle

Increasing frequency makes the sound pitch higher and decreasing frequency has the opposite effect. Amplitude is the loudness of a sound and determines the height of a sine wave.

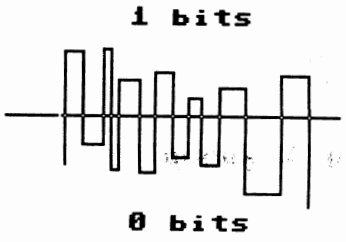


Increasing amplitude makes the sound louder and decreasing amplitude has the opposite effect. Real sound waves are much more complex than simple sine waves. The computer must be able to convert these complex analog signals into digital data. The digital data is then converted back to a analog signal that you can hear. This process is called sampling or digitizing.

SAMPLING
The sampling rate is the frequency at which a digitizer grabs a small chunk of sound. The frequency is usually represented in Hertz. Increasing the sampling rate improves the quality of the sound and increases RAM usage. Decreasing the sampling rate has the opposite effect. If you were to record a sound at 16,000 Hertz (16KHz) the digitizer would grab 16,000 chunks of data per second. The amount of memory required to store the digitized image depends on the method implemented. The highest quality and most widely used method for producing digital play back is called pulse code modulation or PCM. Compact Disc (CD) players use 16 bit PCM at 44KHz.

The stock C64 could only hold .74 seconds of mono sound sampled at CD player resolution! Actually, the C64 is not capable of playing 16 bit data directly or sampling at rates of 44KHz. You can get 16 bit PCM digitizers for IBM PC class computers with lots of memory and hard drive space for about $300.00 to $400.00. There have been some 8 bit PCM digitizers for the C64 and C128, but you still suffer from lack of RAM. What we need is an efficient sampling method that still produces good quality sound.

Square Wave Format
The most efficient uncompressed method of digitizing sound is the one bit square wave format.



Sample complex square wave sound

The C64 version of a Covox Voice Master uses this method. The digitizer stores either a 1 or 0 bit in memory depending on the sound's amplitude. These strings of 1s and 0s determine the sound's waveform. The highest frequency possible would look like 01010101 and a lower frequency would look like 10001000. Data in this format can be played back through the C64's SID chip using software!

DIGI-BLASTER 64
Digi-Blaster 64 was created because I was disappointed with the play back quality of the software that came with the C64 version of The Covox Voice Master. Even using its highest sampling rate (12KHz) the quality was poor, so I set out to write my own Assembler modules to record (maybe I will cover this in another article) and play back sounds. What I ended up with are modules capable of 25KHz sampling on a 1MHz C64! The Assembler module DB64.ASM plays back one bit square wave data.

The way it works is to read a bit of sample data per cycle. If the bit is a 1 then set the SID chip's master volume control at $D418 to the value stored at location $A9. If the sample bit is 0 then set the volume control to 0. This rapid flipping on and off of the volume control vibrates the speaker and creates the sound. Since this process involves intensive real-time processing we have to disable interrupts and blank the screen before entering the processing loop. You can

play back sound with the screen visible. but you lose sound quality. For more detailed information on the workings of DB64.ASM read the documented source code.

CALLING DB64.OBJ
You need to do a few things before calling DB64.OBJ:

1. Set locations $A3 and $A4 to sample the starting address in lo/hi format.
2. Set locations $A5 and $A6 to the sample length in lo/hi format.
3. Set location $A7 to the sampling rate The sampling rate formula is 255-PEEK(167) clock cycles.
4. Set location $A9 to the volume for 1 bits of sample data. Volume must be between 1 and 15.
5. Call DB64.OBJ at $C000.

To call the routine from your own BASIC program just load DB64.OBJ. load your digitized file(s). set the locations listed above with POKE statements and SYS 49152. I use this method for my player program DB64.BAS.

ASSEMBLER PROGRAMMERS
Proficient 6502 Assembler Programmers will be able to exploit the power of DB64.ASM! DB64.ASM is written with Rebel Assembler in C64 mode. but the source code can be easily converted to many 6502 Assembler formats. You can change the origin of the DB64.OBJ file if you need to use memory in the $C000 - $CFFF range. utilize the C64's MMU to stuff digitized data under the Kernal and BASIC ROMs and flip them out for play back. create real-time effects like echo. fade-in. fade out. etc. etc. The possibilities are only limited to your imagination and programming experience. I will cover some of these techniques in future articles.

SOUND SOURCES
There are many sources of digitized sounds. You can make your own sound files with the C64 version of The Covox Voice Master or any digitizer that creates the same one bit square wave format. The sound files used by DB64.BAS were created with a Sound Blaster Pro on a IBM PC running under MS DOS. I used a 16KHz sample rate with a low - 3.2KHz filter. I then converted the 8 bit PCM file to a 1 bit square wave file with a sound editor I'm working on and transferred the files to my 128D with Big Blue Reader 128. In a future article I will cover converting 8 bit PCM VOC files created with the Sound Blaster, Covox Sound Master II, etc. to 4 bit PCM and 1 bit square wave files that can be played back on the C-64 and C-128. That way you can download some of the most popular 8 bit PCM sound formats from GEnie, local BBSs, etc. without even owning a digitizer.

SUMMARY
I have only scratched the surface of digital sound processing without going into Statistics or Calculus III. Hopefully after reading this article and running the sample programs you have gained a basic understanding of this complex subject. I suggest you experiment (play) with the DB64.BAS and DB64.ASM source code to get a feel of what the C64 is really capable of doing. In BASIC try decreasing the volume at location $A9 from 15 to 1 in steps of 1 and playing the same digitized sound each time you decrease the volume to produce an echo effect. Once you get good you should be able to make the aliens in your game say "Prepare to Die Earth Scum!" with echo instead of just making a SID sound effect! You can send me E-mail on GEnie at address S.GOLDSMITH2. I hang out in the Commodore Flagship, CP/M, Borland and MS DOS areas.

If you want some good examples of programs that utilize the techniques described in this article and have access to GEnie or local BBSs in the Sarasota, Florida (813) area code download:

DSS1.SDA
Digital Sound Samples is a long playing music sampler for the C64.
By Steve Goldsmith

DSS2.SDA
Digitized music files needed by DSS1.SDA

DIGISND.SDA
DigiSound is a animation and digitized sound demo for the C64.
By Steve Goldsmith

VEPLAY.ZIP
Voice Editor Player Demo plays digitized voice. music and game sound effects for the IBM PC running DOS.
By Steve Goldsmith

PROGRAM NAME:DB64.BAS

```
aj 10 rem ================================
ec 20 rem digi-blaster 64 1.1      01/24/93
gb 30 rem (c) 1993 by parsec inc
ek 40 rem all rights reserved
db 50 rem ================================
fb 60 on rs goto 160,240
bm 70 gosub 440 :rem display title
db 80 gosub 520 :rem get disk device #
kl 90 gosub 610 :rem set up file names
ll 100 rs = 1 :rem run state = load first play file
ol 110 poke 167,127 :rem sample rate
ca 120 poke 169,15  :rem high bit volume
ho 130 print"loading play back module..."
cf 140 print
mj 150 load"db64.obj",dv,1
md 160 print "playing files..."
im 170 rs = 2 :rem run state = play file
jb 180 if f$(f)="end" then end
en 190 geta$
pe 200 if a$<>"" then end
pe 210 print f$(f);
cl 220 print left$ ("                    " 17-len
       (f$(f)));
jc 230 load f$(f),dv,1
cf 240 gosub 270 :rem play file
bd 250 f = f+1
li 260 goto 180 :rem get next file
nn 270 sl = 0              :rem start lo
ik 280 sh = 194            :rem start hi
li 290 s  = sh*256+sl      :rem start addr
fp 300 el = peek (174)     :rem end lo
mn 310 eh = peek (175)     :rem end hi
je 320 e  = eh*256+el      :rem end addr
ni 330 l  = e-s            :rem length
hf 340 ll = l-(l/256)*256 :rem length lo
hp 350 lh = l/256          :rem length hi
km 360 print s;e;l :rem print info
jn 370 poke 163,sl :rem start address lo
kn 380 poke 164,sh :rem start address hi
ig 390 poke 165,ll :rem length lo
nc 400 poke 166,lh :rem length hi
ci 410 sys 49152 :rem call play module
oh 420 poke 54296,0 :rem volume = 0
dn 430 return
mj 440 poke 53280,0 :rem border color
nc 450 poke 53281,0 :rem screen color
go 460 print chr$ (147);chr$ (14);chr$ (5)
```

```
fm 470 print"Digi-Blaster 64 1.1"
ce 480 print""
dj 490 print"Commercial Software from"
cf 491 print"TC-128/64 Issue #34      "
jn 492 print"copyright 1993 by        "
bo 493 print"Parsec, Inc.             "
le 494 print"PO Box 111               "
ia 495 print"Salem, MA 01970-0111     "
pe 496 print"USA                      "
di 500 print""
in 510 return
kd 520 print
ff 530 print "Disk device number: ";
gd 540 open 1,0
ci 550 input# 1,dv$
de 560 close 1
nf 570 print
np 580 dv = abs (val (dv$))
ha 590 if dv<8 or dv>15 then 520
oi 600 return
de 610 nf=8
nf 620 dim f$(nf)
fd 630 for i=0 to nf
ae 640 read f$(i)
ao 650 next
ce 660 return
mn 670 rem file names
gn 680 data db64.sam
hn 690 data sg1.sam
cf 700 data guitar1.sam
dh 710 data guitar2.sam
jp 720 data sg2.sam
fd 730 data guitar3.sam
gf 740 data guitar4.sam
mb 750 data sg3.sam
ab 760 data end
```

PROGRAM NAME:DB64.ASM

```
ia 100 ;================================
ok 110 ;digi-blaster 64 1.1
lj 120 ;digitized sound player 01/24/93
bp 130 ;(c) 1993 by parsec inc po box 111
lp 131 ;salem ma 01970-0111 usa
fk 140 ;all rights reserved
lc 150 ;================================
nl 160 ;
lk 170 ;designed to play back sound
eg 180 ;sampled with covox voice master
hn 190 ;or other uncompressed 1 bit
ig 200 ;square wave formats.
an 210 ;
ia 220 ;rebel 64 assembler format
cb 230 ;
ao 240 ;cia chip
df 250 ;
nc 260 timea2 = $dd04 ;cia 2 timer a lo byte
pb 270 icr2 = $dd0d ; cia 2 interrupt control register
mo 280 cra2 = $dd0e ;cia 2 control register a
fo 290 ;
hj 300 ;vic chip
hc 310 ;
kj 320 viccr = $d011 ; vic 2 control register
ig 330 ;
jp 340 ;sid chip
jk 350 ;
po 360 sidvol = $d418 ;sid master volume control
ko 370 ;
bj 380 ;0 page memory used by play routine
mc 390 ;
da 400 samsta = $a3 ;sample starting addr
ob 410 samlen = $a5 ;sample length
do 420 samrat = $a7 ;sample rate = clock rate
nj 430 sambyt = $a8 ;sample byte
hm 440 samamp = $a9 ;volume for high bit
po 450 ;
ef 460 * = $c000 ;origin of program
ck 470 :file @0:db64.obj,8 ;make obj file
bm 480 ;
ko 490 ;play back 1 bit uncompressed
de 500 ;square wave sample through sid
dk 510 ;
fl 520 ;before calling play set:
cm 530 ;samsta = start of sample
jb 540 ;samlen = length of sample
bp 550 ;samrat = clock cycles
mo 560 ;samamp = volume for 1 bits
hh 570 ;
ic 580 play = *
ja 590 sei ;disable irq
om 600 lda viccr ;read vic control reg
ne 610 and %11101111 ;blank screen mask
hp 620 sta viccr ;blank screen
ja 630 lda icr2 ;clear icr flag bits
in 640 lda #$7f
hb 650 sta icr2 ;nothing will trigger interrupt on cia2
fi 660 lda samrat ;timer latch value
kk 670 sta timea2 ;set timer latch lo
lc 680 ldy #$00 ;zero samsta index
cd 690 sty timea2+1 ;set timer latch hi
ei 700 lda %00010001 ;cia 2 control register bit
       settings:
ad 710 ;
nh 720 ;0 1 start timer
go 730 ;1 0 no timer a output on pb6
km 740 ;2 0 pulse bit 6 one cycle
dk 750 ;3 0 continuous run mode
lh 760 ;4 1 force latch load
ll 770 ;5 0 count machine cycles
di 780 ;6 0 serial input mode
mg 790 ;7 0 tod clock 60 hz
fo 800 ;
gp 810 sta cra2 ;start timer
fc 820 rep1 lda (samsta),y ;repeat get sample byte
bb 830 sta sambyt ;move to 0 page
nd 840 ldx #$08 ;count 8 bits
me 850 rep2 lda icr2 ;load icr 2 irq status
lh 860 and #$01 ;test interrupt bit
gi 870 beq rep2 ;until timer interrupt flag set
nd 880 asl sambyt ;get amp bit
de 890 lda samamp ;get ls bit volume
oc 900 bcs else1 ;if carry flag = 0 then
ga 910 lda #$00 ;set 0 bit volume
ec 920 else1 sta sidvol ;else set 1 bit volume
bb 930 dex ;count down bits left to play
kj 940 bne rep2 ;until no more bits
hc 950 lda samlen+1
mn 960 bne else2 ;if last block then
dm 970 cpy samlen ;check if last byte
ia 980 beq until1
ek 990 else2 iny ;else set up to read next block
ha 1000 bne rep1
dk 1010 inc samsta+1 ;set up to read next block
jn 1020 dec samlen+1 ;count down blocks to play
lg 1030 jmp rep1
op 1040 until1 lda viccr ;until last byte played
eh 1050 ora %00010000 ;unblank screen mask
df 1060 sta viccr ;unblank screen
eo 1070 cli ;enable irq
cp 1080 rts
ae 1090 :end
```

Turbo Charging CP/M with SG Tools Programmer's Tool Box

by Steve Goldsmith

Part 3 of 3

INTRODUCTION
We made it! This final installment of SG Tools will complete our journey through the strange world of C128 CP/M tool box programming in Turbo Pascal. Along the way I hope you learned that CP/M applications on the C128 do not have to be extremely slow, boring and unusable! The SG Tools package gives you a good set of C128 tools to build on and customize as needed.

OVERVIEW
Last time in #33 we found that it is easier to remember variables and constants when we use prefix and/or suffix naming conventions for each module. The VDC screen manager VDCSCMGR.INC has all the primitives needed to control the VDC in a page flipping environment. Most tool boxes contain some sort of fast write procedure that accesses the video hardware directly instead of using the BIOS or other system level interfaces. SG Tools is no exception with its VDCFW.INC module that allows fast string writes without BIOS or BDOS assistance. Window modules like VDCWIN.INC allow the development of modern Graphic User Interfaces (GUI) with Common User Access (CUA) under CP/M.

This time we are going to cover BDOS calls, reading, viewing and selecting CP/M file names, efficiently reading and viewing CP/M text files, building applications and customizing SG Tools.

Here is our latest SG Tools module hierarchy:

```
VDC.INC          PORT.INC      BDOS.INC    HEXSTR.INC  NUMSTR.INC
  VDCCONST.INC     TIMER.INC     KEYIN.INC
    VDCSCMGR.INC                 DIR.INC
      VDCFW.INC                  READTF.INC
        VDCSCRL.INC
        VDCWIN.INC
          DIRSEL.INC
          FVIEW.INC
          VDCMSGB.INC
```

BDOS CALLS
Turbo Pascal has a built-in function called BDos that allows access to CP/M's BDOS file system. The format is:

```
ReturnCode := BDos (FuncNum,Parameter);
```

It is essential to have a CP/M Programmer's guide to fully understand all the BDOS data structures, function numbers, parameters to pass and return codes. The BDOS.INC module defines all the BDOS constants we will be using. You can easily add new function numbers and parameters to BDOS.INC when you need them. I have also included some handy Pascal types for file names, extensions, paths, DMA (direct memory access) buffers and FCBs (file control blocks). The best examples of using the BDos functions are in the DIR.INC module.

READING CP/M DIRECTORIES
It is often very useful to be able to read a directory into memory rather than relying on error prone user input. The DIR.INC directory module reads a directory from the specified drive and user into a double linked list structure. If you are new to dynamically allocated memory structures using the heap then I suggest you read one of the many Turbo Pascal books available on this subject. I would love to explain the concept in full, but it is beyond the scope of this article.

Basically, a linked list allows efficient use of memory because it is allocated at run time as needed. DIR.INC also allows you to specify which files to select using standard CP/M wild cards such as ????????.??? for all files, ????????.COM just for COM files or A???????.ASM for all ASM files that start with A. DIR.INC can be used on any Z80 CP/M box, not just the C128. Examples of using DIR.INC are in the DIRSEL.INC module.

READING CP/M TEXT FILES
Reading a text file into an array of strings is one of the easiest tasks a programmer can do. However, doing it efficiently is another story all together. The problem is that even if you dynamically allocate each line as a Pascal style string you end up wasting the unused portion of the string. For example, let's say you want to write a text file browser with a maximum line length of 80 characters. The first line may be 78 characters, so only 2 bytes are wasted, but what if the next line is just a 5 character word? Well, you just wasted 75 bytes! This method also forces you to have the same maximum amount of 80 character strings in free memory. The way around this problem is by allocating just enough memory to hold each string on a string by string basis. The READTF.INC read text file module does just that! It stores the text lines in a very similar manner to DIR.INC, but instead of fixed length strings we will be using dynamically allocated length strings. READTF.INC can be used on any Z80 CP/M box or even an MS- DOS box for that matter. Examples of using READTF.INC are in the FVIEW.INC module. RTFDOS.PAS on the TC-128 #34 disk is a MS-DOS program that uses READTF.INC to read and display the RTFDOS.PAS file.

RAW KEYBOARD INPUT
It is sometimes desirable to avoid using Turbo Pascal's Read and Readln for input because they both echo characters to the screen. You can see examples of using the Read function in the programs from TC-128 #33. What we need is a function like Commodore BASIC's GET A$ which fetches a character from the keyboard buffer. We do this in CP/M with a simple BDOS call. The GetKey function waits until a key is pressed and returns a byte value representing the key pressed. To convert the value to a character use C := Chr (GetKey). I have also included the keyboard control code constants you get when you press the CONTROL key with and another key. Examples of using KEYIN.INC are in the DIRSEL.INC and FVIEW.INC modules.

SCROLLING VDC WINDOWS
The WINDOWS.PAS program from last time featured a window scrolling text data from left to right. I basically just redrew the whole window each time which was quite slow. The VDCSCRL.INC module allows you to scroll a VDC window up or down one line without redrawing the whole window! I have also provided a

procedure to clear windows. The big increase in speed comes from using the VDC's block copy and block write features. Examples of using VDCSCRL.INC are in the DIRSEL.INC and FVIEW.INC modules.

SELECTING FILE NAMES

Many CP/M programs like SWEEP and NULU are able to load and display CP/M directories. Their file selection or tagging processes are quite outdated compared to today's user interfaces. Most users just want to pick from a list of file names that pop up in a window. The pick list should let users scroll through the file names with the top cursor keys, let them press the RETURN key to select a file name, or the ESCape key to exit. DIRSEL.INC is the file name pick list I just described. DIRSEL.INC reads the current directory by default, but you can easily modify the InitDirSel procedure to read any drive and user. Examples of using DIRSEL.INC are in the QVIEW.PAS program.

VIEWING TEXT FILES

I know that there are a lot of CP/M text viewers and editors, but they all suffer from being too slow on the C128. Most of the time it's not the program's fault, so what can we do? We write a fast VDC text file viewer. FVIEW.INC reads a text file into memory, displays the text in a window and allows you to browse through the file with your top cursor keys until you press the ESCape to exit. The left and right scrolls are considerably slower than the up and down scrolls because I'm redrawing the window each time. You can apply the same techniques found in VDCSCRL.INC to left scrolls, but it doesn't apply to right scrolls. The VDC's block copy will not allow some source and destination overlaps. Block copy moves the first byte fine, but the next source byte is the last destination byte during a one byte right scroll. You end up coping the first byte into all the destination addresses. You can over come this with a buffer that is not in the source or destination addresses, but it is more complicated than VDCSCRL.INC.

MESSAGE BOX

The VDCMSGB.INC module is a simple VDC window that displays a short message. No word wrapping is performed, so long strings will over write areas outside the window.

CONVERTING NUMBERS TO STRINGS

The TC-128 #32 disk had a program called PORT128.PAS that displayed I/O addresses in hexadecimal. It used a module called HEXSTR.INC that converts bytes and integers into a hex string. Likewise, the NUMSTR.INC module converts bytes and integers to right justified decimal strings. Both modules are quite simple and can be found in many string tool boxes.

BUILDING APPLICATIONS

Building bug-free professional looking applications is the dream of every programmer! Making it reality requires a lot of planning before and during the coding process. No, I do not mean spending a month doing a 300 page flow chart, data flow diagrams, etc. Most of the time you can sketch out the features on a piece of paper and start the coding/testing/debugging cycle soon afterwards. If you are approaching a very large and complex application I suggest using a modern CASE (Computer Aided Software Engineering) tool for help. Here are some basic rules I follow when designing applications:

Design reusable modules when possible. I don't know how many times I've seen code that cannot be reused by many different programs. This is usually caused by lazy programming habits. Some programmers make super procedures (mini programs) that combine several tasks instead of taking the time, analyzing the code, breaking each task out and making procedures more modular and reusable. Notice, for example, that VDC specific code is not coded directly into the DIR.INC and READTF.INC modules.

If an algorithm already exists don't recreate the wheel! There is a large amount of Turbo Pascal source available via TUG, GEnie, BBSs, PD mail order, etc. In other words, don't write a general data compression routine from scratch if you have access to LZW or other PD source code. Sometimes you have to port code from other languages, but it still beats writing code from scratch!

Use Assembler (or inline code) only when fast execution speed is important or if there is no other way to accomplish the task. This was a hard habit for me to break since I started Assembler programming on a 5K VIC-20 in 1981. Back then you had to use Assembler code all the time because of memory and speed limitations. I would have to say that I am many times more productive using a structured high level language rather than using Assembler for the same task. SG Tools was written with only four inline machine code routines! Please don't think I'm anti- Assembler because I'm not. It just doesn't make sense to spend the extra time using Assembler in situations where high level code works fine and is more portable.

Portability is a big issue these days. Many programmer's want their code to run on many different platforms without a lot of modification. SG Tools was designed just for the C128, but you could emulate the VDC on a IBM, Amiga or Mac if you wanted to port SG Tools. Actually, it is much easier to make generic routines that work on each individual system. For example, instead of using FWriteVDC call the procedure FastWrite and write the platform specific code for each machine as needed. That way you know that FastWrite will work the same on all platforms. Both the DIR.INC and RTFILE.INC modules are portable to other systems that have Turbo Pascal compilers. See the RTFDOS.PAS program on the TC-128/64 #34 disk for an example of using READTF.INC to read MS-DOS text files.

QUICK VIEW

The best way to teach you how to build an application with SG Tools is to have one that utilizes most of the modules. Quick View (QVIEW.PAS) is a great working example of how to use SG Tools. First, we initialize the VDC and draw the desk top with the Init procedure. Next, we call the Run procedure which allows the user to keep viewing different files until the directory selector is exited with escape. The Done procedure does all the clean up and restores the VDC. The Init, Run and Done model applies to just about any module or program in any language. I use this structure in many SG Tools modules and programs. You may want to add some features to Quick View such as: Go directly to the top, bottom, left or right of the file, speed up left and right scrolls, use a swap file instead of a linked list to create arrays as large as the disk, PETSCII conversion, etc.

CUSTOMIZING SG TOOLS

One important feature of any tool box is the ability to customize it to your own needs. I kept this in mind

before, during and after the development of SG Tools. For example, ANIMATE.PAS uses its own page flip and character display procedures for speed. TCSLIDE.COM on the TC-128 #33 disk uses an event manager, menus, PCX image decoding and VDC bit map display modules I added to SG Tools. When adding your own modules remember to use unique prefix or suffix naming, so you don't end up with duplicates. If you change existing modules ask yourself how it will impact other programs and modules. Other than that, just use your imagination and go wild!

FINAL THOUGHTS

SG Tools is your door to C128 CP/M applications that are not only as good as native mode programs, but even better in areas such as portability, maintainability, development time, etc. The C128 CP/M mode has very few programs that take full advantage of the hardware compared to native 128 mode and GEOS programs. This is one of the reasons I wrote SG Tools because I'm sure that there are a lot of programmers and users who want new uses for CP/M mode and the 128 in general. Think of SG Tools as a application programmer's interface (API) that sits on top of a CP/M engine. Windows for the PC makes use of DOS for many functions, so why not apply the same thing on a smaller scale to SG Tools and CP/M. Most of all enjoy yourself!

If you have any questions or ideas, send me a message on GEnie at address S.GOLDSMITH2. Until next time...

```
PROGRAM NAME: BDOS.INC

SG Tools (C) 1992 Parsec, Inc.

Common BDOS constants


BDOS function numbers

const

  bdosDirectCon    = $06;
  bdosSelectDisk   = $0e;
  bdosSearchFirst  = $11;
  bdosSearchNext   = $12;
  bdosCurrentDisk  = $19;
  bdosSetDMAAddr   = $1a;
  bdosUserCode     = $20;
  bdosSetErrorMode = $2d;

error modes

  bdosRetErrMode     = $ff;
  bdosRetDispErrMode = $fe;
  bdosDefErrMode     = $fd;

direct console I/O modes

  bdosConInStat = $ff;
  bdosConStat   = $fe;
  bdosConIn     = $fd;

BDOS memory locations

  bdosDefaultFCB = $5c;

type

  bdosNameStr = string[8];
  bdosExtStr = string[4];
  bdosPathStr = string[15];
  bdosPDMABuf = bdosDMABuf;
  bdosDMABuf = array[0..127] of byte;
  bdosPFCBRec = bdosFCBRec;
  bdosFCBRec = record
    Drive : byte;
    FileName : array[0..7] of char;
    FileType : array[0..2] of char;
    Extent : byte;
    Reserved1 : byte;
    Reserved2 : byte;
    RecCount : byte;
    Reserved3 : array[0..15] of byte;
    CurrentRec : byte;
    RandomRec : array[0..2] of byte
  end;
```

```
PROGRAM NAME: DIR.INC

SG Tools (C) 1992 Parsec, Inc.

The Directory module reads a CP/M directory into a double
linked list structure.
No memory allocation error trapping.  READTF.INC uses such
error trapping.


type

  dirPRec = dirRec;
  dirRec = record
    FileName : string [12];
    Prev,
    Next : dirPRec;
  end;

var

  dirCurDrive,              current drive
  dirCurUser : byte;        current user
  dirError,                 error codes
  dirRecs : integer;        records read
  dirDMAPtr : bdosPDMABuf;  dma buffer
  dirFCB : bdosFCBRec       default fcb
  absolute bdosDefaultFCB;
  dirFirstPtr,              first record pointer
  dirCurPtr : dirPRec;      current record pointer

procedure InitDir (Drive, User : byte;
                   WildCard : bdosPathStr);

var

  I : byte;

begin
  dirRecs := 0;
  dirDMAPtr := nil;             init pointers
  dirFirstPtr := nil;
  dirCurPtr := nil;
  dirError :=                   set bdos error mode to return
error
  BDos (bdosSetErrorMode,bdosRetErrMode);
  dirCurDrive :=
  BDos (bdosCurrentDisk);       save current drive and user
  dirCurUser :=
  BDos (bdosUserCode,$ff);
  dirError :=
  BDos (bdosSelectDisk,Drive); select drive to read
  if dirError = 0 then
  begin
    dirError :=
    BDos (bdosUserCode,User);  select user to read
    dirFCB.Drive := 0;         use default drive
```

```
    for I := 0 to 7 do          set wild card
      dirFCB.FileName[I] := WildCard[I+1];
    for I := 0 to 2 do
      dirFCB.FileType[I] := WildCard[I+10];
    New (dirDMAPtr)             allocate dma buffer
  end
end;

procedure DoneDir;

var

  TempPtr : dirPRec;

begin
  if dirDMAPtr <> nil then
    Dispose (dirDMAPtr);      dispose dma buffer
  if dirFirstPtr <> nil then dispose linked list structure
  begin
    dirCurPtr := dirFirstPtr;
    repeat
      TempPtr := dirCurPtr.Next;
      Dispose (dirCurPtr);
      dirCurPtr := TempPtr
    until dirCurPtr = nil
  end;
  dirError :=                 select previous drive
  BDos (bdosSelectDisk,dirCurDrive);
  dirError :=                 select previous user
  BDos (bdosUserCode,dirCurUser);
  dirError :=                 set bdos error mode to default
  BDos (bdosSetErrorMode,bdosDefErrMode)
end;

function PackName (U : bdosPathStr) : bdosPathStr;

var

  I : byte;

begin
  repeat                delete spaces from file name
    I := Pos (' ',U);
    if I <> 0 then
      Delete (U,I,1)
  until I = 0;
  PackName := U
end;

procedure ReadDir;
var
  FCBPtr : bdosPFCBRec;
  TempPtr : dirPRec;

begin
  dirError :=                 set new dma address
  BDos (bdosSetDMAAddr,Addr (dirDMAPtr));
  dirError :=                 read fisrt dir fcbs
  BDos(bdosSearchFirst,Addr (dirFCB));
  if dirError <> $ff then
  begin
    New (dirCurPtr);          allocate first dir record
    dirCurPtr.Prev := nil;    first record's prev is nil
    dirFirstPtr := dirCurPtr;
    while dirError <> $ff do  read and allocate remaining records
    begin
      dirRecs := dirRecs+1;
      FCBPtr :=               pointer to fcb in dma buffer
      Ptr (Addr (dirDMAPtr)+dirError shl 5);
      dirCurPtr.FileName :=   make file name
      FCBPtr.FileName+'.'+FCBPtr.FileType;
      dirError :=             bdos search for next dir entry
      BDos (bdosSearchNext);
      if dirError <> $ff then
      begin
        TempPtr := dirCurPtr;       save cur record pointer
        New (dirCurPtr);            allocate new record
        TempPtr.Next := dirCurPtr; set links
        dirCurPtr.Prev := TempPtr
      end
    end;
    dirCurPtr.Next := nil  last record's next pointer is nil
  end;
  dirError :=               set default dma address
  BDos (bdosSetDMAAddr,bdosDefaultFCB)
end;

function GetFileName : bdosPathStr;

begin
  if dirCurPtr <> nil then
  begin
    GetFileName := dirCurPtr.FileName; get current file name
    dirCurPtr := dirCurPtr.Next        set up for next record
  end
end;
```

PROGRAM NAME: DIRSEL.INC

SG Tools (C) 1992 Parsec, Inc.

Directory Selector uses a VDC window to select a CP/M file
from the currently
logged disk.

```
const

  dselXSize   = 16; window size
  dselYSize   = 15;
  dselCurSize = 14;

var

  dselX1, dselY1, dselX2, dselY2,
  dselCY, dselAttr : byte;
  dselCurPtr : dirPRec;

procedure DispDir;

var

  X, Y : byte;

begin               fill window with file names
  X := dselX1+2;
  Y := dselY1+2;
  repeat
    FWriteVDC (X,Y,dselAttr,GetFileName);
    Y := Y+1
  until (dirCurPtr = nil) or
  (Y = dselY2-2)
end;

procedure DispCur (A : byte);

begin                draw cursor
  FillAttrVDC (dselX1+1,dselCY,dselCurSize,A);
end;

procedure ScrollUpDir;

begin               scroll window up
  ScrollUpVDC (dselX1+1,dselY1+3,
  dselX2-1,dselY2-3);
  FWriteVDC (dselX1+2,dselY2-3,
```

```
  dselAttr,dselCurPtr.FileName)
end;

procedure ScrollDownDir;

begin               scroll window down
  ScrollDownVDC (dselX1+1,dselY1+2,
  dselX2-1,dselY2-4);
  FWriteVDC (dselX1+2,dselY1+2,
  dselAttr,dselCurPtr.FileName)
end;

procedure MoveCurUp;

begin            move cursor up
  DispCur (dselAttr);
  if (dselCurPtr.Prev <> nil) and
  (dselCY > dselY1+1) then
  begin
    dselCurPtr := dselCurPtr.Prev;
    if dselCY = dselY1+2 then
      ScrollDownDir
    else
      dselCY := dselCY-1
  end;
  DispCur (vdcRvsVid+dselAttr);
  FlipPageVDC
end;

procedure MoveCurDown;

begin            move cursor down
  DispCur (dselAttr);
  if (dselCurPtr.Next <> nil) and
  (dselCY < dselY2-2) then
  begin
    dselCurPtr := dselCurPtr.Next;
    if dselCY = dselY2-3 then
      ScrollUpDir
    else
      dselCY := dselCY+1
  end;
  DispCur (vdcRvsVid+dselAttr);
  FlipPageVDC
end;

function SelFileName : bdosPathStr;

var

  C : byte;

begin
  repeat
    C := GetKey;
    case C of
      kbCtrlE : MoveCurUp;
      kbCtrlX : MoveCurDown
    end
  until (C = kbCtrlM) or (C = kbEsc);
  if C = kbCtrlM then
    SelFileName := PackName (dselCurPtr.FileName)
  else
    SelFileName := ''
end;

procedure InitDirSel (X1,Y1,A : byte;
                      WildCard : bdosPathStr);

begin
  InitDir (                read dir of default drive/user
  BDos (bdosCurrentDisk),
  BDos (bdosUserCode,$ff),
  WildCard);
  if dirError = 0 then
  begin
    ReadDir;
    if dirRecs > 0 then
    begin
      dirCurPtr := dirFirstPtr;
      dselCurPtr := dirFirstPtr;
      dselX1 := X1;
      dselY1 := Y1;
      dselX2 := X1+dselXSize-1;
      dselY2 := Y1+dselYSize-1;
      dselCY := Y1+2;
      dselAttr := A;
      DrawWinVDC (dselX1,dselY1,dselX2,dselY2,dselAttr,
      Chr (dirCurDrive+65)+Chr (dirCurUser+48)+':');
      FWriteVDC (dselX1+1,dselY2-1,
      vdcAltChrSet+vdcWhite,IntStr (dirRecs)+' Files');
      DispDir;
      DispCur (dselAttr+vdcRvsVid)
    end
  end
end;

procedure DoneDirSel;

begin
  DoneDir
end;


PROGRAM NAME: FVIEW.INC

SG Tools (C) 1992 Parsec, Inc.

File View allows you to view a text file in a VDC window.


var

  fvX1, fvY1, fvX2, fvY2, fvCol,
  fvXSize, fvYSize, fvAttr : byte;
  fvCurPtr, fvTopPtr, fvBtmPtr : rtfPLine;

return line with current column offset

function DispLine (Old : rtfStr) : rtfStr;

begin
  if Length (Old) >= fvCol then
    Old := Copy (Old,fvCol,fvXSize-2)
  else
    Old := '';
  DispLine := Old
end;

fill window with lines starting at current line pointer

procedure DispWin;

var

  X, Y : byte;
  S : rtfStr;

begin
  Y := fvY1+1;
  X := fvX1+1;
  repeat
    S := DispLine (GetLine);
    if S <> '' then
```

```
      FWriteVDC (X,Y,fvAttr,S);
    Y := Y+1
  until (rtfCurPtr = nil) or
        (Y = fvY2)
end;

procedure MoveUpLine;

begin
  if fvTopPtr.Prev <> nil then
  begin
    fvTopPtr := fvTopPtr.Prev;
    fvBtmPtr := fvBtmPtr.Prev;
    fvCurPtr := fvTopPtr;
    ScrollDownVDC (fvX1+1,fvY1+1,
    fvX2-1,fvY2-2);
    FillDispVDC (fvX1+1,fvY1+1,
    fvXSize-2,32);
    FWriteVDC (fvX1+1,fvY1+1,
    fvAttr,DispLine (fvCurPtr.Line));
    FlipPageVDC
  end
end;

procedure MoveDownLine;

begin
  if fvBtmPtr.Next <> nil then
  begin
    fvTopPtr := fvTopPtr.Next;
    fvBtmPtr := fvBtmPtr.Next;
    fvCurPtr := fvBtmPtr;
    ScrollUpVDC (fvX1+1,fvY1+2,
    fvX2-1,fvY2-1);
    FillDispVDC (fvX1+1,fvY2-1,
    fvXSize-2,32);
    FWriteVDC (fvX1+1,fvY2-1,
    fvAttr,DispLine (fvCurPtr.Line));
    FlipPageVDC
  end
end;

procedure MoveLeftLine;

begin
  if fvCol < rtfMaxStr then
  begin
    rtfCurPtr := fvTopPtr;
    fvCol := fvCol+1;
    ClearWinVDC (fvX1+1,fvY1+1,fvX2-1,fvY2-1,32);
    DispWin;
    FlipPageVDC
  end
end;

procedure MoveRightLine;

begin
  if fvCol > 1 then
  begin
    rtfCurPtr := fvTopPtr;
    fvCol := fvCol-1;
    ClearWinVDC (fvX1+1,fvY1+1,fvX2-1,fvY2-1,32);
    DispWin;
    FlipPageVDC
  end
end;

procedure ViewFile;

var

  C : byte;

begin
  repeat
    C := GetKey;
    case C of
      kbCtrlE : MoveUpLine;
      kbCtrlX : MoveDownLine;
      kbCtrlD : MoveLeftLine;
      kbCtrlS : MoveRightLine
    end
  until (C = kbCtrlM) or (C = kbEsc)
end;

procedure InitViewFile (X1,Y1,X2,Y2,A : byte;
                        FileName : bdosPathStr);

begin
  fvX1 := X1;
  fvY1 := Y1;
  fvX2 := X2;
  fvY2 := Y2;
  fvXSize := X2-X1+1;
  fvYSize := Y2-Y1+1;
  fvCol := 1;
  fvAttr := A;
  InitReadFile (FileName);
  if rtfError = 0 then
  begin
    ReadFile;
    DrawWinVDC (fvX1,fvY1,fvX2,fvY2,fvAttr,FileName);
    rtfCurPtr := rtfFirstPtr;
    fvTopPtr := rtfFirstPtr;
    DispWin;
    fvCurPtr := rtfCurPtr;
    fvBtmPtr := rtfCurPtr.Prev;
    FlipPageVDC
  end
end;

procedure DoneViewFile;

begin
  DoneReadFile
end;
```

```
PROGRAM NAME: KEYIN.INC

SG Tools (C) 1992 Parsec, Inc.

The Keyboard Input module allows raw CP/M input.


const

CP/M keyboard mapping

  kbCtrlA  = 1;  kbCtrlB  = 2;  kbCtrlC  = 3;  kbCtrlD  = 4;  kbCtrlE  = 5;
  kbCtrlF  = 6;  kbCtrlG  = 7;  kbCtrlH  = 8;  kbCtrlI  = 9;  kbCtrlJ  = 10;
  kbCtrlK  = 11; kbCtrlL  = 12; kbCtrlM  = 13; kbCtrlN  = 14; kbCtrlO  = 15;
  kbCtrlP  = 16; kbCtrlQ  = 17; kbCtrlR  = 18; kbCtrlS  = 19; kbCtrlT  = 20;
  kbCtrlU  = 21; kbCtrlV  = 22; kbCtrlW  = 23; kbCtrlX  = 24; kbCtrlY  = 25;
  kbCtrlZ  = 26; kbEsc    = 27; kbCtrlFS = 28; kbCtrlGS = 29; kbCtrlRS = 30;
  kbCtrlUS = 31; kbDel    = 127;

function GetKey : byte;

begin
  GetKey := BDos (bdosDirectCon,bdosConIn)
end;
```

```
PROGRAM NAME: NUMSTR.INC

SG Tools (C) 1992 Parsec, Inc.

The number string module converts bytes and integers to strings


type

  StrByte = string[3];
  StrInteger = string[6];

function ByteStr (B : byte) : StrByte;

var

  TempStr : StrByte;

begin
  Str (B:3,TempStr);
  ByteStr := TempStr
end;

function IntStr (I : integer) : StrInteger;

var

  TempStr : StrInteger;

begin
  Str (I:6,TempStr);
  IntStr := TempStr
end;


PROGRAM NAME: VDCMSGB.INC

SG Tools (C) 1992 Parsec, Inc.

VDC message box.  No line wrapping inside window.


const

  mbXSize = 40;
  mbYSize = 5;

procedure MsgBox (X,Y,W,T : byte;
                  Msg : fwMaxStr);

begin
  DrawWinVDC (X,Y,X+mbXSize,Y+mbYSize,W,'Information');
  FWriteVDC (X+1,Y+2,T,Msg)
end;


PROGRAM NAME: READTF.INC

SG Tools (C) 1992 Parsec, Inc.

The Read Text File module reads a CP/M text file into a double linked list
structure.  Each text line is saved in a line record with a poitner to the
string.  Only enough memory is allocated to hold each string, so there is
no waste!


const

  rtfMinFree = 1024; min free memory
  rtfMaxStr  = 255;  max string size

type

  rtfPStr = tfStr;
  rtfStr = string[rtfMaxStr];
  rtfPLine = tfLine;
  rtfLine = record
    Line : rtfPStr;
    Prev,
    Next : rtfPLine;
  end;

var

  rtfError,               errors
  rtfLines : integer;     line count
  rtfFirstPtr,            first record pointer
  rtfCurPtr : rtfPLine;   current record pointer
  rtfFile : text;         text file to read

procedure InitReadFile (FileName : bdosPathStr);

begin
  rtfError := 0;    initilize vars
  rtfLines := 0;
  rtfFirstPtr := nil;
  rtfCurPtr := nil;
  Assign (rtfFile, FileName);
  $I- Reset (rtfFile); $I+
  rtfError := IoResult
end;

procedure DoneReadFile;

var

  TempPtr : rtfPLine;

begin
  $I- Close (rtfFile); $I+
  rtfError := IoResult;
  if rtfFirstPtr <> nil then dispose linked list structure
  begin
    rtfCurPtr := rtfFirstPtr;
    repeat
      TempPtr := rtfCurPtr.Next;
      FreeMem (rtfCurPtr.Line,
      Length (rtfCurPtr.Line)+1);
      Dispose (rtfCurPtr);
      rtfCurPtr := TempPtr
    until rtfCurPtr = nil
  end;
end;

procedure ReadFile;
var
  TempStr : rtfStr;
  TempStrPtr : rtfPStr;
  TempPtr : rtfPLine;

begin
  $I- Readln (rtfFile,TempStr); $I+
  rtfError := IoResult;
  if (rtfError = 0) and
  (not eof (rtfFile)) then
  begin
    rtfLines := 1;
    New (rtfCurPtr);              set up first record
    rtfCurPtr.Prev := nil;       first records prev is nil
    GetMem (TempStrPtr,           allocate just string length + 1
    Length (TempStr)+1);
    TempStrPtr := TempStr;
    rtfCurPtr.Line := TempStrPtr;
    rtfFirstPtr := rtfCurPtr;
```

```
    while (rtfError = 0) and      read in rest of file
    (not eof (rtfFile)) and
    (Hi (MemAvail) > Hi (rtfMinFree)) do
    begin
      $I- Readln (rtfFile,TempStr); $I+
      rtfError := IoResult;
      if rtfError = 0 then
      begin
        rtfLines := rtfLines+1;  add line record
        New (TempPtr);
        rtfCurPtr.Next := TempPtr;
        TempPtr.Prev := rtfCurPtr;
        GetMem (TempPtr.Line,    allocate string length + 1
        Length (TempStr)+1);
        TempPtr.Line := TempStr;
        rtfCurPtr := TempPtr
      end
    end;
    rtfCurPtr.Next := nil        last record's next is nil
  end
end;

function GetLine : rtfStr;

begin
  if rtfCurPtr <> nil then
  begin
    GetLine := rtfCurPtr.Line; get current line
    rtfCurPtr := rtfCurPtr.Next set up to read next line
  end
end;


PROGRAM NAME: VDCSCRL.INC

SG Tools (C) 1992 Parsec, Inc.

VDC window scroller using block copies and fills


procedure ScrollUpVDC (X1, Y1, X2, Y2 : byte);

var

  Y : byte;
  DispOfs : integer;

begin
  for Y := Y1 to Y2 do
  begin
    DispOfs := vdcSettings.DispMem+Y*vdcScrHorz+X1;
    CopyMemVDC (DispOfs,DispOfs-vdcScrHorz,X2-X1+1)
  end
end;

procedure ScrollDownVDC (X1, Y1, X2, Y2 : byte);

var

  Y : byte;
  DispOfs : integer;

begin
  for Y := Y2 downto Y1 do
  begin
    DispOfs := vdcSettings.DispMem+Y*vdcScrHorz+X1;
    CopyMemVDC (DispOfs,DispOfs+vdcScrHorz,X2-X1+1)
  end
end;

procedure ClearWinVDC (X1, Y1, X2, Y2, C : byte);

var

  Y : byte;
  DispOfs : integer;

begin
  for Y := Y1 to Y2 do
  begin
    DispOfs := vdcSettings.DispMem+Y*vdcScrHorz+X1;
    FillMemVDC (DispOfs,X2-X1+1,C)
  end
end;


PROGRAM NAME: RTFDOS.PAS

SG Tools (C) 1992 Parsec, Inc.

This is a short demo for MS-DOS using READTF.INC to read and
display
RTFDOS.PAS in the current DOS directory.
Compiled with TP 6.0 for MS-DOS.

program RTF
Test;

$A+,B-,D-,E-,I+,N-,R-,S-,V-

$I READTF.INC

procedure DisplayFile;

begin
  repeat
    Writeln (GetLine);
  until rtfCurPtr = nil
end;

procedure Init (F : string);

begin
  InitReadFile (F);
  if rtfError = 0 then
  begin
    Writeln ('Reading '+F+'...');
    ReadFile;
    rtfCurPtr := rtfFirstPtr
  end
  else
    Writeln ('Error reading '+F);
end;

procedure Run;

begin
  if rtfError = 0 then
  begin
    DisplayFile;
    Writeln;
    Writeln (rtfLines:6,' lines read.')
  end
end;

procedure Done;

begin
  DoneReadFile
end;

begin
  Init ('RTFDOS.PAS');
  Run;
  Done;
end.
```

```
PROGRAM NAME: QVIEW.PAS

SG Tools (C) 1992 Parsec, Inc.

Quick View is a VDC text file viewer.  Up and down scrolls are about 20 times
faster than CP/M BIOS!  Left and Right scrolls could be improved though.


program QuickView;

$B-,C-,R-,U-,V-

SG Tools include files

$I NUMSTR.INC
$I BDOS.INC
  $I KEYIN.INC
  $I DIR.INC
  $I READTF.INC
$I PORT.INC
  $I VDC.INC
    $I VDCCONST.INC
    $I VDCSCMGR.INC
    $I VDCFW.INC
    $I VDCWIN.INC
    $I VDCMSGB.INC
    $I VDCSCRL.INC
      $I DIRSEL.INC
      $I FVIEW.INC

application specific code

const

  appScrColor     = vdcDarkGreen;
  appWallPaperChr = 137;

var

  appMsgBoxColor,
  appWallPaperColor,
  appTextColor : byte;

procedure DrawDeskTop;

begin
  ClrScrVDC (appWallPaperChr);    clear screen
  ClrAttrVDC (appWallPaperColor); clear screen attributes
  FWriteVDC (0,0,appMsgBoxColor+vdcRvsVid,
  '      Quick View 1.0 by SG - (C) 1992 Parsec, Inc. - All Rights Reserved
');
end;

function ReadFileName : bdosPathStr;

begin
  ReadFileName := '';
  MsgBox (20,9,appMsgBoxColor,appTextColor,
  'Reading directory...');
  FlipPageVDC;
  DrawDeskTop;
  MsgBox (20,2,appMsgBoxColor,appTextColor,
  'Use top cursor keys, Return and Esc');
  InitDirSel (1,2,appMsgBoxColor,'????????.???');
  FlipPageVDC;
  if dirRecs > 0 then     no recs mean error or empty
    ReadFileName := SelFileName;
  DoneDirSel
end;

procedure BrowseFile (F : bdosPathStr);

begin
  MsgBox (20,11,appMsgBoxColor,appTextColor,
  'Reading '+F+'...');
  FlipPageVDC;
  DrawDeskTop;
  InitViewFile (1,2,77,22,appMsgBoxColor,F);
  if rtfError = 0 then
    ViewFile;
  DoneViewFile
end;

procedure Run;

var

  F : bdosPathStr;

begin
  repeat  view files until user esc dir select
    F := ReadFileName;
    if F <> '' then
      BrowseFile (F)
  until F = ''
end;

procedure Init;

begin
  appMsgBoxColor := vdcAltChrSet+vdcBlack; set app colors
  appTextColor := vdcAltChrSet+vdcWhite;
  appWallPaperColor := vdcAltChrSet+vdcLightGreen;
  InitVDC;                          fire up screen manager
  SetCursorVDC (0,0,vdcCurNone); turn cursor off
  FlipPageVDC;                      now fast writes go to
non-viewable page
  SetScrColVDC (appScrColor,appScrColor);  set app screen
color
  DrawDeskTop;                      draw desk top
  FlipPageVDC
end;

procedure Done;

begin
  ClrScrVDC (32);         prepare screen for return to cp/m
  ClrAttrVDC (vdcAltChrSet+vdcWhite);
  FlipPageVDC;
  DoneVDC                 we are finished with the screen
manager
end;

begin
  Init;
  Run;
  Done
end.
```

Digital Audio On Your Commodore 128

By Mike Neus

When the Commodore 64 was introduced in 1982, it brought with it the most sophisticated sound synthesizer to date. The Sound Interface Device (SID) features three voices, four octave range, and even programmable wave forms and filters. A far cry from the primitive beeps and clicks offered from other computers.

Some time ago, programmers discovered something else that surprised all of us. A SID chip can be used to recreate digital audio. Since that time, many groups have turned the 64 into a mini recording studio. Sound and graphic "demos" (sometimes called digi's) became their calling cards, leaving a trail of music and awe for the rest of the world.

Digital audio has pretty much remained in the C-64's domain with noone taking advantage of the C-128's increased memory and speed, until now.

With that in mind and a lot of spare time, I set out to make a program that will record and playback digital audio. After a brisk two months of development, a program called ZOUNDS! was released to let the 128 do what the 64 has done for years.

Ok, ok, ok. Before you get too excited, remember the SID chip can only reproduce digital audio. So...how do we record it? To answer this question, we must first answer what digital audio is. Afterwards, you can build the simple circuit in this article to record and play back your own sounds.

As the name implies, digital audio is nothing more than a series of digital numbers. This is the basis of how computers work. You may have also heard the term "analog" to describe our world. Sound is an example of an analog signal. At any given moment in time, it has a measurable amplitude. Graphs can be made that approximate the appearance of sound as it travels through the air (Figure 1).

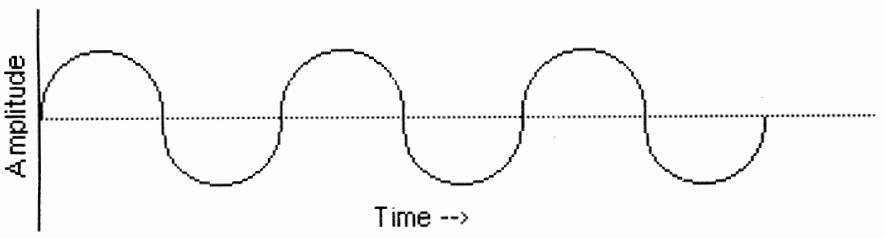


Figure 1: Graphical representation of a sound wave in air.

If we are to record the sound on a record or cassette tape, the imprinted signal closely matches the graphical interpretation in Figure 1. We can also record the signal by assigning numeric values representing amplitude at a given time. If this is done quickly and repetitively, the numbers begin to approximate the analog signal (Figure 2)

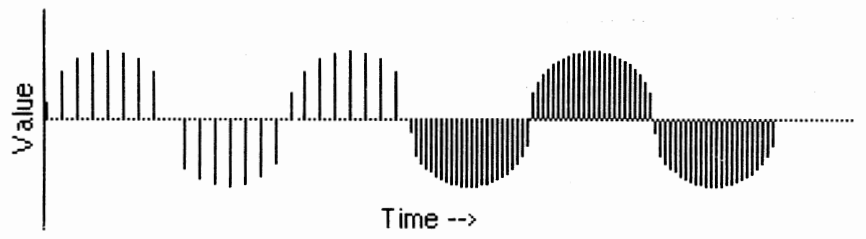


Figure 2: Approximation of Figure 1 by using numbers. Since a number is readily converted into binary form, an approximation of the signal can be stored in digital memory. Converting an analog signal into digital format is called digital sampling.

Figure 2 illustrates two important principles. Note the first half of the figure has less samples (lines) than the second. Also note the more lines we have, the more accurately the signal is approximated, particularly where the curve is steep. Therefor, increasing the sample rate will give a better recording. Also note that to sample faster, you will need a faster computer and additional memory to hold more samples.

Another important principle is resolution. This is best explained using Figure 3.

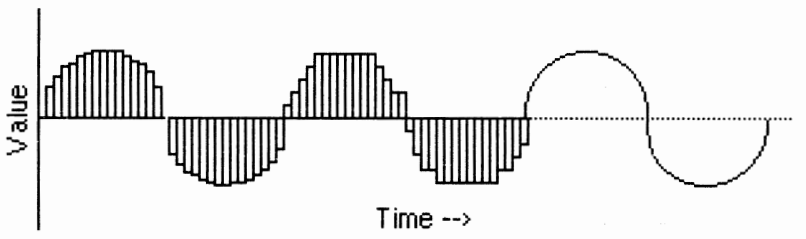


Figure 3: The affect of resolution in sampling

The first third of the graph assumes infinite resolution. Notice the graph closely matches the analog signal in the last third of the graph. Now, suppose the resolution is dropped to five numbers above or below the middle line. When the signal is sampled, the nearest value must be used, resulting in a stair step pattern.

The effects of resolution and sampling rate are well defined. One rule is the sample rate must be at least twice the highest frequency you wish to record. In the case of compact disc, a sample rate of 44.1KHz was used to ensure the highest audible frequency of 20KHz could be recorded.

The above process may sound a bit complicated, but fortunately a number of integrated circuits are available to sample analog signals into a computer. An Analog to Digital Converter (ADC) is designed to provide an interface for the computer into the analog world. Voltages in an audio signal are thus easy to convert into digital information.

To convert the digital pattern back into an audible analog waveform, a Digital to Analog Converter (DAC) is required. A DAC is needed convert a digital number back into an analog voltage. This is the exact opposite function of an ADC. In the case of the ZOUNDS! project, we will use the SID as a DAC.

Technically, the SID has no DAC. However, it does have something that can be adapted. The volume control register at 54296 ($D418) "features" turn on transients which can be heard as a pop when the volume is changed.

The greater the change, the louder the pop. This closely approximates what a DAC does. Because the volume register is 4 bits (0 through 15), we are limited to four bits of resolution. With these principles and limitations in mind, we are now ready to design the ADC circuit.

For the ZOUNDS! project, I chose to use a Harris CA3306. This particular circuit is plug compatible with the RCA CA3306 and Micro Power Systems MP7682. The Harris version is available in a number of suffixes. Each version varies slightly in performance. The cheapest version (about $10) is adequate.

Most any converter will work for this project, but the driving factors for using this particular ADC were its low parts requirement and quick speed. The CA3306 is a six bit converter so the two least significant bits will be ignored. Experimenters may wish to try the CA3304, a slightly cheaper four bit version.

There is one more problem that must be addressed. A glance at Figures 1-3 shows sound waves consist of positive and negative voltages. Unfortunately, the ADC will only have access to ground and +5 volts. This means the converter is restricted to converting only between 0 and 5 volts.

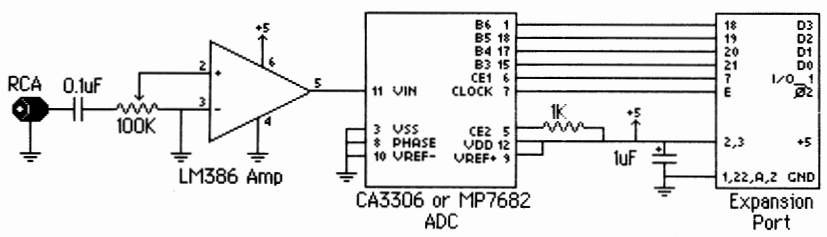
What is needed is a way to shift the input signal to center around 2.5 volts instead of ground. Fortunately, there is a simple and inexpensive solution. The LM386 amplifier features an output stage that adjusts the output signal to be centered around 1/2 the supply voltage (ie: 2.5 volts with a five volt supply). With this condition imposed, the ADC will now easily convert both positive and negative voltage swings.

The ADC will interface directly to the C128 expansion bus. At this time, I must point out that you must proceed carefully. Any miswiring can result in permanent damage. Neusoft Software Systems, the author (Mike Neus), Parsec Inc., or Twin Cities 128 Magazine shall not be responsible or liable for any damage.

You will need the following parts to build the audio sampler:

1) Harris or RCA CA3306 -or- Micro Power MP7682 flash ADC
2) LM386 low power audio amplifier
3) 0.1uF ceramic capacitor
4) 1uF electrolytic capacitor rated for 10 volts or more
5) 100K potentiometer
6) RCA jack
7) A Commodore compatible prototype board
8) Wire, solder, sockets etc to suit your building habits

(This is the "original" circuit, the following (2nd) circuit is a newer version that Mike gave to us shortly before press time so we are including both!)



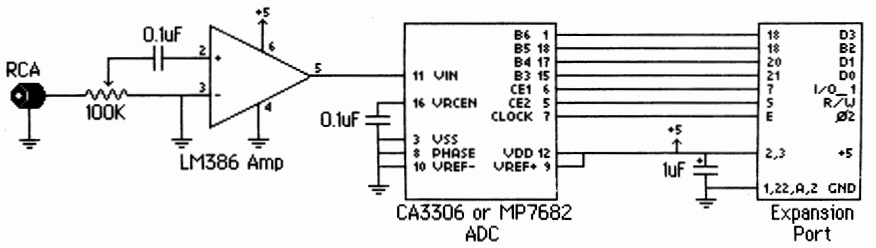


Figure 4: Schematic of the ZOUNDS! sound sampler. Use this newer one as it is suppose to produce cleaner sound samples.

The ADC is visible to the 128 as a memory device using the I/O 1 control line on the expansion port. When the ADC is connected to I/O 1 and the 8502 data bus, it is accessible by reading location 56832 ($DE00).

This sampler is compatible with the ZOUNDS! software available through the public domain. If you do not have ZOUNDS!, I have included a scaled down version of ZOUNDS! (called TCZOUNDS!) to help you get started. TCZOUNDS! includes capture and playback routines, plus a sound monitor routine. The following information is intended mainly for TCZOUNDS! users. ZOUNDS! users should refer to the documentation included with the program for information and instructions.

All routines use the CIA hardware timers to generate the sample clock. This allows compatibility with both 2MHz and 1MHz modes. It also means the VIC screen does not have to be blanked if you are using 40 columns. The routines also take over the non maskable interrupt (NMI) routine for their own use. It assumes that any NMIs come from the RESTORE key, so be sure to turn off your modem or other attachments that use an NMI.

TCZOUNDS! limits the sample to RAM bank 0 at $0AFD through $CFFF. Memory locations $0AFD - $0AFF, hold registers special to TCZOUNDS! files. Two registers are taken care of automatically: Ending RAM bank ($0AFE) and ending page ($0AFF). The ending page points to the last page of memory that holds data. The ending RAM bank indicates which bank the ending page is in. Normally, the ending RAM bank is 0 for bank 0 or non zero for bank 1. In TCZOUNDS!, the ending RAM bank register will be automatically set to zero.

The third parameter, sample speed ($0AFD) will require manual entry with the machine language monitor. Valid numbers are 15 - 255 ($0F - $FF) at 2MHz CPU speed. At 1MHz, the lower limit will be 30 ($1E). To determine the sample rate, the formula is 1x106/(register value). The quickest sample speed is determined by software overhead. Smaller values will not result in faster sample speeds. The fastest speed is 66.7KHz, about 66% faster than the sampling speed of compact disc!

As a default, try 100 ($64), or 10KHz. 100 offers an excellent trade off between memory usage (you will have about 10 seconds worth), and sound quality. A high sample rate will give better sound quality (refer to Figure 1), but as can be expected, the maximum sample length will decrease.

The sample will be stored sequentially starting at $0B00. Since the resolution is fixed at four bits, two samples are stored in each byte. The first sample is stored in the upper nibble, while the second is placed

in the lower. Sampling will stop when you halt it or if memory limits are exceeded.

This exact same format is used by ZOUNDS!. This means files you make with TCZOUNDS! will be compatible with ZOUNDS! should you chose to upgrade. Likewise, ZOUNDS! files will be compatible with TCZOUNDS!. Unlike TCZOUNDS!, ZOUNDS! will have access to both RAM banks, and will automatically set each register.

To use TCZOUNDS!, entry into the C128 machine language monitor is required. Because TCZOUNDS! uses BASIC storage space, care should be taken to avoid the BASIC interpreter after a sound is recorded or loaded. For this reason, use the monitor's J command instead of G. J will always return to the monitor but G will (usually) return to BASIC. To end a TCZOUNDS! session, reset the computer to insure BASIC will function properly.

Use the monitor (not to be confused with the C128s ML monitor) routine (J 0E000 or 'M' from ZOUNDS! menu) to verify the ZOUNDS! interface is functional. Plug in the sampler and an audio source--line level signals from a stereo component work well. Adjust the potentiometer while you are listening to the sound you intend to sample. If the sound is distorted, turn the potentiometer to decrease the volume. Ideally, the volume should be as high as possible before distortion is audible. Press the restore key to leave the monitor.

The distortion is generated when the voltage entering the ADC wants to exceed the supply voltages of 0 and 5 volts. The signal is clipped, looking something like a square wave. It won't damage anything, but it should be avoided.

You may have noticed that in order to hear the sound on your monitor, you had to turn the volume way up. If you did, you have an "improved" SID chip. It seems Commodore figured out how to reduce the SIDs transient characteristics, reducing its effectiveness as a DAC. A soft SID chip is normal and does not indicate a problem. If you feel uncomfortable turning your monitor up all the way, the included BASIC program "LOUD" will make the SID louder. Run this program before entering ZOUNDS! or TCZOUNDS!.

When the sampler is set up, you can record your sound (J 0E017 or 'R' from ZOUNDS! menu). Sampling will begin as soon as the routine is entered and will end when you press the restore key or memory runs out. If you care to save your sound to disk, use the machine language monitor to examine memory location $0AFF (ending page number). Add two zeros to the end of this number to get the ending address, then save file:

```
S "0/filename",device,0AFD,address
```

The 0/ prefix indicates it is a ZOUNDS! file compatible with any ZOUNDS! player.

Use the playback routine to hear what you have recorded (J 0E04E or 'P' from ZOUNDS! menu). For an interesting effect, try adjusting the speed register ($03FD) before playback. A smaller number will speed the sound up. Larger values will slow the sound down.

To load a pre-recorded sound for playback, use the monitor load command:

```
L "0/filename",device
```

This is a small smattering of what can be done with digital audio. With the sound now stored in a computer memory, the possibility for digital signal processing on the 128 is now a reality. Echoes are one possibility I've heard on a 64. Try flipping the sound backwards for fun. Try editing patches together for your own custom song.

ZOUNDS! will not do all this, but it is an important step towards it. If you would like a complete copy of the ZOUNDS! software, it is available through GEnie, Parsec Inc., or local BBSs. It is also available by sending three blank 5 1/4" disks and $5.00 to:

Neusoft Software Systems
Attn: Mike Neus
396 S.W. Parkway #2725
Lewisville, TX 75067

```
PROGRAM NAME = KICK

10 *= $E000
20 .d kick tczounds.bin
30 .f tczounds.src

PROGRAM NAME = TCZOUNDS.SRC

dk 1000 ; filename = tczounds.src
ha 1010 ;*********************************
ke 1020 ;* tczounds!--special twin       *
lo 1030 ;* cities zounds! version.       *
ac 1040 ;* copyright 1992, neusoft       *
md 1050 ;* software/parsec inc.  refer   *
po 1060 ;* to "Digital Audio For Your    *
ie 1070 ;* commodore 128" for            *
ii 1080 ;* instructions.                 *
mb 1090 ;*********************************
il 1100 ;
nf 1110 ;*********************************
af 1120 ;* define labels                 *
oj 1130 ;*********************************
ld 1140 ;
id 1150 column = $ee;          columns on current display
jm 1160 istore = $fb;          temporary storage/vector
eb 1170 restor = $fe;          flag for restore key
ne 1180 inmi   = $0318;        nmi indirect vector
ij 1190 cnmi   = $0a70;        location to copy nmi routine
ch 1200 strbot = $0b00;        bottom of sound storage
lg 1210 strtop = $d000;        top of sound storage
pk 1220 scroly = $d011;        vic ii y smooth scroll & control register
nj 1230 clkrte = $d030;        cpu clock rate control register
ki 1240 sigvol = $d418;        sid volume register
bo 1250 mmurcr = $d506;        mmu ram configuration register
ec 1260 d1t1l  = $dc04;        cia#1 timer register, lo byte
aj 1270 d1t1h  = $dc05;        cia#1 timer register, hi byte
el 1280 d1icr  = $dc0d;        cia#1 interrupt control register
ag 1290 d1cra  = $dc0e;        cia#1 control register a
mk 1300 convrt = $de00;        memory location of adc
pe 1310 mmucr2 = $ff00;        mmu configuration register
gi 1320 ;
lc 1330 ;*********************************
bo 1340 ;* define variables              *
mg 1350 ;*********************************
ja 1360 ;
cg 1370 ram0   = %00111110;    mmu configuration for bank 0 & i/o
in 1380 onek   = %00000100;    mmu configuration for 1k common ram
il 1390 start  = %00000001;    cia timer start code
ok 1400 stop   = %00000000;    cia timer stop code
ab 1410 clricr = %00011111;    clear cia interrupt sources
```

```
fl 1420 fast   = %00000001;   fast cpu enable
eg 1430 blank  = %00001011;   blank vic screen
oa 1440 ;
ck 1450 ;*********************************
kd 1460 ;* start of monitor routine.     *
po 1470 ;* enter this routine after the *
nk 1480 ;* speed register ($0afd) has    *
mf 1490 ;* been set with a legitimate    *
di 1500 ;* value.                        *
gg 1510 ;*********************************
da 1520 ;
op 1530 *= $e000
ef 1540 ;
jb 1550 monitr jsr setup:     set up ram, nmi  cia, and vic
og 1560 mloop  ldx dlicr:     read timer status
di 1570 beq mloop;      wait for timer to underflow
il 1580 lda convrt;     get adc reading
ae 1590 and #$0f;       screen upper nibble
gf 1600 sta sigvol;     play it
cp 1610 ldx restor;     was restore key pressedprint
af 1620 beq mloop;      no---get next sample
fg 1630 jmp reset;      yes--restore computer config, exit
kj 1640 ;
pd 1650 ;*********************************
fk 1660 ;* start of record routine.      *
mh 1670 ;* enter this routine after the *
kd 1680 ;* speed register ($0afd) has    *
io 1690 ;* been set with a legitimate    *
jf 1700 ;* value.   the last page of     *
jo 1710 ;* ram the sample occupies will *
oa 1720 ;* be stored at $0aff.           *
ed 1730 ;*********************************
an 1740 ;
ba 1750 record jsr setup;       set up ram, nmi, cia and vic
eb 1760 rloop1 ldx dlicr;       wait for timer underflow
lj 1770 beq rloop1
eh 1780 lda convrt;             get reading from adc
ob 1790 asl;                    move digitized data
pe 1800 asl;                    to upper nibble of
ol 1810 asl;                    accumulator...
oo 1820 asl
bc 1830 sta orme+1;             stash it for later
ic 1840 ldx restor
me 1850 bne endrec;             continue if restore not pressed
og 1860 rloop2 ldx dlicr;       wait for next underflow
ce 1870 beq rloop2
km 1880 lda convrt;             get reading from adc
cn 1890 and #$0f;               mask upper nibble
fn 1900 orme   ora #$ff;        or result with first reading
io 1910 sta (istore),y;         store data in memory
po 1920 iny;                    increment counter
fc 1930 bne rloop1;             get next byte if counter not 'flipped'
ng 1940 inc istore+1;           modify vector
op 1950 lda #>strtop
gn 1960 cmp istore+1;           reached top of memoryprint
ip 1970 bne rloop1;             no-- fill next page
ek 1980 endrec lda istore+1
co 1990 sta strbot-1;           store last page filled
pn 2000 jmp reset;              restore computer config and exit
fm 2010 ;*********************************
eg 2020 ;* this is the play routine.     *
ih 2030 ;* no settings are required      *
fm 2040 ;* since it will use the info    *
ph 2050 ;* stored during the record      *
jc 2060 ;* process.                      *
jj 2070 ;*********************************
gd 2080 ;
ek 2090 play   jsr setup;       set up ram, nmi, cia and vic
lc 2100 lda strbot-1
id 2110 sta endpla+1;           store ending page for later
cj 2120 playit lda (istore),y; get sample
ec 2130 lsr
he 2140 lsr;                    isolate first sample
fg 2150 lsr
ga 2160 lsr
nl 2170 ploop1 ldx dlicr;       wait for timer underflow
en 2180 beq ploop1
em 2190 sta sigvol;             play sample
ni 2200 lda (istore),y;         get sample
af 2210 and #$0f;               isolate second sample
jg 2220 ploop2 ldx dlicr;       wait for timer to underflow
if 2230 beq ploop2
ho 2240 sta sigvol;             play sample
ej 2250 iny;                    increment counter
mc 2260 bne playit;             play it again, sid!!!
hj 2270 inc istore+1
af 2280 endpla lda #$ff;        get ending memory location
jo 2290 cmp istore+1;           end of memoryprintprintprint
pc 2300 bne playit;             no-- play next page
nl 2310 jmp reset;              yes--reset config, exit
fe 2320 ;
jo 2330 ;*********************************
lj 2340 ;* setup routine.  initializes  *
hf 2350 ;* mmu, nmi, cia, and vic.      *
lm 2360 ;*********************************
ig 2370 ;
do 2380 setup  sei;             disable irq
fn 2390 pla;                    pull return address from stack
ka 2400 sta cnmi-2
bo 2410 pla
kl 2420 sta cnmi-1
mc 2430 ;
on 2440 lda mmucr2;     get current memory bank
oa 2450 pha;            store it
bg 2460 lda #ram0;      set for ram(0) and i/o
eh 2470 sta mmucr2
np 2480 lda mmurcr;     get current ram configuration
di 2490 pha;            save it
ha 2500 lda #onek
ia 2510 sta mmurcr
bm 2520 ;
pi 2530 lda column;     80 columns displayedprint
ce 2540 cmp #$4f
ba 2550 bne nofast;     no---don't set fast mode
om 2560 lda #fast;      yes--set fast mode
gh 2570 sta clkrte
cn 2580 lda #blank
bh 2590 sta scroly
gn 2600 ;
ba 2610 nofast ldy #$0d;        initialize counter
ab 2620 cpynmi lda nmi,y
hg 2630 sta cnmi,y;             copy nmi routine to unused ram
in 2640 dey;                    decrement counter
en 2650 bpl cpynmi;             copy next byte
bc 2660 lda inmi;               save nmi vector
bl 2670 pha
jj 2680 lda inmi+1
cp 2690 pha
jl 2700 ldx #<cnmi;             reset to custom routine
cn 2710 ldy #>cnmi
oc 2720 stx inmi;               store new vector
dk 2730 sty inmi+1
en 2740 ldy #$00
hk 2750 sty restor;             clear restore flag/counter
an 2760 ;
pj 2770 sty istore;             set up storage vector
fa 2780 lda #>strbot
jd 2790 sta istore+1
pe 2800 sty strbot-2;           clear ram bank register
dp 2810 ;
fj 2820 lda #clricr
dh 2830 sta dlicr;              clear all interrupt sources in cia#1
bg 2840 lda #stop
aa 2850 sta dlcra;              stop timer
```

```
db 2860 lda strbot-3;          get sample speed
bl 2870 sta d1tll;             set timer lsb
fp 2880 sty d1tlh;             set timer msb
fk 2890 lda #start
do 2900 sta d1cra;             start timer
ke 2910 ;
kd 2920 lda cnmi-1;            push return address back
ca 2930 pha
jh 2940 lda cnmi-2
de 2950 pha
io 2960 rts
ca 2970 ;*********************************
cp 2980 ;* this routine restores the     *
jf 2990 ;* computer to normal operation *
cl 3000 ;* before leaving tczounds!      *
ei 3010 ;*********************************
bc 3020 ;
pf 3030 reset  lda #stop
nl 3040 sta d1cra;      stop cia timer
pc 3050 pla;            restore nmi vector
dn 3060 sta inmi+1
le 3070 pla
ej 3080 sta inmi
an 3090 pla;            restore mmu configuration
nb 3100 sta mmurcr
nn 3110 pla
ne 3120 sta mmucr2
id 3130 cli;            enable system irq
ia 3140 rts;            exit program
jf 3150 ;
np 3160 ;*********************************
ip 3170 ;* this section is copied into  *
ig 3180 ;* common ram at cnmi.  it       *
pi 3190 ;* is used when the restore key *
kk 3200 ;* is pressed.                   *
bb 3210 ;*********************************
ni 3220 ;
io 3230 nmi    lda #$ff
he 3240 sta restor;     set restore flag
po 3250 pla;            get ram configuration register
of 3260 sta mmucr2;     restore memory
hn 3270 pla
md 3280 tay;            pull .y
jb 3290 pla
mk 3300 tax;            pull .x
ll 3310 pla;            pull .a
ik 3320 rti;            end interrupt
dn 3330 .end
```

Quick start instructions:

```
run"tczounds-loud"
moN                    (monitor command)
L"tczounds.bin",8
L"0/energize",8
J 0E04E
```

C-128 ".RAW" PLAYERS

BY SHANE BURTON

Editor's note: Because we have already run two articles in issue #34, one for the C-64 and one for the C-128, explaining how analog sound is recreated in digital form on computers, I have elected not to run a full article with the following program listings.

The functions of the programs are fairly easy to understand and most assembly language programmers can easily modify the source code to work within various applications and with various assemblers.

The machine language modules (rawxxx.bin) can easily be merged into any Basic 7.0 program for playing digital sound within programs or games. There are hundreds, maybe thousands, of ".raw" digital format sound files available on bulletin board services and public domain disks.

You can adapt these source code files to play other digital sound formats too! Often the only difference between the various digital sound format files is the header for that particular machine or program. By using the BLOAD command or a sector editor this is a fairly easy task to accomplish.

A good example might be that to create "DigiTalker 128" files all you have to do is prepend the DT128 file headers to the beginner of the ".raw" file data. Or you can prepend the ."raw" file header to the DT128 file data to create ".raw" files. The only thing you have to watch out for is setting the speed byte correctly in the created files. The original source code was developed for/from Atari ST sound files and source code from GEnie! The "vietnam.raw" file was converted from an Atari ST sound file.

PROGRAM NAME:RAWSYS128.BAS

```
ho 100 rem ------------------------------------------------
kh 110 rem filename = rawsys128.bas
no 120 rem rawplayer 128 v1.0 is commercial software from
ag 130 rem twin cities 128/64 issue #34 by:
mo 140 rem parsec inc po box 111  salem ma 01970-0111 usa
lc 150 rem copyright 1992 - all rights reserved
lk 160 rem ------------------------------------------------
lp 170 rem plays sample from system bank 1
pm 180 fast
no 190 bload"rawsys128.bin",b0,p2816:bload"vietnam.raw",b1,p1024
mc 200 a=peek(175):a=a-3:poke2879,a:poke2882,4:poke2884,22:poke2883,1
eo 210 sys2816
```

PROGRAM NAME:RAWSYS128.SRC

```
ce 1000 ; ------------------------------------------------
fl 1010 ;   filename = rawsys128.src
nm 1020 ;   rawplayer 128 v1.0 is commercial software from
jm 1030 ;   twin cities 128/64 issue #34 by:
ah 1040 ;   parsec inc po box 111  salem ma 01970-0111 usa
ki 1050 ;   copyright 1992 - all rights reserved
gb 1060 ; ------------------------------------------------
gn 1070 ;
ba 1080 sidvol = $d418
fn 1090 indfet = $ff74
il 1100 ;
```

```
fa 1110 *= $0b00
jp 1120 ;
eb 1130 sei;                    disable all maskable interrupts
kp 1140 ldy offset;             offset for indirect y add for indfet kernal
                                routine
mn 1150 lda startlsb;           startlsb holds the lsb of the samples location
ib 1160 sta $fa;                store lsb in zero page
bj 1170 lda startmsb;           startmsb holds the msb of the samples location
kh 1180 sta $fb;                store msb in zero page
pg 1190 start lda #$fa;         $fa is where to find the pointer to your address
je 1200 ldx bank;               bank holds the bank# from which to read data (0/1)
on 1210 jsr indfet;             call indfet kernal routine located at $ff74
ad 1220 ;
cf 1230 clc;                    clear the carry flag
fj 1240 lsr;                    manipulate the byte in the accumulator
mo 1250 lsr
ni 1260 lsr
oc 1270 lsr
gn 1280 sta sidvol;             store the acc. in the sid volume register ($d418)
ff 1290 iny;                    increment y register
ik 1300 sty offset;             put increased y value in offset to be read again
fo 1310 ;
ej 1320 cpy #$ff;               is y 255print
pe 1330 bne speed;              if not then go to the speed routine before before
ei 1340 ;                         fetching the next byte
cm 1350 ldy #$00;               reset y register to zero
ng 1360 sty offset;             put a zero in offset
fh 1370 inc $fb;                increase the msb. we have already read 255 bytes
hd 1380 lda $fb;                what is our msb now
cg 1390 cmp endmsb;             is the current value of $fb the last byte we wish
jf 1400 ;                         to read
eh 1410 bne speed;              if not then go to the speed routine before
                                fetching
pc 1420 ;                         the next byte
fb 1430 cli
ji 1440 rts
ok 1450 ·
ok 1460 speed ldx begval;        beginning value in countdown
df 1470 decrease dex;            decrease begval by one
lg 1480 bne decrease;            are we down to zero yetprint
af 1490 jmp start;               let us do it all again for the next byte
bm 1500 ;
km 1510 endmsb .byte $00
hi 1520 offset .byte $00
np 1530 startlsb .byte $00;      initialize all addresses with zero
bf 1540 startmsb .byte $00
pa 1550 bank .byte $00
hj 1560 begval .byte $00
fg 1570 .end
```

PROGRAM NAME:RAWREU128.BAS

```
ho 100 rem ------------------------------------------------
in 110 rem filename = rawreu128.bas
no 120 rem rawplayer 128 v1.0 is commercial software from
ag 130 rem twin cities 128/64 issue #34 by:
mo 140 rem parsec inc po box 111  salem ma 01970-0111 usa
lc 150 rem copyright 1992 - all rights reserved
lk 160 rem ------------------------------------------------
jg 170 rem plays sample from reu bank 0
po 180 slow
pk 190 poke 53265,32
ih 200 bload"rawreu128.bin",p2816:bload"gameover.raw",b1,p1024
co 210 a=peek(175):a=a-4:poke2910,a:poke2911,1
ao 220 bank1:stash (a*256),1024,0,0
bi 230 bank15:sys2816
```

PROGRAM NAME:RAWREU128.SRC

```
ce 1000 ; ------------------------------------------------
fe 1010 ;   filename = rawreu128.src
nm 1020 ;   rawplayer 128 v1.0 is commercial software from
jm 1030 ;   twin cities 128/64 issue #34 by:
ah 1040 ;   parsec inc po box 111  salem ma 01970-0111 usa
ki 1050 ;   copyright 1992 - all rights reserved
gb 1060 ; ------------------------------------------------
gn 1070 ;
hc 1080 dmalo = $df04
fj 1090 dmahi = $df05
jh 1100 dmaadl = $df02
jf 1110 dmaadh = $df03
al 1120 dmabnk = $df06
oo 1130 dmadal = $df07
om 1140 dmadah = $df08
fg 1150 sidvol = $d418
mh 1160 ;
im 1170 *= $0b00
nl 1180 ;
hn 1190 sei;                    disable all maskable interrupts
gm 1200 lda startlsb;           startlsb holds the lsb of the sample location in
                                reu
pn 1210 sta $fa;                store leb in zero page
li 1220 lda startmsb;           startmsb holds the msb of the sample location in
                                reu
nj 1230 sta $fb;                store msb in zero page
bh 1240 ;
ni 1250 start lda $fa;          $fa contains the lsb of reu address we want 2 read
on 1260 sta dmalo;              store the value that we found in $fa at $df04
gk 1270 lda $fb;                $fb contains the msb of reu address we want 2 read
ad 1280 sta dmahi;              store the value that we found in $fb at $df05
ek 1290 ;
lo 1300 lda #$fc;               lsb of internal ram address we are going to write
                                to
io 1310 sta dmaadl;             store #$fc at $df02
nc 1320 lda #$00;               msb of internal ram address we are going to write
                                to
ml 1330 sta dmaadh;             store #$00 at $df03
hm 1340 ;
gb 1350 lda reubank;            reubank contains reu bank# we want to address(0-7)
gl 1360 sta dmabnk;             store reubank at $df08
jk 1370 :
ma 1380 lda #$01;               lsb of # of bytes we want to read
jo 1390 sta dmadal;             store #$01 at $df07
ng 1400 lda #$00;               msb of # of bytes we want to read
jf 1410 sta dmadah;             store #$00 at $df08
mm 1420 ;
gf 1430 ldx #$00;               c128 bank we are going to write the fetched byte
                                to
ac 1440 ldy #$81;               #$81 indicates we are going to fetch data from reu
mb 1450 jsr $ff50;              jump to dmacall kernal routine and fetch the data
pe 1460 ;
ih 1470 lda $fc;                load the fetched byte into the accumulator
ca 1480 clc;                    clear the carry flag
fe 1490 lsr;                    manipulate the byte in the accumulator
mj 1500 lsr
nd 1510 lsr
nn 1520 lsr
kb 1530 sta sidvol;             store the accumulator in the sid vol reg ($d418)
ef 1540 ;
mo 1550 inc $fa;                add 1 to the value in $fa so we can read the
mj 1560 ;                         next byte
jh 1570 ldy $fa;                load $fa into the y register
fh 1580 cpy #$ff;               is y 255
pm 1590 bne speed;              if not then go to the speed routine before
                                fetching
kh 1600 ;                         the next byte
ke 1610 ldy #$00;               put a zero in the y register
fc 1620 inc $fb;                increase the msb. we have already read 255 bytes
```

```
go 1630 lda $fb;              what is our msb now
km 1640 cmp endmsb;           is the current value of $fb the last byte we
hj 1650 ;                        wish to read
ec 1660 bne speed;            if not then go to the speed routine before
                              fetching
on 1670 ;                     the next byte
em 1680 cli
jd 1690 rts
of 1700 ;
of 1710 speed ldx begval;     beginning value in countdown
da 1720 decrease dex;         decrease begval by one
lb 1730 bne decrease;         are we down to zero yetprint
aa 1740 jmp start;            let us do it all again for the next byte
bh 1750 ;
kh 1760 endmsb .byte $00
el 1770 begval .byte $00
hc 1780 startlsb .byte $00; initialize all bytes to zero
ap 1790 startmsb .byte $00
hi 1800 reubank .byte $00
eh 1810 .end
```

(IN THE NEWS - continued from page 4)

Best Computer Supplies
4980 Longley Lane Ste 104
Reno NV 89502
1-800-544-3472 Voice
1-702-826-4376
1-702-826-4392 FAX

A SCARY THING ?

From Mark Dulski: I am currently reading "Needful Things" by Stephen King. The story takes place in 1991 and guess what one of the main characters just bought his son? A new Commodore 64! That's the first time I've seen Commodore mentioned in a work of fiction. The 64's are still known.

NEW GENIE SIGNUP NUMBER

There is a new GEnie signup number and password for the Commodore 64/128 Roundtable. It is xtx99018,commrt Make sure you use it!

RIO COMPUTERS

Rio Computers is having a summer time special on the Video Digitizer we reviewed in this issue, only $149.95!

A PROFILE OF A COMMODORE USER: EDITOR'S PROLOGUE

BY JOHN W. BROWN

I first inquired about Mr. Lauder's involvement with the Leonard Wood Memorial Foundation when I noticed his letterhead and wondered if he was using his C-128 to help run such a big business. Subsequently he sent me just about everything you could want or need to know about the L.W.M., himself, and his work. His professional credits are quite impressive and and take two pages by themselves so I will not list them all here. I would like to add though looking at his biographical summary you know he is someone special, more so for the work he currently does.

People often wonder about using their Commodore computers for "serious" business use. You can not get more serious than this and the budget he helps handle on his C-128 approaches seven figures, that is serious!

Excerpted from their annual report:
" Most people think leprosy was a scourge of past centuries and no longer exists. Unfortunately, this is not true. Leprosy is still very much with us today. There are 6,000 KNOWN (1990) leprosy cases in the United States. Worldwide, there are an estimated 10-12 million people with the disease. Leprosy is a far more serious problem than sheer numbers, because it involves disabilities, economic loss, and emotional trauma for those involved."

"Leprosy is caused by a bacterium. There are two types of the disease, one relatively mild and non-progressive, the other severe, progressive and disfiguring. We do not know what determines which form a person acquires. The majority of people are either immune or for some reason non-susceptible to the infection."

"The American Leprosy Foundation subsists by donations from the general public, bequests, special gifts. Combined Federal Campaign, national and international research grants, and gifts from private citizens. We are a Combined Federal Campaign #0306."

If you would like to contact them further or make a donation call or write:

Leonard Wood Memorial
American Leprosy Foundation
11600 Nebel St, Suite 210
Rockville MD 20852
USA

Tel# 301-984-1336
FAX# 301-770-0580

The following is his letter in a pretty much unedited form. Enjoy the reading.

A SELF-PROFILE OF A COMMODORE USER

BY ROBERT B. LAUDER, JR.

Dear Mr. Brown:

This is in response to your request as to how I use the Commodore 128. The following programs are the ones I use:

1. Timeworks Wordwriter 128
   Reports and lengthy correspondence

2. PCFILE-80 (CP/M):
   Cataloging a recorded music library
   Personal asset records
   Membership records for our amateur radio club

3. DataComp-Business Manager-General Ledger (CP/M):
   Financial records for the Leonard Wood Memorial
   (aka American Leprosy Foundation)
   Personal financial records

I am not into programming - a computer is a means to an end, not an end in itself. I want a program to provide a certain function and do not have the patience to program. I do not have a modem and therefore I am not into online communications. I have an advanced class amateur radio license and communicate with fellow amateurs, many of whom have computers. So far I am not into graphics, desktop publishing, or games. I have a couple of games but seldom have time to play them.

I have a B.A. in Accounting, acquired after 11 years of evening classes at the George Washington University, and in 1979, retired from the Federal Government as a division budget officer at the National Institutes of Health in Bethesda, Maryland.

In 1980, I became the accountant for the Leonard Wood Memorial which is also known as the American Leprosy Foundation. This non-profit, medical research organization was founded in 1928. Administrative headquarters are in Rockville, Maryland and research laboratories on the grounds of a Philippine leprosarium in Cebu City, The Philippines.

In 1983, we moved from Rockville, Maryland to Fairfield, Pennsylvania (eight miles west of Gettysburg). Since then, I been the accounting consultant for the Leprosy Foundation. After I acquired my first computer, at age 62 in 1985, the Leprosy Foundation let me computerize their manual accounting system and I prepare

all their monthly financial reports on
my Commodore 128 at home

from summary data prepared by their bookkeeper. I keep in touch by telephone and financial data is either mailed or faxed to me and I make trips as necessary to their Rockville office.

My first computer was a Commodore 64 which I obtained with the intention of using it for amateur radio. I soon became intrigued with the possibility of using it for accounting - entering figures once and letting the computer prepare the various reports.

I tried a number of general ledger programs for the C-64 but none had the two basic characteristics needed for the Leprosy Foundation's financial requirements. All of the Foundation's research projects are funded by grants, mostly from foreign sources which do not recognize overhead. It is necessary to allocate all expenses, however small, to the various projects instead of just charging direct expenses and then allocating indirect expenses via one or two overhead amounts.

With 29 active projects, any of which could use all of the 42 expense categories, plus accounts for balance sheet and income items, the basic requirement is for at least 1,500 general ledger accounts. In addition, eight alpha/numeric characters are required for charging expenses to the projects (i.e., 120A/530).

At this point, some consideration was given to obtaining an IBM compatible computer. My brother's CPA firm sent me summaries of over 35 accounting programs in MS-DOS. However, none of of these had the ability to provide both the capacity for the large number of accounts and the eight characters for the account number.

Fortunately, in 1986, I saw an ad in RUN Magazine for a general ledger program. The price in the ad was $39, although I later was sent an invoice showing the original price as $450 discounted this once to $39. After paying a lot more for C-64 accounting programs that did not work as advertised, I reasoned that it was worth taking a chance and I ordered this program - the DataComp-Business Manager- General Ledger.

After thoroughly reading the manual and listening to the tutorial audio cassette, I took another chance and purchased a Commodore 128, (2) 1571 disk drives, and a Magnavox RGB monitor because this program was in CP/M. I had previously purchased two reconditioned Commodore printers, a DPS-1101 (actually a Juki 1000) daisy wheel and a 1526 dot matrix.

This general ledger program was obtained from Software Marketers & Publisher, Grants Pass, Oregon. The owner of this firm died suddenly in 1987 and one of the employees, a Mr. I. J. Blevens. took over the business, changed the name and moved to another location in Grants Pass. Other than having a brief conversation with Mr. Blevens when he told me the new address. he has not responded to numerous letters or messages left on his answering machine regarding possible upgrades, etc. and even Gale Rhoades, then Executive Director of FOG or her friends in Oregon were unable to contact Mr. Blevens.

This program might be a good candidate for a public domain program but I have no idea how this might be accomplished. Futhermore, while this program is excellent for the Leprosy Foundation's requirements, it may not be suitable for one conducting a public accounting practice. It takes quite a lot of time to enter the data and each item has to be entered, one at a time instead of compound entries but to offset this, it has several features which can immediately detect errors in entering the data.

If there is any interest by Parsec or others in this program, I would be glad to provide additional information as I have been using this program since 1986 with excellent results. At least I purchased the C-128 system the right way - bought the computer to run the program.

In 1991, I bought the following for my C-128 system:

(2) 1581 disk drives, now used almost exclusively for all programs.

A Star NX-1000C Rainbow dot matrix printer (painfully slow in the NLQ mode).

A used (but tested) C-128 to use as backup, if ever necessary.

64K Video Ram module for the flat C-128.

The CMD 1 Mb RamDrive with JiffyDos chips for all four disk drives.

I was very upset and felt betrayed by CMD when I discovered that RamDrive only supports CP/M in the 1541 mode in a limited way and neither the 1571 nor 1581 drives. Their ads indicate that RamDrive is compatible with CP/M - the only reason for my purchase.

I need the additional RAM for data based work - cataloging a large music library. I do have Fleet System 4 but my limited attempts to try this program have been frustrating. I decided this year that if CMD was so oriented towards GEOS that perhaps this would be the way to go. I have obtained the GEOS program, Version 2.0 from RUN Magazine, bought a 1351 Mouse, CMD sent me a free copy of Gateway and I'm still waiting after four months, to receive GeoFile from RUN and plan to wait until I receive this latter program to boot GEOS for the first time. (Editor's note: CMD bought out the TechMedia/RUN supply of GEOS products after this article was written.)

I have tried, several times, to copy the Timeworks Wordwriter 128 program disk, without success. (Editor's note,it must be a old copy protected version).

There are no user groups in a convenient range of where we now live. I belonged to FOG until recently. When they raised their annual fee to $40 for four issues a year and since they are not devoted entirely to CP/M now, I decided to drop my membership. I have had to teach myself about computers. I plan to continue to use my C-128.

You have my permission to use any of my comments or enclosed material for whatever purposes you wish. May Parsec have a long and profitable existence.

Sincerely yours,
Robert B. Lauder, Jr.

DR.EVIL
ONLINE INTERVIEW.

BY JOHN BROWN

The following is a unpublished online (GEnie) interview hosted by Parsec, with Kent Sullivan of Dr.Evil fame who originally made and sold the Stereo SID and Swiftlink RS-232 cartridges CMD now sells. Lack of page room in previous issues (due to the smaller page format) prevented us from printing this article before now. A very interesting interview with some unique perspectives and insights.

Date: 91-01-10
Time: 22:48EDT

Minutes:

<C128.JBEE> I like Della Street :D
<DR.EVIL> Hi Andrew!
<[DTJ Andy] A.BERNHARDT> Hi JBEE, Ken
<C128.JBEE> Hi Andrew and Andy :)
<C128.JBEE> just a little chat before we start :)
<C128.JBEE> Kent, how do you have Desterm setup ?
<C128.JBEE> Hi Tom :)
<A.BERNHARDT> Hi Tom
<[Tom] T.ADAMS18> Hi
<[Kent] DR.EVIL> Howdy.
<C128.JBEE> you got the hang of the nickname :)
<[Kent] DR.EVIL> Finally!
<C128.JBEE> Golly. JBEE is my nickname. suppose I should use my real name -lol
<[Kent] DR.EVIL> Good point.
<H.HERMAN1> is here.
<C128.JBEE> Hi Howie :)))
<[Tom] T.ADAMS18> Hi Howie
<[Howie] H.HERMAN1>hello
<A.BERNHARDT> Hi Howie
<[Kent] DR.EVIL> Howdy Howie. I love saying that.
<C128.JBEE> lol
<[Howie] H.HERMAN1>Hi Guys!
<[Howie] H.HERMAN1>Was it rough signing on tonite...
<C128.JBEE> People love introducing my wife as "TBEE" :D
<C128.JBEE> (tracy)
<[Howie] H.HERMAN1>Ha
<C128.JBEE> really? lines were good today, must be the weather down there?
<C128.JBEE> everyone ready to start?
<[Kent] DR.EVIL> Everything was fine here in Seattle.
<[Howie] H.HERMAN1>everything just sorta went slo
<[Tom] T.ADAMS18> Ready here.
<A.BERNHARDT> Here too
<[Howie] H.HERMAN1>AOK here
<[Kent] DR.EVIL> Fire when ready.
<C128.JBEE> okay.... lights, curtains.. action? :)
<[Howie] H.HERMAN1>BLAST!!!!!!!!

<C128.JBEE> Tonight we are interviewing Dr.Evil for Twin Cities 128 to kick off a series of in depth articles on modems (can you say 9600?) and the SID chip. I know Kent and got a chance to meet him at WoC 90, though I imagine our readers would like to know a little about yourself. Kent, do you want to fill in a little about yourself and where you work?

<[Kent] DR.EVIL> There's nothing too exciting. I started Dr. Evil Labs in college (Purdue) with two friends. Our first project was distributing Kermit for the C-64. We then developed a shareware adventure game system that went nowhere. Then I got the bright idea that what the world needed was a SID cartridge. I have since graduated and moved to Seattle.

<C128.JBEE> What is "Kermit"? Besides green :)

<[Kent] DR.EVIL> Kermit in the context above is a public domain telecomm program that provides an implementation of the basic Kermit file transfer protocol plus some useful terminal emulations such as DEC VT-52, VT-100 and limited Tektronix 4010/4014. Kermit in general is a very flexible file transfer protocol designed basically for mainframe <--> mf or mf <--> micro transfers.

<C128.JBEE> Did your idea of the SID cartridge come before or after the internal SIDprojects were published?

<[Kent] DR.EVIL> Definitely after. I met Mark Dickenson at the first "SIDFest" in Columbus OH in 8/87. I then monitored the goings-on on Q-Link (gasp!) in the Music Room and decided that not enough people could have stereo SIDs because the hardware was hard to install.

<C128.JBEE> Would you say it is a safe bet that Mark and Sidplayer 10.3 can be credited with much of the success of the Sid cartridge?

<[Kent] DR.EVIL> Absolutely. We didn't have the resources or time to write our own stereo SIDplayer. I have been friends with Craig Chamberlain (Sidplayer author) a long time, so it's possible that he would have written a player for us. Hard to say...

<C128.JBEE> How many major versions of the SID cart. did you produce, I know there was a battery backed version earlier.

<[Kent] DR.EVIL> Version 1 was the original. Not battery backed but battery powered--you HAD to have the battery to get it to work. Then we did v2, which removed the battery and added a first level of static protection. We gave CMD our plans for v3, which included some cleanup work and further static surge protection. I'm not sure if those have gone into production or not yet.

<C128.JBEE> I believe the earlier versions worked from the cassette port?

<[Kent] DR.EVIL> It's always been the same port--the "expansion" or "cartridge" port. There were no extra wires or connections. Just a cable to the RCA jack for audio out.

<C128.JBEE> What were the reasons for turning over the production of the SID cartridge to CMD?

<[Kent] DR.EVIL> Basically a lifestyle decision. Those of us doing the work found ourselves spending every free minute working on company business. With the intro of the SL-232 we were very busy filling orders and building units. We were doing this after the regular work day. 80-hour weeks were the norm for close to a year. The two principles (Bryan Minugh and myself) just ran out of gas. We knew tha CMD would do a quality job and might even be able to expand the market.

<C128.JBEE> Was this also the reason for turning over the SL-232 to CMD?

<[Kent] DR.EVIL> yes, the same reasons apply.

<[Kent] DR.EVIL> Also, I made a decision that I wanted to pursue my full-time career with Microsoft and not "further adventures" of Dr. Evil.

<C128.JBEE> Well, it was a good choice since it has made it available to many more people.I understand there a few games now that now support stereo music.

<[Kent] DR.EVIL> That's great! I only know of one--Mark's Rockfall (Boulderdash clone).

<C128.JBEE> Yes, great game, though I haven't played it in stereo yet.

<C128.JBEE> What originally made you go out and buy a C64 ?

<[Kent] DR.EVIL> My older brother bought one for me! :-) I had a Sinclair ZX-81 that he passed on to me, and I liked it but saw the limitations. I used Apple IIes in high school but knew there was no way to afford one of those. I didn't know much about the C-64 vs. the Apple feature-wise when I first got it. So the motive was basically economic.

<C128.JBEE> I know you work on IBM clones (maybe even real ones;) at work, are there any other computers you use when you come home, or is it just your trusty C128?

<[Kent] DR.EVIL> I have a Mac SE/30. I bought it just before I left college. I like it because it packs a lot of Ocmp in the small-box Mac and it goes well with the IIci I use at work. My word processor of choice has been MS Word on the Mac for several years so I wanted a machine to run it on. Also Pagemaker.

<[Kent] DR.EVIL> I have a C-128D, 1581, CMD 40 HD, RAMLink w/ 1 MB 1750 REU, SL-232, SID, etc.

<C128.JBEE> How does the Ramlink works with the SID cartridge?

<[Kent] DR.EVIL> They work together fine as long as you don't want to use a REU in direct (non-RAMLink) mode. If you use the REU in direct mode the second cart plug in (where the SID would be) is turned off. Same goes for SL-232.

<C128.JBEE> (btw that's a power Commodore system for sure

<C128.JBEE> Open question to the audience and Dr.Evil. I know I SID myself to sleep often about you people?

<[Kent] DR.EVIL> I just moved into a house. The stereo is now in a different room from the computer so it's tough to do. I plan to get a second amp some time.

<C128.JBEE> I use a couple of Radio Shack amps, not the cleanest sound but I like using the rechargeable batteries!

<C128.JBEE> So, you must be signed on now with the SL-232, where is that plugged in?

<[Kent] DR.EVIL> Yep, into the RAMLink. The cart expansion ports on the RL are very solid, much more so than the Aprospand.

<[Howie] H.HERMAN1>Kent, did I see you say that SL can't work with RL and a 1750 set to direct?

<C128.JBEE> yes, standing a REU or any large cartridge is to flirt with RFI or brokenboards!

<[Kent] DR.EVIL> I believe that's the case. The RL manual says that the Direct position turns off the other cart port.

<C128.JBEE> What prompted you to develop the Swiftlink-232?

<[Kent] DR.EVIL> I was on a crusade. At Purdue they developed a service where you could get a direct 9600 bps connection to your dorm room for ScheapS but there was no way for me to use it. I know that Desterm will supposedly do it thru the user port but I never had much luck. I think the Unix boxes on the other end spit out data too fast for it. As it turned out, the product didn't get off the ground until almost a year after I graduated.

<C128.JBEE> For our readers, what exactly is the SL-232?

<[Kent] DR.EVIL> The SwiftLink-232 is a high-speed serial interface cartridge for the C-64/128 (including CP/M) that allows much faster and more efficient serial communication (modem, printer, etc). It utilizes the chip that CBM "left out" of the machines -- the 6551 ACIA (UART) to offload most of the work from the CPU.

<C128.JBEE> How fast does the SL-232 operate?

<[Kent] DR.EVIL> Interestingly, one machine that CBM produced -- the Plus/4 -- has a 6551 in it.

<C128.JBEE> (yes, is that the TED chip?)

<[Kent] DR.EVIL> The 6551's normal top speed is 19,200 bps. Through a clocking trick we have doubled that to 38,400 bps.

<[Kent] DR.EVIL> No, the TED is a separate chip that was custom. It is a combo. music and graphics chip, I think.

<C128.JBEE> 9600 modems are coming down to around $400 street price, so the use for the really high speed would be from computer to computer. How is it for computer to computer transfers ?

<[Kent] DR.EVIL> The 6551 is off-the-shelf and is the same UART used, for example, in the Apple serial interface boards.

<[Kent] DR.EVIL> Not sure what you mean, JBEE -- can you clarify?

<C128.JBEE> Would the SL-232 be a good choice for computer to computer transfers?

<[Kent] DR.EVIL> Yes, qualified. If you do things like move files to an MS-DOS pc now and then, Big Blue Reader is more convenient because there's no cable. But if you do this type of thing regularly, the SL-232 might be more convenient. I think the biggest market so far has been BBS owners and "telecomm junkies". :-)

<C128.JBEE> Have you tried this with your Mac (BBR128 won't do Mac disks).

<[Kent] DR.EVIL> Yes. That was the test bench. Works great at 38,400 with Desterm. :-)

<C128.JBEE> wow, I bet you have a lot of pictures converted Macpaint pixs ;)

<[Kent] DR.EVIL> Our user group demo consists of 6 C-64 hi-res screens sent from the C-64 running at 38,400 using Xmodem (cops, I mean from the Mac to the C-64) and displaying on the C-64. We first show it at 1200 to demonstrate the speed difference.

<C128.JBEE> does the SL-232 have a ram buffer for such high speed telcommunications?

<[Kent] DR.EVIL> No. The 6551 does not have an on-chip buffer. Newer UARTs such as the NS 16550A (found in many 386 PCs) has one (16 bytes I think). The 6551 was designed in the early 80s when RAM was much more expensive.

<C128.JBEE> well, I think I will open up the conference to free for all questions, comments, and chats now. I want to thank you for dropping by and for all the past and future C128 articles :))))

<A.BERNHARDT> Where did you get the name Dr. Evil?

<[Kent] DR.EVIL> One of the partners, Roy Riggs. He got that nickname in high school for doing offbeat experiments in chemistry class.

<[Howie] H.HERMAN1>Are there any new 8-bit things you re doing that we might like to know about?

<[Kent] DR.EVIL> Nope, nothing. The company will officially close on the last day of December. Bryan and I both bought houses so, to be honest, that's where our attentions are these days!

<[Howie] H.HERMAN1>that's only fair :)

<C128.JBEE> Well, houses are a major consideration for sure :) BTW- Thanks for pumping up the economy :D

<[Kent] DR.EVIL> You're welcome. Hah! :-)

<A.BERNHARDT> Do you know of any software that will convert MIDI to stereo SIDs?

<[Kent] DR.EVIL> yes, Frank Prindle's MIDI <--> SID Connection v2. Should be in the Dr. Evil library here.

<C128.JBEE> Maybe I will grab it and stuff it on a PD disk for those interested.

<C128.JBEE> Classic Affair.arc here online was designed for midi playback, quite good! Just wish I owned a midi interface for my cheapo keyboard.

<[Howie] H.HERMAN1>Kent what are you doing for Microsoft?

<[Kent] DR.EVIL> I am a "Usability Specialist" which is sort of like a Human Factors Engineer. The group I'm in works to make sure the end user's voice is heard in the development process through design reviews and user testing with early versions of products.

<C128.JBEE> you make sure the Microsoft windows are in the right place? (sorry, small pun)

<[Kent] DR.EVIL> That's ok. :-)

<[Howie] H.HERMAN1>sounds interesting!

<C128.JBEE> Better to correct mistakes at the beginning instead of after the beta version, eh?

<[Kent] DR.EVIL> I love it. Amen, JBEE. Product support costs kill some companies.

<[Howie] H.HERMAN1>Have you run across conflicting wants? and what happens then?

<[Kent] DR.EVIL> Sure. It's all data and it's up to the program designers to make the final choices. We depend on a variety of quantitative and qualitative measures to hopefully make decisions easier.

<C128.JBEE> What piece of Microsoft software is your personal favorite?

<[Kent] DR.EVIL> The one I use the most is Mac Word. My favorite probably is something that hasn't been released yet. :-)

<C128.JBEE> keeping us on the edge of our seats :-)

<[Howie] H.HERMAN1>Do you get involved in any Micro sysoping support on the services?

<[Kent] DR.EVIL> No, I'm not involved with the MS online support.

<[Kent] DR.EVIL> We shipped over 250 products last year, counting localization for different foreign languages.

<C128.JBEE> (I think I read today MS shipped 300,000 copies of Windows to Japan so far)

<C128.JBEE> What is your favorite piece of software (besides Desterm and Sidplayer) on the C-64?

<[Kent] DR.EVIL> Probably the game "Uridium" from England. Awesome.

<C128.JBEE> Uridium! On of my favorites too, once you are past the copper you are golden;) I send a Uridium game pack (not pirated!) to a TC128 author who wanted a shoot-um up and he says it is too tough! :D

<[Kent] DR.EVIL> Are you selling the commercial version or ? I didn't quite get that

<C128.JBEE> Now, I just happened to have a couple of used copies from buying C128 systems :) I guess later versions were packaged with TOP GUN on the same disk

<[Kent] DR.EVIL> Ah! Gotcha. Thanks everyone. Talk to you soon!

<C128.JBEE> thanks you *bye*

<A.BERNHARDT> G'nite Kent
<[Howie] H.HERMAN1>Nite Kent. Nice see'in ya!!'

end of transcript.

DR.OCTAL'S SHARP OPERATING TIPS <TM>

Tip &0013
USING THE FD-4000
From: John W.Brown
The FD-Tools program on the FD-4000 disk is handy and very easy to use. However, if you are formatting 20 or more disks, that means cursoring this way and that, and replying to all those menu prompts over and over. It also takes time and requires your attention. Not to mention it can get boring and tedious having to answer the same set of questions twenty times in a row. So I just use this small Basic program, that requires a minimum of attention, to format disks while I read junk mail.

How to format a Extra High Density disk using Basic.

```
10 rem ft=hdn 4 megs
20 open 15,dn,15:print#15,"n1.name.id,ft":close 15
30 open15,dn,15:input#15,e,e$,t,s:close15:print
   e,e$,t,s
```

Tip &0014
USING GEOS 128
From: John W.Brown
This problem was originally solved for me by either Jim Collette or Robert Knop, so I can not take credit for it When you want to escape from GEOS 128 V2.0 using the reset button your system becomes locked and the 80

(continued on page 48)

GeoStamp Review

by John W Brown

Quincy Softworks "GeoStamp" disk is a collection of three GEOS programs, GeoStamp, StampEdit. StampCollect. and various data files to be used with GeoStamp. The purpose of the GeoStamp set of programs is to replace the cut - edit - move - option in geoPaint with a far better set of tools that let you easily and accurately cut and paste within a geoPaint document.

This is just about the best GEOS add-on I have ever used and it changes the nature of geoPaint entirely for me. There have always been many things I disliked about GEOS and one was the clumsy editing in geoPaint when trying to paste Photoscraps or to paste cut objects back onto the page. GeoStamp does such a good job I would consider it a "must have" for any serious GEOS artist. It works on the C-64 and on the C-128 in forty and eight columns. GeoStamp and Stamp Collect are Desk Accessories that can be called from within geoPaint.

GEOSTAMP
GeoStamp allows you to take up to thirty previously drawn pictures (or captured images) in a "stamp" collection and paste (stamp) them just about anywhere on a geoPaint document, within certain limitations. Using the forty column or eighty column screen you can not paste an image beyond the boundaries of the screen you are viewing. You have to scroll the screen and start your pasting of the "stamp" all over again. This is not so much a fault of the program, but the system and how GEOS stores documents.

A "stamp" is a small bitmap that you can use to stamp images on the screen. You can stamp a image into a solid background, leaving behind only the lines or bitmap of your drawing, or you can use another option and erase everything under the stamp when you place it onto the screen. Think of stamps as small pieces of clipart.

Where you are on the screen is displayed by X and Y pixel coordinates, which makes placing stamps much easier. A feature that is absolutely beautiful is an option to move the stamp according to "pixels", the width of the stamp, or by color cells (8 pixels).

The "S" option, for "S"tamp width, lets you effortlessly lay down a tiled imaged across the screen with all the borders and edges of the stamps lined up evenly. You do so by pressing the cursor keys to move the stamp and the P key to Place it. The other option lets you move the stamp with the cursor key by color cells, 8 pixel increments. This is very handy if you want to fill the image in with color later and want to avoid the "jaggies" or "stairs" of color in the printed out image if you own a color printer. Being able to move the stamp image with the cursor keys makes placement precise and accurate. You can also use your input device, but I prefer the cursor keys for the final placement.

Another good feature is the ability to turn off the coordinates for a smoother freehand drawing when placing multiple stamps over an image. This did not really effect the eighty column screen much, if at all, but was noticeable on the forty column screen.

STAMPS
Picking stamps from within a stamp collection is easy. You just use the #1 key to scroll backwards in the file and the #2 key to scroll forwards in the file. If you are on one of the boundary numbers. #1 or #30, you can still use the number keys to scroll backwards or forwards through the file. Example, if you are on stamp #1 and you want to reach stamp #30, all you have to do is press the #1 key and the stamp file will scroll backwards and stamp #30 will appear. You do not have to scroll through the other twenty-nine stamps to get at the one you want.

StampCollect and StampEdit are two companion programs that enable you to build (StampCollect) your stamp collection beyond those provided on the disk and to edit (StampEdit) existing ones. The amount of stamp collections you can have is only limited by disk size. Since all three programs work basically the same, I only want to touch briefly on the following subject.

StampEdit was designed and meant to be used on the forty column screen so when you use it on the C-128's 80 column screen the dialogue boxes do not always line up or erase part of the screen like they should. I think the only way around this would have been separate 80 column versions programmed only for the C-128. Though this would have been nice, GeoStamp is the main program, works fine on the C-128. and is the main reason for buying the package. Usually I subtract from the rating for something like this, but the main program GeoStamp is so nice to use and filled with such strong features like rotating a stamp on its X and Y axis, I did not.

SAFETY FIRST
When you quit a GeoStamp application without pressing S to save your data, you get a dialogue box prompting you if you want to save your data or not, before destroying it by exiting the application.

I tried my to make this program fail or corrupt data, I was not successful. Oh well! It was tested using Gateway V2.5 and worked well except that the ESC key for using Switcher on the C-128 happens to also be the key used by the GeoStamp applications to undo a past event on the C-128. This is not really a problem because you can use the C key to clear a stamp image and the U key to undo a placement of the stamp. All it means is you can not use the Switcher feature while using GeoStamp.

WHAT DIDN'T I LIKE
Sorry, I liked everything and was basically enthralled from having something that makes geoPaint worth using for me : -)

There are a few things that could be improved. The data files should be changed so the 40/80 column flag is set. Then the application can be launched on the 80 column screen by clicking on the data file. This is not how I usually launch my applications, though many people enjoy doing it this way. You can set the flags yourself with various PD utilities but you should not have to bother doing that since you are paying to use the program.

There is no printed manual. I understand this is to keep the costs down on both the product and the shipping. Considering the low cost of the product and the shipping, this is acceptable. Especially when all the features and commands are reproduced on the help screens, which can be accessed by pressing the H key.

The manual is provided on the disk in geoWrite format which is okay, but this being GEOS I would have liked a geoPublish version that I could have dumped to my Postscript printer for really nice looking documentation.

Grade: A+

Price: $13.95

Checks or money orders, U.S. funds. CA residents add sales tax, overseas add 15% for S&H

GEnie: DiBieF"
Quincy Softworks
9479 E Whitmore Ave
Hughson CA 95326-9745
USA

MOUSE-CARE

By James Robbins

I was under the impression that the mouse I use needs no more care than cleaning the ball once in awhile. as per the instructions in almost any mouse manual. But. after three dead mice. I have learned the hard way that is not always so!

Because of my three dead mice, I was forced to find a solution to keep mice healthy and happy. If you live near a smog-riddled city, like I do. and you use your mouse on one of those 'mouse-mats'. I have some care tips for you.

Get a VCR cleaning kit. like the ones sold at Radio Shack. The kit should include cleaning sticks. cleaning fluid, and a white anti-static glove. Yes, there are computer chips in mice. Next buy some lint-free cloths or towels, which are also sold at Radio Shack and other computer stores. You will need a can of Static Guard. a anti-static sprav. and a roll of masking tape.

Turn off the computer. put on the static free glove. and unplug the mouse. Then remove the ball from the socket. which is underneath the mouse.

Stretch out a piece of the masking tape and roll the ball over it to get off the excess dirt and dust. then clean the ball with the cleaning fluid and a lint-free cloth.

Next. clean the cover that held the mouse ball in place, using the cleaning fluid and lint free cloth.

If you have a small computer vacuum, it would be a good idea to vacuum the inside of the mouse to keep the moving parts free of dust and lint You do not have to take the mouse apart any further than removing the ball, so do not get overly involved. You might wind up losing a spring or two and then you would have a real non-functioning mouse.

Now take one of the sticks provided in the VCR kit If it is too big to fit inside your mouse. then cut it down to size with a pair of scissors. Put some cleaning fluid on the stick. Clean the brass rollers inside the hole where the mouse ball rolls around. This might take a little practice, as the hole is small and the rollers move easily. but it is the most vital part of the cleaning!

The brass rollers should look smooth and shining clean. If it looks like it has hair on it. or a kind of blackish tape or strip, you are in for some extra work. That gunk is a build-up of fibers from the mouse mat and plain old greasy dirt. If the dirt builds up too much it may unseat the sensors inside the mouse or jam the ball in the socket.

You may have to rub a bit hard to get it off. but that gunk is what causes the mouse to get a mind of its own when it does not move. Just do not go crazy and rub too hard. you might break one of the roller supports. Just use enough pressure to get off the dirt.

Set the mouse aside for 5 or 10 minutes so the excess cleaning fluid dries. Now put the ball back inside the mouse and recover the ball.

CLEAN THE MOUSE-MAT
If you stretch out a long piece of masking tape and fold it around the roll of tape upside down. you will have a home-made lint remover. This is a good technique for your sweaters and clothes too. Take some of the masking tape and roll it over the mouse mat a few times to get off the excess lint and dusty dirt. After you are satisfied that the mat is clean. sprav it with the Static Guard. This will help save your mouse from static damage. plus. it will help keep dust particles from sticking to the mat that cause premature cleaning of the mouse.

There are no set schedules to follow for when to clean a mouse. just clean it when it starts acting erratic. All you need to do is once in awhile remove the ball and take a look inside. If the ball or rollers are dirty then clean the mouse.

Keeping the mat free of dust and lint on a daily basis is a better idea and will help stop the mouse from picking up too much dust and lint. Also avoid placing food and drinks on top of your mouse mat. That attracts dirt.

Happy Mousing!!!

CARDFILE/RECIPES
A SUPERBASE RECIPE PROGRAM

BY AL ORAM

I have cooked since I was a teenager but not seriously until I retired. Once I became involved in cooking (my wife was delighted), I began to search for a good recipe program. There were not any recipe programs I tried that were satisfactory, until I used Superbase to design my own. What I like about Superbase is the ability to print what you want, where you want it. As with all programs there are some things that could be improved but Cardfile/recipes satisfies more of my requirements than any other recipe database I tried to date.

I do not like writing recipes long hand because my handwriting is not that good and it was a chore to copy recipes for others. As I found recipes that looked good, they were entered into Cardfile/recipe. They were deleted if not found to be a recipe that was worth cooking many more times. This method created a recipe file that was easy to search for good recipes. You do not have to use this method if it does not suit you, it will be your recipe file. You may even find ways to improve the program, or these instructions might give you ideas on how to develop a recipe program using another database.

Cardfile/recipes uses "Superbase for the Commodore 128 version 3.0" as the database. Superbase was selected because the printing function was flexible enough to print the recipes onto 4 x 6 cards and the editing function was easy to use. A disk containing the necessary programs for Cardfile/recipes, and many recipes, is available on the TC-128/64 Companion disk. For those who want to create your own data disk, you must enter your own recipes.

There are other applications for a database of this type using 4 x 6 cards for hardcopy. Informational records, such as material used for repairs (identity and where to get it) or personal information (social security numbers, birthday gift choices, etc.) are ways to use this file. This approach to a database is good for records with fields that contain random data (unstructured). The recipe record has three fields that are structured, name, category (bread, soup, salad, etc.) and type of recipe. The other 18 fields are unstructured.

The following instructions are written for a novice user of Superbase (experts please be patient). After the instructions for creating Cardfile/recipes, there will be a few helpful hints and some of the problems, with solutions, that you may encounter.

PREPARING THE DATA DISK
Format a disk before loading Superbase. The disk must be formatted for the data (recipes) with a disk name that will identify it, something like "sb data disk 1" will do Next the Superbase programs listed in TC-128/64 must be saved to the disk, and then the recipes entered.

Load Superbase, put the formatted data disk in the drive and select RETURN from the prompts on the screen. Press RETURN again to get to menu 1 and again to get to menu 2. Press F5 (prog) to get the memo screen. The screen will be blank. Now type in the program START.P. On the line following line 190 enter SAVE"START (or save"start.p - either one works) and press RETURN. The program will be saved to the data disk.

Start.p on line 140 refers to a memo (help) that puts a menu on the screen listing the programs available. The menu gives a choice of using a program or going to menu 1. To create the memo, press CONTROL Q to return to menu 1 then press return for menu 2 and then press F7 (memo) to get to the memo screen. A prompt will ask for a name. Enter CARDFILE/RECIP (this filename must be the same as that on line 140 in start.p) and press RETURN. Type in a title for the menu (List of programs). Press RETURN to obtain a blank line to separate the title from the list (or between items).

Enter (without quotes)
"1. RECIPES - Print all recipes to 4 x 6 cards".

Then the next line
"2. SELECTRECIPE - Select recipe and print to 4 x 6 cards".

The third line is
"3. RECIPELIST - Print names of all recipes on 8 1/2 x 11 sheet".

The fourth line
"4. SORTRCPTLIST - Print list sorted by category, type on 8 1/2 x 11".

And the fifth line
"5. DISP - (Display) go to menu 1".

Insert a couple of blank lines and type in at the prompt,

"Enter a number (1 to 5) from the keyboard".

Press F1 then RUN/STOP to save the memo. The filenames with numbers must be in the same order as the filenames on line 190 of Start.p.

Now type and save the programs RECIPES.P, SELECTRECIPE.P, RECIPELIST.P, SORTRCPTLIST.P and DELETE H8.p as instructed in a previous paragraph for entering START.P After each program is saved, enter NEW to clear the screen then type in the next program. The disk will then have all the required programs.

Make a copy of the disk for future use and in case you make mistakes. Press CONTROL Q to return to menu 1 then press RETURN to get menu 2 and select F6 (maintain). From the maintain menu, select F6 (backup). Have a blank disk ready. Follow the prompts for formatting the disk and for a disk name enter - sb data disk 2,sc. The prompts will request a source disk (the disk that you will copy) and a destination disk (the blank disk that has just been formatted). Follow the prompts to copy the source disk. The copy will be useful, if more than one data disk is planned.

ENTERING RECIPES ON THE DATA DISK
The disk now has all the required programs. Before entering any recipes, the database name, file name, and record format must be entered. After saving the last program to disk, clear the screen (new) then press CONTROL Q to return to menu 1, then press the space bar to get into the command mode at the top of the screen. Type - database"cardfile" Press return,

press the space bar to get back to the command mode and type - file"recipes" then press return. The name of the database and the file are now on the disk.

A format for the record is next; refer to the Superbase manual pages R-8 through R-26 for detailed instructions. Select F2 (format) from menu 2. Type "name" (do not type in the quotes) then press F1 and a K to signify that the field is a KEY field. Press the right cursor key 30 times (a count will show at the top right corner of the screen). Use the cursor keys to add or reduce spaces. Press RETURN to set the field and a return to go to the next line. Enter the label "text1" (no quotes) then press F1 and a T for a text field. Press the cursor key 55 times and then RETURN to set the field The next 17 fields are TEXT fields labeled "text2" through "text18", they are 55 characters long. The last two are labeled "category" and "type" and are text fields 20 characters long. When finished press F1 then RUN/STOP to save the record format. The disk is now ready for the entry of recipes.

Plan how to type in the recipe. For example: if the recipe is long, enter two ingredients on one text line, use abbreviations for words (Tbsp for tablespoon etc.) and reduce the verbiage in the instructions.

Press CONTROL Q to return to menu 1. Press F1 (enter) and the format of the record will be on the screen. The cursor will be in the KEY field (name). Have a recipe ready. Enter the name of the recipe in the key field then press return to get to the next line (text1). From the keyboard, enter the ingredients and the instructions. Use the cursor keys and return to move about the screen. Press RETURN to get to the next line (field). Spaces are counted as characters. Avoid using symbols such +, :, /, ", =, etc. They will sometimes give problems during the printing of your recipe hardcopy. The two known safe characters are a period and a comma. A blank line can be obtained by just pressing RETURN at character 1 in the field.

The next two text fields are CATEGORY and TYPE. Category can be beverage, bread, dessert, entree, miscellaneous, salad, sauce, snacks, soup, etc. Type is the type of recipe within category. Within the category of entree could be the type of recipe such as meat, mixed dish, oriental, pasta, pie, pizza, vegetable, etc. and within dessert you could have types such as cake, cookie, fruit, gelatin, pie, pudding, etc. Create categories and types that will divide the file into groups of recipes that make the file easy to use. It is easy to revise the entries in category and type, if you do not like the first attempt.

A 3 1/2 INCH DISK AS A DATA DISK

Superbase 3.0 is compatible with the 1581 disk drive. The best way to make use of this feature when saving recipes, is to set the dip switch on the 1571 and 1581 drives to device 8; then turn OFF the 1581 when loading superbase and when loaded, shut OFF the 1571 and turn ON the 1581, before pressing return. The 3 1/2 inch disk will hold approximately 500 recipes.

First the 3 1/2 disk must be formatted with (HEADER"SB DATA DISK 1",ISB,D0,U9), if the 1581 is set as device 9. If the 1581 is set at device 8 and the 1571 drive is off, (HEADER"SB DATA DISK 1",ISB) will do.

Load Superbase, shut off the 1571 drive and turn on the 1581 drive (both set as device 8) and enter the 6 programs (as instructed for the 5 1/4 inch disk). Then enter the datafile and file names (cardfile and recipes). Next enter the file format. The disk is then ready to enter your recipes.

TEST THE PROGRAMS:

When several records have been saved. it is time to check out the programs. From menu 2 select F5 (prog). Enter load"filename and press return (entry of the ".p" following the filename is optional). Filename is the name of any of the five programs that were put on the disk. Have the printer ready. It is not necessary to have cards in the printer for this check. Press CONTROL Q to return to menu 1 and select F7 (execute) and the program that is in memory will run. If the printer does not work correctly. check the program for errors. If the recipe is not printed correctly. or completely, check the recipe for control characters such as, colons, quotes, etc. Check out all programs except "delete h8.p" Delete h8.p can be checked when it is necessary to delete some programs.

If you want to create a large file of recipes, plan on how to divide the recipes between two or more data disks. A 5 1/4 disk will hold about 200 recipes. A 3 1/2 disk will hold about 500 recipes.

PREPARING ADDITIONAL DATA DISKS.

First print a list of the recipes using SELECTRCPTLIST.P.; From menu 2 press F5 (prog) then enter NEW to delete the current program and enter - load"sortrcptlist. Return to menu 1 (have the printer ready) and press F7 (execute). This list will be a handy reference when dividing the data from one full disk onto two or more data disks.

FIRST METHOD:

The first method requires copies of the full disk and the deleting of the unwanted recipes for each disk. If more than 200 (or 500 for the 3 1/2 disk) recipes are to be entered a second disk must be made. The recipes should be broken into one or several different recipe categories for each disk. Do not fill the disk before creating a second disk. leave some room on the disk for a file and a program. Make as many copies of the disk as needed for the division of the files. Use the BACKUP command on the maintain menu. Maintain is on menu 2, press F6 (maintain) and then from the maintain menu press F6 (backup). Do not use a file copy program to copy the data disk! (a "nibbler" will work okay) Follow the prompts. When asked for a disk name enter "sb data disk 1" and for the second disk "sb data disk 2" and so on for as many data disks that there are planned. The disk ID should be unique for each disk. It is worthwhile to copy an extra disk for a backup in case of a mistake. The records of the categories not wanted are deleted from the disks in two ways, one by category and two individually.

To delete by category: From menu 2, press F5 (prog) type NEW to delete the current program and then enter the program DELETE H8.p. Save"delete hb to each of the above disks. Return to menu 1 and press F3 (find). The format template will appear on the screen with the cursor on the key line. Move the cursor to "category" and enter the name of the category to be deleted. Press RETURN twice and a search for the records in that category will take place. The name of the record will show on the screen as it is found. These names will be automatically stored as "h8list". Return to menu 1 and press F7 (execute) to activate the program "delete h8". Repeat this process to delete other categories not wanted on that disk. Insert another disk and delete the categories not wanted on that disk. There will be different categories on each disk when finished.

Remember, enter the database and file names using the "database" and "file" commands. Otherwise you might corrupt your data disks.

To delete individually: Insert one of the copied disks into the disk drive. From the menu 1 command line enter the database and file name. From menu 1 press F2 (select). From the select mode press F6 (first) and the first recipe on the key list will appear on the screen. Check the name of the category on the first record and if the record is to be deleted, press D to delete it. press Y to confirm the delete. Press N (next) to get to another record. Repeat the process until only the categories wanted remain on the disk. Repeat the process for the second (and third if required) disk.

SECOND METHOD
A second method for making a second data disk that would eliminate the work required to copy the disk and delete records. is to make a second disk in the same manner as the first. Use the prepared disk, the one that has only the programs on it. Do not divide the categories between disks, just make the second disk in the same way as the first. The same categories will be on both disks. This method is not as neat as the first but it is easier.

COPYING TO A 3 1/2 INCH DISK
Set the 1571 drive to device 8 and the 1581 drive to device 9 using the dip switches. Turn on the computer and both disk drives but do not load Superbase. After READY shows on the screen format the 3 1/2 inch disk (HEADER"SB DATA DISK 1",ISB,D0,U9) Next place the Superbase program disk into the 1571 drive press F2 for Dload and enter UTILITY128.

Utility128 will give you the option to "1. Recreate database" or "2. Copy non-database file". Press 2 to copy the programs from the 5 1/4 inch prepared disk to the 3 1/2 inch disk, empty but formatted. Enter the filename START.P press Return to select 8 and 0 from the next two prompts. The next prompt will be an 8. delete the eight (with the DEL key) and enter a 9 Press Return to select the next 0 and the copying will begin. The copying will be done when READY shows on the screen. If you have forgotten the other file names. press F3 to see the directory of drive 8. Then after Ready. enter RUN to return to the copy program. Repeat the above sequence for for all the files listed in the directory. except Cardfile and Recipes.

When all the programs have been copied. select 1 (Recreate Database), enter DATAFILE and RECIPES and the utility program will copy all the recipes on the disk in drive 8 to the disk in drive 9. The 3 1/2 inch disk will have all the recipes that were on the 5 1/4 inch disk and will have room for 300 more.

To use this disk. have both disk drives set as drive 8. With the 1581 drive OFF. load Superbase and when the prompts show on the screen turn OFF the 1571 drive and turn ON the 1581 drive. Press RETURN and the 3 1/2 inch disk will be the data disk.

COMMENTS AND HINTS:
If two or more data disks are being used. do not exchange disks without entering the database name and the file name for the second disk. After putting the second disk into the disk drive. return to menu 1. press the space bar, and enter DATABASE"CARDFILE. press return and next on menu 1 enter FILE"RECIPES and press return. An error message "INDEX MISMATCH" will show in the command box at the top of the screen if the database name is not entered and the "file selected" box will be blank until the file name is entered. Even if the disk is a copy of another disk, the database name and the file name must be entered.

Spaces entered at the beginning of a field in a record will not remain unless there is a character in the first column. Sometimes spaces to indent text are used to accent paragraphs. To enter spaces enter a " " (period) as the first character then the spaces The period is hardly noticeable when printed.

On the prepared disk of recipes from Parsec, the name of the recipe is entered in capital letters. When using capitals do not enter a shifted space between the words in the name. The shifted space creates a printing problem.

Category and type are used to arrange (sort) the recipes in a list or in the hardcopy file (a box of recipes) so that a recipe can be easily retrieved, replaced correctly, or found when missing. The printed list created by SORTRCPTLIST.P is invaluable as a double check of the recipes. If you enter a new recipe, write its name on the list as a reminder to print a card later.

The index cards in the hardcopy file are arranged with a category card followed by several type cards. If the category index card or the tab on the card is a different color than the type index card, it is easier to locate a recipe in the file. With the category and type printed at the bottom of the card, it is easier to replace the card correctly in the hardcopy file.

Those with the prepared disk of recipes from Parsec you will find under the category of bread, type yeast, that the names of a few recipes that begin with BB. BB means Bread Baker. The amounts of the ingredients are adjusted for an automatic Bread Baker (Panasonic) that makes one 1 lb. loaf. The same bread made manually (two loaves) may also be on another card, but, without BB before the name.

Printed copies of programs, memos or file definitions are made as follows.
Programs: From menu 2 press F5 (prog) and load the program. Press CONTROL Q to return to menu 1, then type PRINT:LIST:DISPLAY on the command line.

Memos: From menu 2 press F7 (memo) and type in the name of the memo. If the memo screen remains blank. the name of the memo was entered incorrectly or the memo does not exist (check the directory and try again). When the memo shows on the screen, press F1 then P to print the memo.

File definitions: From menu 1 type PRINT:STATUS:DISPLAY on the command line.

The cards used are Continuous Index Cards for computer printers, one wide. 100 lb. stock. 4 x 6 inches. Two known firms that make them are. Rediform (#06731) and Avery. Both are usually stocked at Office Supply Stores such as Office Max and Staples. The size of the card may be listed as 4 x 7, six inches plus the tractor feed width. If they can not be found in the local stores, ask them to order some for you.

I hope that Cardfile/recipes is useful for you. I have enjoyed it for two reasons; it helps when cooking and it gives me another use for my Commodore computer.

```
File   Definition : cardfile/recipes .
   #          name      type        Format/ Calculation
-------------------------------------------------------
   1    name            key         length  30
   2    text1           text        length  55
   3    text2           text        length  55
   4    text3           text        length  55
   5    text4           text        length  55
   6    text5           text        length  55
   7    text6           text        length  55
   8    text7           text        length  55
   9    text8           text        length  55
  10    text9           text        length  55
  11    text10          text        length  55
  12    text11          text        length  55
  13    text12          text        length  55
  14    text13          text        length  55
  15    text14          text        length  55
  16    text15          text        length  55
  17    text16          text        length  55
  18    text17          text        length  55
  19    text18          text        length  55
  20    category        text        length  20
  21    type            text        length  20
```

PROGRAM NAME: start.p

```
 5 rem copyrighted commercial software from TC-128/64 #34 by parsec inc
10 rem Superbase start program
20 rem *** set system parameters ***
30 lmarg 1:rmarg 80:rem margins
40 plen 66:tlen 60:rem page & text length
50 pdev 4:pdef 0:rem printer device 4 cbm code
60 lfeed 0:cont 1:rem no line feeds, continuous print
70 space 0:across:rem print across page
80 screen 0
90 rem *** SELECT DATABASE AND FILE  ***
100 database "cardfile":file "recipes"
110 rem  LOAD PROGRAM NAMES INTO ARRAY
120 for i=1to 5:read prg$(i):next
130 rem   MENU STARTS HERE
140 help "cardfile/recip":rem this file for the screen
150 display @0,5" ";
160 wait op:if(op<1)or(op>5)then 160:rem select number 1 thru 5
170 if op=5then menu
180 load prg$(op):rem load the selected program (1 thru 4)
190 data recipes,selectrecipe,recipelist,sortrcptlist disp
```

```
 5 rem copyrighted commercial software from TC-128/64 #34 by parsec inc.
10 rem:RECIPES.P
20 rem:This program will print out ALL recipes onto 4x6 cards
30 file "recipes":rem identifies file chosen
40 print:across:cont 1
50 plen 24:tlen 24:rem page and text length for 4 x 6 cards
60 output all @3,1[name]plus
70 @3,3[text1]@3,4[text2]@3,5[text3]plus
80 @3,6[text4]@3,7[text5]@3,8[text6]@3,9[text7]@3,10[text8]plus
90 @3,11[text9]@3,12[text10]@3,13[text11]@3,14[text12]@3,15[text13]plus
100 @3,16[text14]@3,17[text15]@3,18[text16]@3,19[text17]@3,20[text18]plus
110 @3,22[category]@30,22[type]
120 endreport:display:menu:plen:tlen:rem ends report and resets parameters
```

```
 5 rem copyrighted commercial software from TC-128/64 #34 by parsec inc.
10 rem  SELECTRECIPE.P
20 rem This program will print SELECTED recipes - One or more
30 rem To print two or more of the same recipe enter the [name] two or more
       times on the memo list.  Limit of 22 cards. If more run pgm again.
40 rem Press F1,STOP at end of list
50 maintaino"s0:h8recipelist":rem Removes the old h8recipelist
60 display @10,5"1. Enter list of recipes"@10,8"2. Press  F1,stop at end of
       list"@10,14"3. Press any key to continue":wait
70 memo "h8recipelist":rem this is the list of recipes entered
80 report "recipes":rem identifies file used
90 print:across:cont 1
100 plen 24:tlen 24
110 output from "h8recipelist"@3,1[name]plus:rem output from memo line 70
120 @3,3[text1]@3,4[text2]@3,5[text3]plus
130 @3,6[text4]@3,7[text5]@3,8[text6]@3,9[text7]@3,10[text8]plus
140 @3,11[text9]@3,12[text10]@3,13[text11]@3,14[text12]@3,15[text13]plus
150 @3,16[text14]@3,17[text15]@3,18[text16]@3,19[text17]@3,20[text18]plus
160 @3,22[category]@30,22[type]
170 endreport:display:menu:plen:tlen:rem resets parameters
```

```
 5 rem copyrighted commercial software from TC-128/64 #34 by parsec inc.
10 rem: RECIPELIST.P
20 rem:This program will print a list of ALL recipes by [name] on 8x11 sheet
30 file "recipes"
40 cr$=chr$(13):lfeed 0:space 0
50 print:across
60 output all [name]
70 endreport:display
```

```
 5 rem copyrighted commercial software from TC-128/64 #34 by parsec inc.
10 rem:SORTRCPTLIST.P
20 rem:This program will print a list of recipes sorted by [category]
30 rem:and [type] on an 8x11 sheet in three columns Category, Type and Recipe
       Name
40 lmarg 5:cr$=chr$(13):plen 66:tlen 62:rem set parameters
50 d$="--------------------------------------------------------------------------"
60 sort all on [category][type]to "sortrecipe":rem sort file
70 print:across:cont 1
80 report "recipes"
85 rem CREATE TITLE AND HEADING
90 title @20"CARDFILE - Recipes"cr$cr$@15"Recipe List by Category and Type"cr$
   cr$"Category"@16"Type"@32"Recipe Name"@1d$
95 rem PRINT LIST
100 detail from "sortrecipe"&[category]&@16[type]@32[name]
110 endreport:display:menu:lmarg
```

```
80 rem DELETE H8.P
90 file "recipes"
100 select from "h8list":eol menu
110 select d
120 goto 100
```

LEARNING MACHINE LANGUAGE PART 1

BY CRAIG T TAYLOR

INTRODUCTION
This column is intended for readers who wish to learn Machine Language (ML) and who have a rudimentary knowledge of programming. Knowing how to write Commodore Basic programs, for an example. Each column will progress farther and farther into the "guts" of the machine by explaining various programming tricks. various programming methods and styles, and will also examine a lot of the "inner" routines that are known but not properly documented anywhere.

A full listing of the 6502 operational codes and Kernal routines would take far too much space to give them the full treatment that they deserve. However, in the References sections are notes on which books. magazines, and software that document them in detail. It will help if you have such a reference by your side while reading through this article.

A. C-64 MEMORY

1. RAM
The Commodore 64 has 64 kilobytes (64K) of memory, hence the name C-64, and that 64k of RAM (random access memory) maps in from location 0 to 65535. RAM means that values can be stored in a location and retrieved intact. as long as the power does not go off, and that they can be used over and over again.

2. ROM
However, the computer has to know what to do upon initial power-up so there exists a special piece of memory called ROM. which stands for Read Only Memory In the Commodore 64 ROM there is also a section of memory called the KERNAL. which is a set of basic low-level routines needed by most machine language programs, and the BASIC interpreter. The ROM appears to "overlay" RAM in that the ROM value is returned whenever a memory location is read. ROM occupies locations 40960 to 49151 (the BASIC interpreter) and from 57344 to 65535 (the KERNAL).

3. I O BLOCK
Yet the computer has to have some way of communicating with the outside world. The 6502, the heart of the Commodore 64, treats Input/Output devices as memory locations and reserves a 4k (4096 bytes) block of memory for such I/O devices as the keyboard, sound chip (SID). and the video chip (VIC), as well as others. The I/O block occupies the range of 53248 to 57343 and also appears to "overlay" RAM.

4. LAYERED SCHEME
If we are overlaying the RAM and preventing access to it then why did Commodore add the extra RAM if it is not usable? It is - it is just not easy to use. There is a register located at memory location #1 which allows the programmer to switch RAM in, I/O out. the KERNAL out, RAM in. etc. Doing this from basic usually causes programming problems. This register will be examined in more detail later. More information about it can be found in the book "Mapping the C-64"

B. The 6502/6510 CPU

1. HISTORY
Originally Commodore used the 6502 in the KIM-1. a hobby computer that first appeared in the 1970's. Once the "techness" of computers disappeared and it was recognized individuals did not have to be a rocket scientists to use them, more personal computers came to market such as the Commodore PET. and the Vic-20.

The Vic-20 used a 6502 and much of the Commodore 64 is based on it. However, it had some severe shortcomings such as limited memory space. 4k. of which only 3.5k was available. The Commodore 64 actually works using a 6510 CPU. This 6510 CPU is 100% identical to the 6502 except for using locations 0 and 1 for selecting certain internal data ports as input or output locations. This helps the Commodore 64 to manage its memory-"banking" scheme. Note that when I refer to the 6502 below, I am also referring to the 6510.

2. INSTRUCTIONS

a. Data Storage and Retrieval
The instructions. LDA, LDX, LDY, STA, STX, STY, TAX, TAY, TXA, TYA

These instructions cover movement of data between the three registers of the 6502 and provide for transfer of numbers from the CPU registers to memory, from memory to registers, and movement of numbers from one register to another.

b. Stack Operations
Instructions: PHA, PHP, PLA, PLP, TSX. TXS

These six instructions allow for saving the processor's accumulator (.A). the current processor flags, recalling the accumulator or the processor state and also allows for manipulation of the stack pointer.

c. Arithmetic Operations
Instructions: ADC, SBC

The 6502 does not provide for any multiplication instructions and any additions or subtraction without carrying must be handled by first setting the carry flag appropriately. Multiplication and Division and other operations other than addition and subtraction can be handled through a series of instructions.

d. Increment Decrement Operations
Instructions: DEC, DEX, DEY, INC, INX, INY

These instructions will increment (increase by 1) or decrement (decrease by 1) a memory location or register.

e. Shift Operations
Instructions: ASL. LSR. ROL. ROR

These instructions will shift the bits in the accumulator or memory to the right or to the left, with some taking into account the value of the carry flag, other instructions shift a 0 into a bit place.

f. Logical Operations
Instructions: AND, BIT, CLC, CLD. CLI, CLV. CMP, CPX. CPY, EOR, ORA, SEC, SED, SEI

These operations allow specific bits within the accumulator or memory to be tested, compared, set,

reset, and toggled as well as setting flags within the processor.

g. Branch Jump Operations
Instructions: BCC, BCS, BEQ, BMI, BNE, BPL, BVC, BVS, JMP, JSR, RTI, RTS

These instructions allow the program to branch to certain statements if a certain flag is set. Other instructions allow a branch to a subroutine, return from a subroutine, return from an interrupt-type subroutine, and an unconditional jump to a memory location.

h. Miscellaneous
Instructions Covered: BRK, NOP

These two instructions are primarily used during debugging. BRK generates a "soft" - Interrupt request - usually dumping the program back to the machine language monitor, while NOP performs a "No-Operation" instruction and is usually used to overwrite, or to skip over, certain program instructions in memory.

3. Special Treatment of Page 0 and Page 1
A "page" in 6502 programming is defined as a group of 256 bytes. So, memory location numbers 0-255 would be a page, memory locations 256-511 would be another, etc.

The 6502 has a special addressing mode called Zero-Page addressing which allows it to access memory in locations 0-255 much faster. Hence, often used variables are stored here and space for other data is often very small. Only a few locations are available for use by the basic programmer, and a little bit more for the machine language programmer.

Page 1 is where the "stack" is located. The stack is a group of memory locations from 256 to 511 where data can be stored temporarily and it helps keep track of where in memory to return to after calling a subroutine. Using this page of memory is not recommended, although when using certain advanced techniques this can be done. For now, do not try storing any values in these memory locations.

4. Addressing Modes
The 6502 is made even more powerful by its various addressing modes The .X and .Y registers often act as indexes into memory where data can be manipulated instead of using an absolute set location. All of the addressing modes available to the 6502 are listed below, with a brief explanation:

a. Implied Addressing Format: MNE
The instructions in this addressing mode are ones like TAX, SEC etc. where no additional data is needed.

b. Immediate Addressing Format: MNE $xx
Typically these are the "load register with numerical value" instructions (ie: LDA #$xx). This addressing mode allows for an immediate value to follow the Mnemonic command.

c. Absolute Addressing Format: MNE $xxxx
Direct reference to a memory location is made through this addressing mode. Note that the address in 6502 instructions are usually stored low byte, high byte so the value $C000 would be stored as $00 $C0.

d. Zero-Page Addressing Format: MNE $xx
This is similar to Absolute Addressing except that locations in zero page are referenced and hence only one byte needs to be given instead of two.

e. Relative Addressing Format: MNE <offset>
This mode is used exclusively with the branch instructions and allows an offset of -128 bytes to +127 from the base of the instruction. Most assemblers instead of making you specify the offset will calculate it for you automatically if given a label name for the target address.

f. Indexed Addressing
Format: MNE $xxxx,<.x.y>
This allows referencing of a memory offset by the value in the .X or .Y register - allowing strings to be read one by one by loading the value offset by .X, then incrementing the .X register for the next value. This mode is also used in a wide variety ways - the address used is effectively $xxxx + .X or $xxxx + .Y

g. Indexed Indirect Addressing Format: MNE ($xx,.x)
This mode requires a zero page location and after getting the two byte values at $xx + .x it returns the value that is at that location. Example:

Memory: $80 = $00 $c0 $00 c1

If .x = 0 then LDA ($80,x) would return the equivalent of LDA $c000

If .x = 2 then LDA ($80,x) would return the equivalent of LDA $c100

h. Indirect Indexed Addressing Format: MNE ($xx),.y
This is similar to the above addressing mode except that it takes the two bytes $xx and $xx+1 and then from that memory address adds register .y to obtain the value. Assuming we have the same Memory values in $80 as above then:

If .y = 0 then LDA ($80),y would return the equivalent of LDA $c000

If .y = 5 then LDA ($80),y would return the equivalent of LDA $c005

5. Binary Numbers
The computer's RAM Memory can be broken down into bytes and then further broken down into the basic fundamental unit called a bit. Because computers are digital, each bit can have only one of two possible values: 0 or 1. Within in the computer, instead of 0 and 1's, we have either the presence of voltage or the absence of it. If we combine eight of these bits then we can form a byte. This byte can have any of 256 possible values (0 to 255), by changing the values of the individual bits. Example:

0 0 0 0 0 0 0 0 in binary is 0 in decimal
0 0 0 0 0 0 0 1 in binary is 1 in decimal
0 0 0 0 0 0 1 0 in binary is 2 in decimal
0 0 0 0 0 0 1 1 in binary is 3 in decimal
0 0 0 0 0 1 0 0 in binary is 4 in decimal
0 0 0 0 0 1 0 1 in binary is 5 in decimal.

We can continue until we get the number 1 1 1 1 1 1 1 1 in binary which is 255 in decimal How do we know that it is? Well, we could list out all the preceding numbers like above - but there is an easier way. If you place the values 128 64 32 16 8 4 2 1 above each position (note that each bit position within the byte is one half the numerical value of the previous bit position on the left) and then only add the values with

The Video Digitizer
A Review

By John W. Brown

The Video Digitizer is a means of capturing images from various types of video sources such as video cameras, Video Cassette Recorders, and Camcorders.

The Video Digitizer consists of one ROM sized cartridge that plugs into the User Port and three programs. Digison, Digifox, and Digimulti. It also comes with three gels to be used for capturing color pictures, which we will cover later in this article. Because we will be reviewing the Pagefox cartridge in detail in a later issue, I will only cover the features I find most exciting, unusual, or not offered by other programs.

The first thing you have to ask yourself before buying the Video Digitizer is if you need a video digitizer or if you need a handscanner. The video digitizers for the Commodore computers are generally cheaper than the HandyScanner 64, but both service a market segment where each excels.

You can not beat the HandyScanner 64 for maximum resolution when reproducing a small photo or small printed work, such as line art. Then again, you can not run a HandyScanner 64 over someone's dog with pleasant results, though you can digitize the dog easily with the Video Digitizer. The HandyScanner does not produce color pictures which can be easily imported into the most popular paint programs for the C-64, the Video Digitizer does.

If you are buying solely on price, you should consider the TOTAL cost of purchasing the Video Digitizer. You will need a camera and tripod, or a Camcorder and tripod, or a four head VCR with a steady freeze frame. If you do not already own these items, the additional purchasing costs can far exceed the cost of the HandyScanner 64.

Although the Video Digitizer says in the manual it will not work with freeze frames on VCRs, if I understand the manual correctly, it should work with the newer four head VCRs which produce a clear and steady video signal. I know my Computer Eyes could never produce an acceptable scan from my (older) VCR's freeze frame.

The installation of the Video Digitizer is fairly straight forward. With the power to the computer off, you plug the Video Digitizer cartridge into the User Port on the left hand side of your Commodore 64/128. You load and run the first program on the disk, Digison, and you are ready to scan. For a more detailed explanation of how the scanning is done and how to set your system up, see the article at the end of the review.

The program disk for the Video Digitizer is unprotected so you can make backup copies. Since the program can be run from any type of disk drive with any device number, it is possible to put all your files on a 1581 set as device #9 to take advantage of the increased storage space. It worked fine with my CMD FD-4000 and CMD HD.

The cartridge has two adjustments, one for image width and one for contrast. There is no brightness control knob. Brightness control is done through software. The threshold level for determining what pixel will be off (white) or on (black) can be set manually or done automatically by the computer. I could not figure out exactly what the "width" control knob did. All I can guess is that it it has something to do with the width of the video signal and plays no part in how wide the captured image will be. Turned all the way down it hung the computer. The contrast knob is a something you only have to set once for your individual setup and once done it is to be left alone. Which is good because these knobs are actually recessed screws and are not easy to reach or adjust, especially on a C-128D.

DIGISON

Digison is the part of the package that produces the Black and White capture of images and converts well to other formats that use either HiRes mode such as Doodle and FGM. It also converts well to geoPaint. This is strictly a B&W paint program, though the work area is a big 640x400 pixel area. Using the program "Digifox" provided on the disk, the work area is 640x800 pixels with the addition of the Pagefox Desk Top Publishing cartridge. There are also some enhancements to the Pagefox software when using Digifox, such as being able to draw lines and circles wider than the screen and having the 24 pin drivers resident.

One of the best things about the HandyScanner 64 and Pagefox are the full featured paint programs that come with them. No disappointment here, the Video Digitizer has almost the exact same program. All three programs included with the HandyScanner 64, Pagefox, and Video Digitizer function the same way and for the most part, their menus are the same, except for a few minor changes. (see the menu pictures in red ink on page 45). The Video Digitizer program works with a C-1351 mouse in joystick port#0, a joystick in port#1, or the keyboard. As you can see from the menu pictures, menu one contains all the things you expect in a good paint program such as a small lined freehand, large line freehand, lines, rectangles, circles, a "fill" or paint feature for large areas, a really good "spray can", a means of moving large graphic objects, the ability to place text on the screen, "brush" and stamp utilities, a large eraser, a zoom mode, a tool for large scale reduction of a screen, and a x-y coordinate read out.

This is a very nice paint program that is smooth to use and easy to understand. Almost all the commands can also be called by one letter keyboard commands. One of the best features, which few programs can match, is the MOVE command.

With the MOVE command, all you do is click on the icon, go to the area of the screen you want to clip, click once to set the beginning corner of the MOVE box, click a second time to set the ending corner of the MOVE box, and a copy of the image will appear under your cursor in a few seconds. You are then free to move the image anywhere within the 640x400 pixel area and you can place it within any 320x200 quadrant, using the mouse or joystick to position it. Once it is in place, you get to use the other excellent part of this tool, the "move one pixel" icon. By clicking on this icon immediately after depositing a MOVED graphic onto the page, you can scroll it left, right, up or down a pixel at a time. This makes lining up pieces of a graphic, such as left and right halves, or tops and bottoms, a cinch.

The "MOVE" command can also rotate or flip any image you are moving. How the image will be treated is determined by where the beginning and ending corners of the move window were set. Once you get the hang of this, the moving and flipping of images, it becomes very easy and natural and does not require the picking of multiple menu items to accomplish the task like other paint programs.

The "MOVE" command also contains another handy feature. After setting one corner of the "move" box, if you hold the fire button after the second click, you can reduce the image! This is a fantastic feature for scaling something to fit within a small area.

The "TEXT" function is good and allows the use of almost any Pagefox font to place nice looking text with your captured images. You can adjust the space between characters once the character set is loaded. Using the cursor keys you can flip a line of text in 90 degree increments. What I like best about this function is that the line of text does not get placed until you press the fire button. You so can type a line of text and move it about the screen for accurate placement. With other paint programs you have to hope you have the text started in the right position before you begin typing. Other paint programs, including geoPaint, do not offer the ability to rotate fonts easily.

PATTERNS

You can load and use any pattern you want or even capture them from a scanned picture. You can have a company logo as a pattern, you can use letters from a font as a pattern, you can use just about any image that will fit within the pixel confines of the pattern area.

ADDITIONAL FEATURES

There are other features, such as "file" and "plane" which help remove stray pixels and picture "noise" from a video scan. I found that using the zoom editor was the most effective feature for this and resulted in nicer looking pictures. On page 45 (the red ink) see the difference between my scanned picture as it was originally and how it looked after the background was eliminated and the image was flipped.

DIGITIZING

The Video Digitizer offers three types of Black and White digitizing. A simple B&W capture mode and two gray scale modes. Each digital capture lasts about eight seconds and has to be a picture of an unmoving object or has to be a steady unchanging video signal. The resulting scan is about 400*230 pixels and the image is placed starting in the middle of the first 320x200 pixel screen. I found this kind of irksome at times because it makes aligning multiple images a bit tough when you want to piece together a very wide picture.

With the gray scale - dithering modes you can choose two dithering settings, either 7 levels or 13 levels of gray scale dithering. In each you can set the threshold mark where white or black will be determined If you do not answer the prompt for the threshold setting it will be performed automatically by the software, which usually gives superior results.

PRINTOUTS

To give you an idea of how good of a job the Video Digitizer does, compared to the hand scanner, I have included various captures and examples of what you can accomplish. All the images on page 45 (the red one) or page 46 (the blue one) were produced in one way or another with the Digison, Digifox, or Digimulti programs. All the images were printed on the Epson Action Laser II using either the Hewlett Packard (Laserjet II) mode or using the Epson LQ 24 pin emulation mode, where noted.

The Video Digitizer menus were grabbed with the Video Digitizer software and converted to geoPaint documents. The top picture of the tiger was HandyScanned and printed out under geoPaint at 150 dots per inch. The second picture of the tiger was captured using the Video Digitizer using a 7 level gray scale capture. The third picture of the tiger, the one at the bottom, was produced with a plain Black and White capture with a threshold setting of 9. The last two pictures of the tigers were printed at 100 D.P.I.

As you can see the HandyScanner 64 produces far superior results for smaller images you want to print unchanged at high resolutions. Although, for large objects that you want displayed on the screen or for images you want to include in your own programs, the Video Digitizer might be the way to go. If you want to reproduce color images with color on screen, the Video Digitizer is the only way to go. The capturing of black and white art of "simple" images like silhouettes will yield roughly the same results on screen using either piece of hardware.

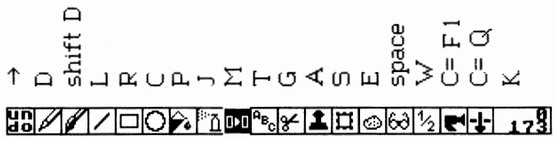
On page 45 is a 7 level, gray scaled image, from the front of a Dr.Who magazine. Considering the wide range of colors, large image size to capture, and the glossy surface this came out excellent. I am only using the magazine as an example because I did not have any wide color photographs to use instead.

I have included my picture on page 45 as an example of how you can clean up an image and flip it. I removed the background by hand using the "sponge" and "zoom" modes. It took about 15 minutes.

On page 46, the one with the blue ink, we come to some of the fun things you can easily do with the Video Digitizer that would be a chore with the HandyScanner because you would need printed images first. I have the Dr.Who figures from Dapol so I made an animated scene of a Dalek chasing and shooting a Cyberman in the back. Since the Cyberman has to come off his feet and I did not want to hang a wire, that would have to be edited out later. I opted for laying the models on their sides on top of a white sheet of 8x14 paper. After each image was captured and saved to disk I moved the Cyberman's arms and legs slightly. If you wanted to go all out you could draw in laser beams, shells, explosions, and even add sound in either C-64 or C-128 mode with the programs provided in this issue! Animated this looks pretty cool! The images were printed at 150 dots per inch.

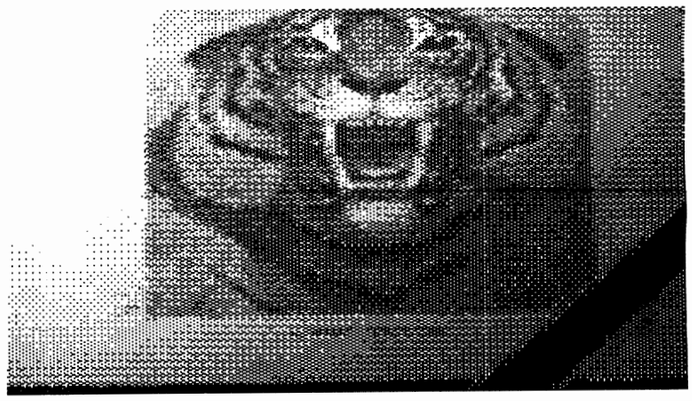
As an EXAMPLE, the next image is of the Dr.Who logo captured with Digimulti from the cover of the magazine. The first screen (top left) is the digital capture using the red gel in front of the camera lens (screen#1), the top right corner (screen#2) is the digital capture using the green gel, and the lower left corner (screen#3) is the capture using the blue gel. Screen #4, the lower right quadrant, is the result of all three screens being combined together. On screen this produces a color image, that with touching up, would be usable. You can save screen number four as a Koala file or you can save the whole graphic with all four screens for later use. My later use was to load

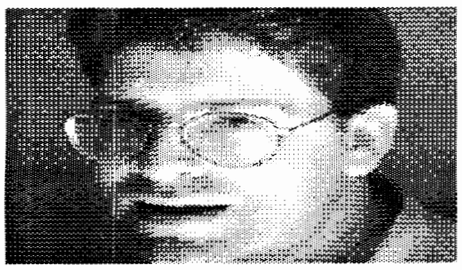
Video Digitizer 01



Video Digitizer 02

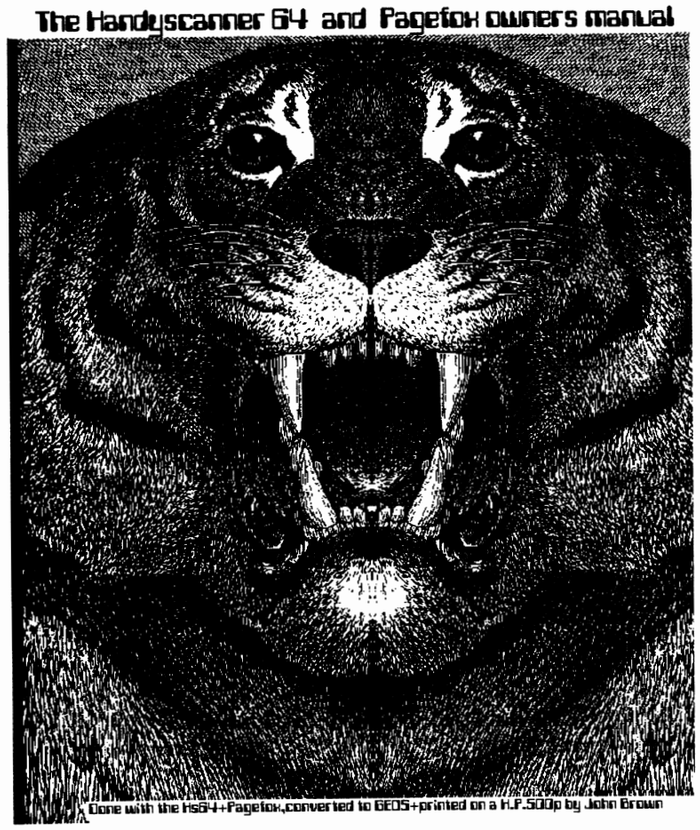
Video Digitizer 03



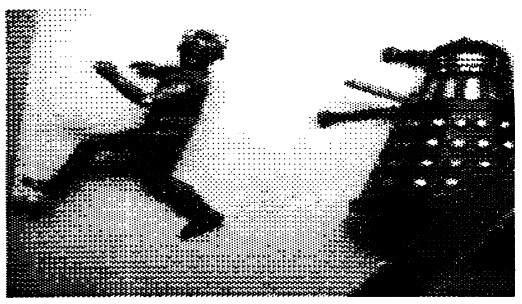


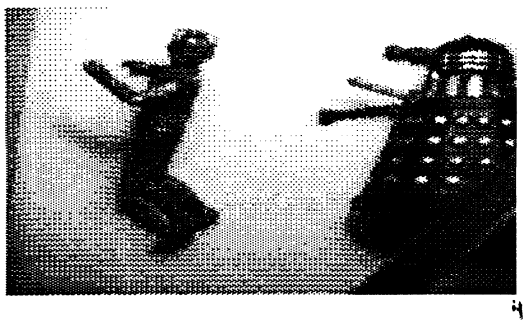
C-128.JBEE on GEnie

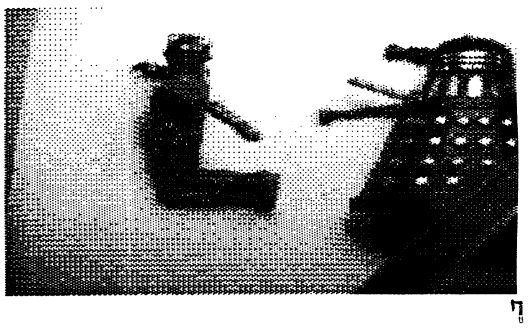
The Handyscanner 64 and Pagefox owners manual

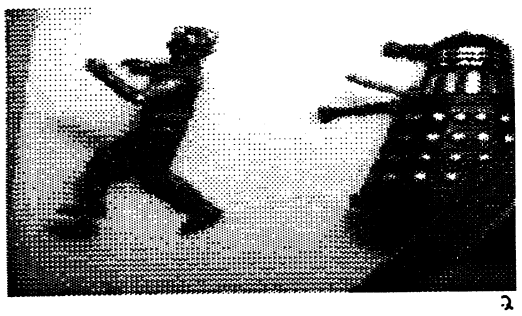


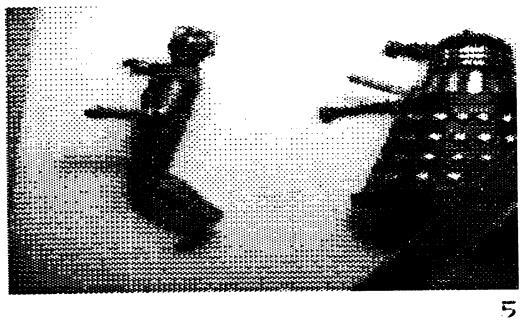
Done with the HS64+Pagefox,converted to GEOS+printed on a H.P.500p by John Brown

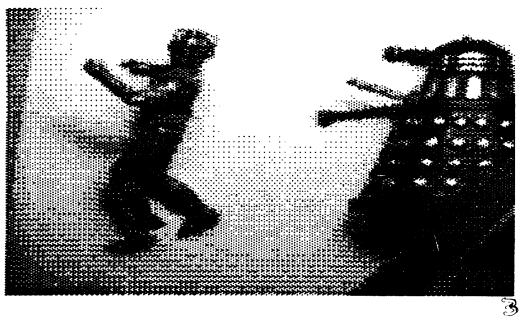


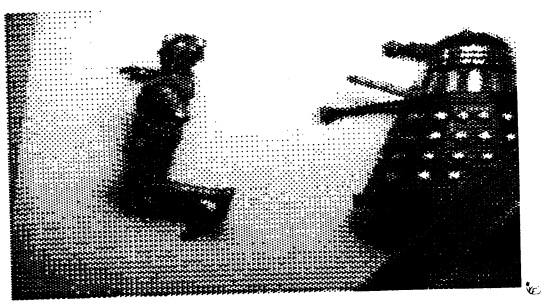


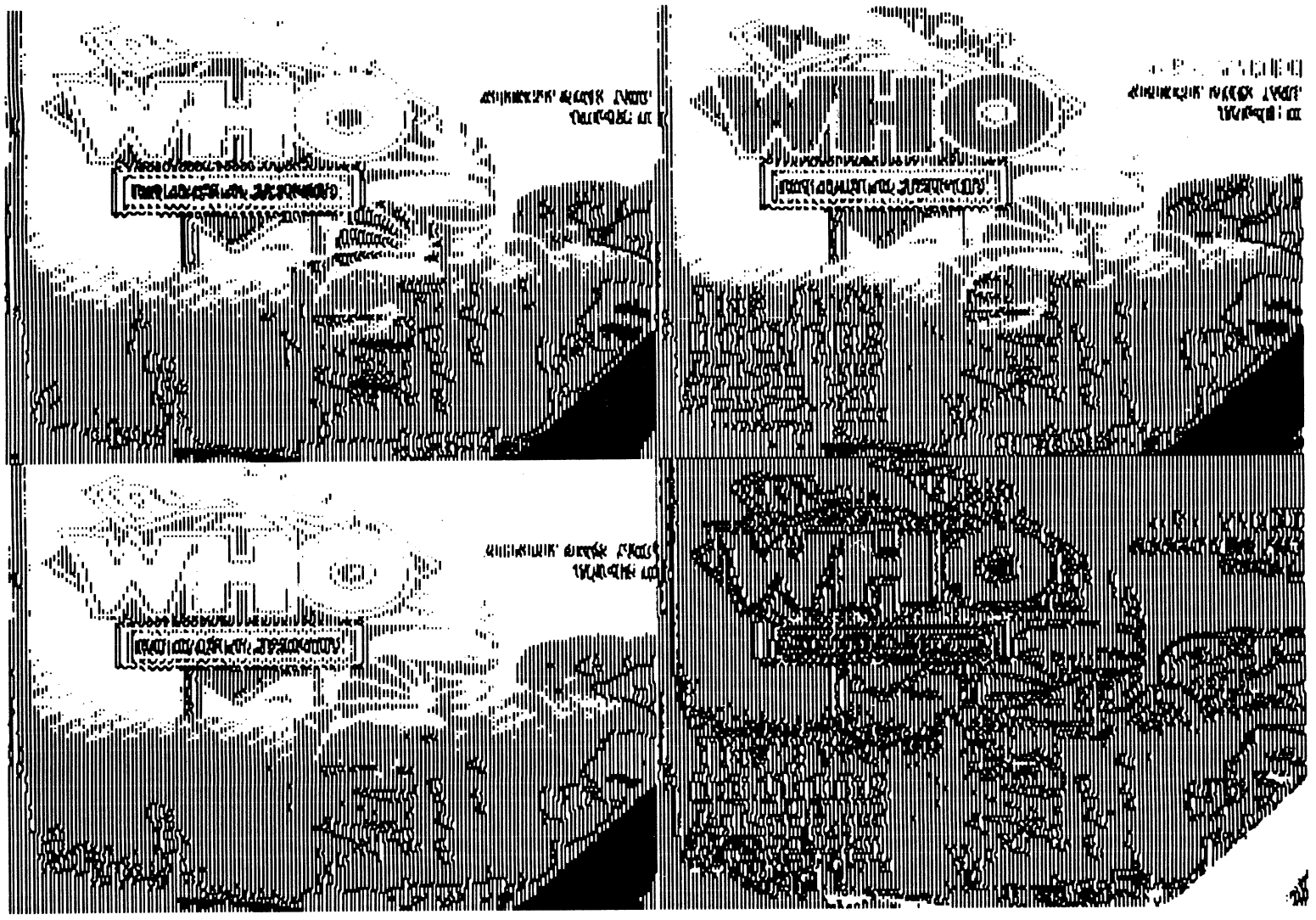












it into Digifox and print it out in B&W using the 24 pin printer drivers, on our laser printer. A real 24 pin printer would produce a better 2D picture.

DIGIMULTI
DigiMulti is probably the most unique and useful program on the disk and the best reason to buy the Video Digitizer instead of a scanner. It allows you to have various levels of gray scale captures, which would be even more useful if color printer drivers were provided.

The program produces color scans by capturing three color bitmaps red, green, blue, and then merging them together for the final color picture.

First you set to screen one, which is one of four screen areas. You hold the red gel in front of the camera lens and start the scan which takes about four seconds. Then you set to the second screen, hold the green gel in front of the lens and start the scan. When finished you start the third scan and hold the blue gel in front of the camera lens. Once all three color scans are done, you move to the four screen, hit the scan key, and the software combines all three color scans into one color picture.

The color picture can then be saved to disk in many of the popular drawing program formats such as Koala and OCP Art Studio where you can further edit the program. You can also save the whole graphic area to disk and edit it in Digison. The graphic bitmaps convert fine to GEOS. There are various tools to balance the colors or to change the colors globally within the scanned picture. This is helpful if you want to change all the reds to blacks, all the blues to greens, etc. You can eliminate colors in the picture by changing it to an existing color, such as black, or "blend" a feature into the background by using the background's color.

FILENAMES
Besides providing information about about how to save the images in various picture formats, the manual also provides detailed information on how to name the files. To save a koala file with that "spade" character needed at the beginning of the filename, all you have to do is start the filename with a question mark and the software does the rest. This eliminates having to have a separate program to rename the files later on.

WHERE'S THE BEEF?
I was left wondering "where's the beef". There is no way, as far as I could tell, to use anything other than drive #8 and no way to send disk commands. This is a step backwards today, especially since Digison supports both. The most disappointing thing ... there are no COLOR printer drivers. If you are going to capture color images which exceed the color resolution of the C-64, why not provide color printer drivers? A lot of people both in the US and overseas own color printers, especially the Star NX-1000R series, and the color inkjets have been popular for some time. Even software that used multiple color ribbons and multiple passes of the paper would have been fine.

One idea that would be worth looking into would be to print all your color images out in a specific position on the paper, make multiple copies, and then run it through the printer again for the black text using your word processor or maybe even the Pagefox DTP cartridge. This would be great for cards, small newsletters, or labels.

I consider the lack of color printer drivers and not being able to position and print out multiple copies of color images a serious oversight.

TIPS
Some of the things not readily available in the manual is the fact you can scroll around the screen area (screens 1-4) with the cursor keys. You can also load graphics into the area you are viewing, even if it is not centered within a quadrant! This allows you to stagger images over each other for interesting effects.

SCREEN SAVES SUPPORTED
You are allowed to save a single screen, which is the last screen you viewed, or all the screens in the graphic area as one file for later use. These graphic formats are also supported and you would usually set to screen four that contains the final color image and then hit the save key.

Hi-Eddi
Cheese
Koala Paint
Paint Magic
Blazing Paddles
Vidcom
Art Studio
Doodle

MANUAL
The manual is a "dot matrix dox" and very hard to read or follow. The German manual has been translated from German to English and left at that. To be fair, I was suppose to redo the manual like I did for the Handyscanner 64 but just could not find the spare time. If you want to use the paint package I would suggest ordering the Handyscanner manual I wrote. The original masters were laser printed so you will not go blind reading it. The manual is $4.95 from RIO and will save you a lot of reading, frustration, and time.

COMPARISONS
I have used many devices for capturing video images on the C-64. The most widely known one I have used is Computer Eyes. In my opinion, the smaller hardware package of the Video Digitizer beats Computer Eyes hands down on graphic format support, features, software, easy of use, and results.

So it earns a (weak) A- instead of a solid A due to the very poorly done manual that would have done in a lesser product and because of the lack of color printer drivers and other features for Digimulti. If you do need this product you will not regret the purchase and will enjoy it.

Grade: A-

Cost $149.95 Summer Special
Reg. Price $249.95
Six month warranty

ITEMS USED

VIDEO DIGITIZER - $149.95
1-702-454-7700
RIO Computers
3310 Berwyck St
Las Vegas NV 89121

CAMERA - $250.00
Panasonic WV-1410, Closed Circuit B&W Video - Security Camera. Available through various mail orders sources and computer dealers. I believe DAMARK and COMB use to carry such cameras.

CAMERA TRIPOD - Cost $49.99
Radio Shack
Cat# 16-2017 RST-84V

BNC-Phono Adapter (gold plated) - Cost $4.19
Radio Shack
Cat# 278-303a

6 Foot Gold AV/Cable - Cost $5.69
Male RCA to Male RCA
Cat# 15-1519

VIDEO CAMERA SETUP

BY JOHN W. BROWN

Make sure you have a well lit and clean place to work. The area needed bv the tripod will be 18 inches wide bv 29 inches long. I set my camera and tripod on the left hand side of the computer so I could use a short video cable.

First, take the camera tripod out of the box. Notice the various tags on the knobs which control the camera angles and platform height. Now remove them. We are going to leave the tripod at its lowest height for our work. The hook at the bottom is for storing and hanging the tripod up when it is not in use. Next spread the legs out and push downward until the support arms for the legs lock in place and the center ring is seated down firmly as far as it can go. The level, the piece of glass with a bubble floating in it, should be on the left hand side of the tripod mount. If not, turn the tripod head around.

On the very top of the tripod there is a platform where the camera will be mounted. It is the part with the rubber pad and two metal studs sticking out of it. Remove it bv pressing on the lever near the rounded edge, it is spring loaded so you have to hold the lever while lifting up the small platform. Now take your camera out of the box and flip it over.

Lift the C-ring on the underside of the platform pad you removed and screw it into the threaded hole provided on the bottom of the camera. The small stud, the one without the threads, will face towards the front of the camera and is inserted into the hole provided. Firmly with two fingers tighten the C-ring - screw assembly as tight as you can (if you are a weight lifter use your thumb and baby finger ;). Push the ring down so it is laying flat on top of the screw assemblv

Make sure all the control knobs on the tripod are firmly turned. You are going to put a heavv camera onto it and you do not want it to move or flop around.

Now mount the camera on the platform bv placing the mounting pad on the top of the tripod and moving the lever to the side. Once mounted, move the lever under the platform, back into its original position. It will move by itself. it is spring loaded. But, make sure the lever is in place by firmly pressing on the lever. but do not force it.

Now remove the BNC connector and cable from the packages. Place the BNC connector on the camera bv lining up the slot and pushing forward. while turning it. Uncoil the cable and plug it into the back of the camera.

Now turn the camera around by loosening the second knob from the near the middle of the tripod. When the camera lens is facing forward, towards the front of the computer, tighten the knob. When turning the camera make sure the cable and power cord do not become entangled or hit a connector port on the back of the computer.

Now angle the camera down until it is in a 90 degree angle towards the desk. The lens will be facing directly at the desk top.

With the electrical power and computer off. plug the Video Digitizer cartridge into the User port on the left hand side your your computer. Now plug the other free end of the RCA cable attached to the camera into the Video Digitizer cartridge (V.D.C.). Since the cartridge does not sit level on the desk, you might want to plug the cable into the Video Digitizer before inserting it into the computer. This might save some wear and tear on the Video Digitizer and your User port. C-128D owners might have a tough time doing this since the RCA plug is almost directly underneath the C-128D power cord. Now turn the computer system on and enter C-64 mode. Now turn the camera and your light source on. I used a small table lamp to provide indirect lighting, you could also use a light table.

(DR. OCTAL TIPS CONTINUED FROM PAGE 48
column screen is filled with garbage The solution: When you want to use the reset button to exit GEOS. press the C= kev and the reset button to enter C-64 mode, once there type Poke 996,96 <RETURN> Then press the reset button again to return to a normal C-128 mode.

Tip &0015
USING CP/M and the CMD HD
From: John W. Brown
Though it is stated in the manual, manv people forget to ALWAYS create "1581C" partitions for CP/M mode on their hard drives. This is easv to forget in the middle of a gut wrenching low level format or if vou have not read the manual ; - D

Tip &0016
USING CP/M and the CMD HD
From: John W. Brown
Always have your boot or menu program sent the @pU command over the serial bus to turn off the Ramlink s fast parallel access before running CP/M It will save you some typing and it will also save vou from having to reset the computer, when the drive locks up, if you forget.

a one in their bit location then 11010110 would be 214

```
128 64 32 16 8  4  2  1
 1  1  0  1  0  1  1  0
```

in binary = 128+64+16+4+2 = 214 in decimal.

Binary numbers are used in Machine Language programming quite often The classic rotate and add algorithm that is usually implemented for multiplication requires a good knowledge of how bits and bytes relate to values.

6. Hexadecimal Numbers

While binary numbers could only hold the values 0 and 1, hexadecimal numbers can hold the numbers 0-9, 10, 11, 12, 13, 14, and 15 in each digit. Instead of using "10" "11" "12" etc... we use the letters A B C D E and F to stand for 10, 11, 12, 13, 14 and 15. So we can count in hexadecimal (also called "hex") by:

0	0	10	A	20	14
1	1	11	B	21	15
2	2	12	C	22	16
3	3	13	D	23	17
4	4	14	E	24	18
5	5	15	F	25	19
6	6	16	10	26	1A
7	7	17	11	27	1B
8	8	18	12	28	1C
9	9	19	13	29	1D

But given the number $C134 how do we find the decimal value?? Note the dollar sign $ in front of the number. the $ sign is usually used to denote a number is hexadecimal as $1000 can also be a number but is not the same as 1000.

We do it in the same manner that we did with Binary numbers except with some different values.

(Just think 16 to the 3rd, to the 2nd....)

```
x4096  x256 x16  x1
  $C    1    3    4
```

where C = 12 so we get 12x4096 + 1x256 + 3x16 + 4x1 = 49460.

Hex numbers are used to represent memory locations frequently in machine language. Because of their compact format they make values easier to remember and use instead of a 5 digit decimal number.

C. C=64 Kernal Routines

1. Their purpose and use

When Commodore introduced the PET series, long before the Vic-20 and Commodore 64 and 128, they set in the highest memory locations a series of jumps (tables) to other routines so the programmers did not need to bother checking if any revisions had been made within the ROM's memory locations. They were assured that the address (index) they were jumping to, would indeed, be the address to do that specific function in machine language. The KERNAL handles such low-level chores as serial devices, screen, scanning the keyboard, updating and reading the system clock, etc...

2. Common Kernal Routines Used and What They Are

Some of the more common and often used Kernal routines with a short description are listed below:

```
READSS $FFB7    : Return I/O status byte.
 Registers In   : None.
 Registers Out  : .A = status byte.

SETLFS $FFBA    : Set logical file #, device #,
                  secondary # for I/O.
 Registers In   : .A = logical file #, .X = device #, .Y
                  = secondary #
 Registers Out  : None.

SETNAM $FFBD    : Sets pointer to filename in
                  preparation for OPEN.
 Registers In   : .A = string length, .XY = string
                  address.
 Registers Out  : None.
 Note           : To specify no filename specify a
                  length of 0.

OPEN $FFC0      : Open up file that has been setup
                  by SETNAM,SETLFS
 Registers In   : None.
 Registers Out  : .A = error code, .X,.Y destroyed.
                  .C = 1 if error.

CLOSE $FFC3     : Close a logical file.
 Registers In   : .A = logical file #
 Registers Out  : .A = error code, .X, Y destroyed
                  .C = 1 if error

CHKIN $FFC6     : Sets input channel.
 Registers In   : .X = logical file #.
 Registers Out  : .A = error code, .X,.Y destroyed.
                  .C = 1 if error

CHKOUT $FFC9    : Sets output channel.
 Registers In   : .X = logical file #
 Registers Out  : .A = error code, .X, Y destroyed.
                  .C = 1 if error

CLRCH $FFCC     : Restore default input and output
                  channels.
 Registers In   : None.
 Registers Out  : .A, .X used.


BASIN $FFCF     : Read character from current input
                  channel.

                  Cassette - Returned one character a
                             time from cassette buffer.
                  Rs-232   - Return one character at a
                             time, waiting until
                             character is ready.
                  Serial   - Returned one character at
                             time, waiting if nessc.
                  Screen   - Read from current cursor
                             position.
                  Keyboard - Read characters as a string
                             and then return them
                             individually upon each call
                             until all characters have
                             been passed ($0d is the
                             EOL).
 Registers In   : None.
 Registers Out  : .A = character or error code, .C = 1
                  if error.

BSOUT aka
CHROUT $FFD2    : Output byte to current channel
 Registers In   : .A = Byte
 Registers Out  : .C = 1 if ERROR (examine READST)
```

```
LOAD $FFD5      : Loads file into memory (setup via
                  SETLFS,SETNAM)..
 Registers In   : .A = 0 - Load, Non-0 = Verify
                  .XY = load address (if secondary
                   address = 0)
 Registers Out  : .A = error code .C = 1 if error.
                   XY = ending address

SAVE $FFD8      : Save section of memory to a file.
 Registers In   : .A = Z-page ptr to start address
                  .XY = end address
 Registers Out  : .A = error code, .C = 1 if error.
                  .XY = used.

GETIN $FFE4     : Read buffered data from file.
                  Keyboard - Read from keyboard buffer,
                   else return null ($00).
                  RS-232   - Read from RS-232 buffer,
                   else null is returned.
                  Serial   - See BASIN
                  Cassette - See BASIN
                  Screen   - See BASIN
 Registers In   : None.
 Registers Out  : .A = character, .C = 1 if error.
                  .XY = used.

PLOT $FFF0      : Read or set cursor position.
 Registers In   : .C = 1 (Read) .C = 0 (Set)
                     .X = Col    None.
                     .Y = Row
 Registers Out  : .C = 1 (Read)       .C = 0 (Set)
                    .X = Current Col  None.
                    .Y = Current Row
```

D. Sample Program

1. Notes.

This sample program will simulate a bouncing ball on the screen. It starts out by first clearing the screen, setting the position and direction of the ball, and then performing an endless loop which consists of drawing the character (a lowercase "O") and then pausing briefly and erasing it by displaying a space at that location. The position is then updated and the keyboard is checked to see if the "Q" key was pressed. If it was not then the loop continues.

I chose this sample program instead of the typical ones that display a name over and over because this demonstrates the length and detail to which machine language has to be planned out. It also does something a little bit more entertaining : -) The text on the right side after the semi-colons are comments. Almost all assemblers will let you insert comments into the source code on what the instructions are doing, notes to yourself, and so on, if you put a semi-colon after the instruction or on a blank line. I have used these comments to describe what we are doing.

A careful reading of this program, typing it in, and changing one thing at a time will help you to understand Machine Language better. Ask yourself when you make changes, What should it do? What did it do? Why did it or why did it not do that?

2. The Program "ballbounce"

In C-64 mode type load"ballbounce.bin",8,1
sys 49152

In C-128 mode type bload"ballbounce8.bin"
sys 4864

The C-128 version works on either the 40 or 80 column screen.

PROGRAM NAME:ballbounce.src

```
ad 1000 ;-----------------------------------------------
ap 1010 ; filename = ballbounce.src
pe 1020 ; ballbounce rawplayer is commercial software from
jm 1030 ; twin cities 128/64 issue #34 by:
ah 1040 ; parsec inc po box 111  salem ma 01970-0111 usa
ld 1050 ; copyright 1993 - all rights reserved
ea 1060 ;-----------------------------------------------
gn 1070 ;
fa 1080 ; this program is intended to show an example of
be 1090 ; some of the addressing and kernal routines for
pg 1100 ; the commodore 64 and the 6502 processor.
jf 1110 ;
kf 1120 * = $c000;indicates to the assembler to make a file
kd 1130 ;         with a starting starting address of 49152
ho 1140 ;         ($c000), on the c-128 use 4864 ($1300)
ln 1150 ;
ne 1160 ; since we are going to be using the kernal
lf 1170 ; routines by name in this program we will have to
mm 1180 ; tell the assembler where in memory the routines
ih 1190 ; are located
op 1200 ;
co 1210 plot = $fff0;
ad 1220 ;
of 1230 bsout = $ffd2;    display character .a at current
ad 1240 ;                 cursor position
dc 1250 getin = $ffe4;    return key pressed in .a
cl 1260 ;
kk 1270 jmp start;        skip ahead, over variable storage
ea 1280 ;
dn 1290 xpos .byte $00;   holds current .x, .y position
lo 1300 ypos .byte $00
gh 1310 xinc .byte $00;   x increment-value of 1=+1,0=-1
ia 1320 yinc .byte $00;   y increment-value of 1 = +1,0=-1
hc 1330 ;
mb 1340 start lda #147;   here we start the program by
mp 1350 ;                 clearing the screen
dd 1360 jsr     bsout;    by displaying chr$(147)
cl 1370 lda     #$01;     and then reset the position to 1,1
ce 1380 sta     xpos
dc 1390 sta     ypos
di 1400 sta     xinc;     since .a = 1 we're are going to
hn 1410 ;                 also use it to set the x
bb 1420 sta     yinc;     and y increment to both +1 - ie:
gd 1430 ;                 down & to the right
dh 1440 ;                 now we can draw the character on
jp 1450 ;                 the screen after setting the
ai 1460 ;                 cursor position.
hk 1470 doplot ldy xpos;  .y = xpos
ln 1480 ldx ypos;         .x = ypos
nn 1490 clc;              .c = 0 indicates we want to set
fn 1500 ;                 instead of read
ei 1510 jsr plot;         the cursor position
la 1520 lda #"o";         display a lower case  oh
fj 1530 jsr bsout
ga 1540 ;                 now we need to display so that it
mn 1550 ;                 will be visible. we do this by
op 1560 ;                 setting the .x and .y registers in
ln 1570 ;                 a nested loop and repeatdly
ie 1580 ;                 decrease them.
hh 1590 ;
nh 1600 ldx #$12;         .x =$12 -main speed control-found
bd 1610 ;                 by experimenting
jj 1620 ldy #$00;         .y = $00 -note that first dey will
```

```
jg 1630 ;                make .y = $ff
jg 1640 loop dey;        decrease .y by 1
hk 1650 bne loop;        and use the implicit comparsion
nc 1660 ;                with 0 to branch back
bh 1670 dex;             decrease .x by 1 after .y = 0
om 1680 bne loop;        if .x is also not equal to 0 then
kn 1690 ;                jump back, repeat.
of 1700 ;
ml 1710 ;                note that we don't need to reset
nm 1720 ;                .y to 0 after we decrease .x as
lh 1730 ;                we've already established it is 0
oa 1740 ;                because of the implicit comparison
em 1750 ;                in the 1st bne.
cb 1760 ;
jd 1770 doerase ldy xpos;move cursor back to erase
hh 1780 ldx ypos;        load registers .x, .y
ki 1790 clc
nh 1800 jsr plot
jd 1810 lda #" ";        and display a space to erase
hn 1820 jsr bsout
gi 1830 ;
cg 1840 ;                now we need to update the posit-
ij 1850 ;                ions by loading .a with xinc or
db 1860 ;                yinc to get an implicit comparsion
gm 1870 ;                during the lda (almost similair to
fe 1880 ;                an automatic cmp #$00 after each
og 1890 ;                lda). note that dex and inx also
ab 1900 ;                perform this implicit comparison.
ad 1910 ;                it's technically not a "true"
am 1920 ;                comparison but the beq and bne
ie 1930 ;                flags are valid.
ng 1940 ;
co 1950 update ldy xpos; get positions
im 1960 ldx ypos
he 1970 lda xinc;        if xincrement
og 1980 beq xminus;      is 0 then jump ahead
eb 1990 iny;             else increase .y
hc 2000 cpy #39;         check to see if equal to 39 - max
lh 2010 ;                col of the screen
ej 2020 bne updatey;     if it isn't then go update the y
ka 2030 ;                position
cd 2040 lda #$00;        else change xincrement
eg 2050 sta xinc;        for next time....
ak 2060 beq updatey;     use that implicit comparison to
eh 2070 ;                jump ahead
gd 2080 ;
ph 2090 xminus dey;      the dey also does an implicit
bl 2100 ;                comparison to $00
ki 2110 bne updatey;     so if it's non-zero then skip ahead,
li 2120 lda #$01;        else reset the xincrement for
km 2130 sta xinc;        next time. and we can fall through
gb 2140 ;                to updatey.
kj 2150 ;
gl 2160 updatey lda yinc;if yincrement is
ea 2170 beq yminus;      0 then jmp ahead
oo 2180 inx;             else increase .x
ij 2190 cpx #23;         comparse with maximum row #
ma 2200 bne setpos;      else store the new positions -
bb 2210 ;                finished updating them
oi 2220 lda #$00;        else change yincrement
mc 2230 sta yinc;        for next time...
cj 2240 beq setpos;      use the implicit comparison to
pl 2250 ;                jump ahead
ij 2260 yminus dex;      the dex also does an implicit
mf 2270 ;                comparison to $00
ec 2280 bne setpos;      so if it's non-zero then skip ahead
pl 2290 lda #$01;        else reset the increment for
ob 2300 sta yinc;        next time. and we can fall
bk 2310 ;                through to setpos
ja 2320 setpos sty xpos; set new x position
gb 2330 stx ypos; set new y position
gi 2340 ;
ki 2350 ;                now we can check to see if the
oo 2360 ;                "Q" key is pressed and if it is
ce 2370 ;                not we can go back and repeat all
dh 2380 ;                these instructions. if it is, we
gh 2390 ;                will clear the screen
kp 2400 ;                and rts - return back to basic.
ko 2410 ;
be 2420 getkey jsr getin;call kernal routine to get a
nb 2430 ;                character
kp 2440 cmp #"Q";        if it's an uppercase "Q"
nj 2450 beq exit;        jump ahead to exit.
ji 2460 cmp #"q";        else compare to a lowercase "Q"
ek 2470 beq exit;        and if it's that, then jump ahead
ll 2480 ;                to exit.
ka 2490 jmp doplot;      else failing both of those, go
nl 2500 ;                back and replot....
fg 2510 ;                here at exit we just clear the
eh 2520 ;                screen and rts.
cg 2530 ;
ap 2540 exit lda #147;   code for clearing the screen
mo 2550 jsr bsout;       display it.
fi 2560 rts;             and return to basic.
ec 2570 .end
```

E. REFERENCES / RESOURCES / TOOLS

1. Books

6502 Software Design by Leo J. Scanlon
Indianapolis, Indiana
Howard W.Sams, 1980.

6502 Assembly Language Subroutines
Lance Leventhal and Winthrop Saville
Berkeley, California
Osborne/McGraw Hill, 1982

Commodore Business Machines.
Commodore 128 Programmer's Reference Guide
New York: Bantam, 1986

Mapping The Commodore 128 by Ottis R. Cowper
Greensboro, NC
Compute! Publications, 1986.

Mapping The Commodore 64 & 64C
Sheldon Leemon
Greensboro, NC:
Compute! Publications, 1987

2. SOFTWARE - PUBLIC DOMAIN / SHAREWARE

Those of you on Genie can check the following files
(among the many others available)

file#	Program Name	Description
10329	SAL.SDA	C64 assembler
8794	LEARN-ML.SFX	machine language tutor
7109	CYCLE CHART	Chart of 6502 opcodes/ clock cycles
5336	MEGAMON.SDA	Excellent Machine Language Monitor
4818	PD PAL COMPAT. ASSEMBLER	Compile your PAL files with this!
3297	ML/HEX/BIN CHART	Prints a 5 page conversion aid
3081	EA.DOC	Documentation for EA
3080	EA	An intergrated Editor /Assembler
5417	C128 MEMORY MAP	memory map by Jim Butterfield

THE COLOR 64 BBS SYSTEM

Color 64 is a BBS program that has been around for 7 years, and in those years, more and more people have realized that Color 64 is one of the most unique BBS systems available for the Commodore 64. Color 64's greatest feature is that it is completely modifyable. Beginning Sysops can modify the system by running the easy-to-use Setup program. Advanced Sysops can modify the very programming of Color 64 itself, because changing Color 64 is as easy as editing a BASIC program.

You won't have to do everything for your BBS program, though, because Color 64 is packed full of features that make it easy to install and maintain. The latest version of Color 64, version 8.00, supports: standard modem interface up to 2400 bps; SwiftLink RS-232 interface for modem rates up to 38,400 bps (SwiftLink is a Creative Micro Designs product); use of any 100% Hayes compatible modem and most modems that support Hayes 'AT' commands; use of almost any Commodore compatible disk device (1541, 1581, 1571, CMD FD series, CMD HD, Lt. Kernal HD, ICT HD, IEEE drives, and more); use of the 1750 and 1764 REU's, expanded up to 2 megabytes; use of CMD's RamLink and RamDrive; use of some fastloader systems, such as the Skyles Flash! 1541 Interface. At least 2000 blocks or more of minimum disk space is recommended. Color 64 is a very disk intensive system, so it is HIGHLY recommended that your system have fast disk access, whether you use a Hard Drive, RAM expander, or a speed enhancer like JiffyDos (a CMD product).

Color 64 also supports: up to 26 upload and download directories with download descriptions; both Punter and Xmodem download protocols, with multi-punter transfers; a public message system, with up to 20+ categories, with full subject chaining; a fully featured message editor with built in MCI commands for message formatting; 40 and 80 column support; an automatic page-pauser function for long messages; Commodore graphics and color; ANSI graphics and color; ASCII text conversion; built in networking, with networked mail, public messages, and file transfers.

Color 64 has scores of public domain online-games and modules, all of which are accessible on our support BBS, and can be installed easily even by a novice Sysop. Included in the package are some games that have been specially designed to run with Color 64 v8. As you learn more about your BBS system, you can even write your own games and modules, which you can then share with all of the many other Color 64 Sysops. This system of free access to games and modules is one of the things that make Color 64 unique among other BBS systems, some of which ask money for their extra modules.

This leads to perhaps the best feature of all -- Color 64 is sold as a complete package. This means that you get the programs (meaning you get the source code too), built in networking capability, all of the necessary support files, the games and modules, installation and operating instructions, and a technical and programming reference guide. Other BBS systems may require an additional fee for networking, source code, games, or even programming information, but these are included free of charge in the Color 64 BBS system. Also, any future version 8 updates (8.01 to 8.99) will be provided free of charge to current version 8 owners, and can be obtained by downloading the updates/fixes from the Support BBS or via U.S. mail (fee charged for diskettes and postage).

More information about Color 64 BBS V8.00 can be obtained by contacting the Sysop of Sonic Temple BBS (Fred Ogle, AKA Betelgeuse, user #2) -- 1200/2400 bps, phone number (410) 285-0428, set terminal to 8-N-1. Or you can send GEnie mail to F.OGLE, or Inet/Usenet mail to F.Ogle@geis.com Pricing: For Color 64 version 8.00, a 4 disk set with documentation, the cost is $65.00; to upgrade from Color 64 v7.XX to v8.00, the cost is $25.00; There is a 20% discount with a trade in of another BBS program -- This offer is limited to certain BBS programs, so you must contact Fred Ogle to confirm a trade-in PRIOR to ordering. Send orders to: Fred Ogle, P.O. Box 35427, Dundalk, MD 21222-7427

Postage/Handling Charges: US Mail 3rd Class (ground), add $2.00; US Mail 1st Class (Air), add $3.00; Priority Mail (2nd Day), add $4.00; Express mail (Next Day), add $15.00. Check or money orders accepted -- $50.00 charge for returned checks. U.S. funds only. Canadian orders add $5.00. Prices and features listed are as of 8/3/1993, and are subject to change.

TWIN CITIES 128/64

SUBSCRIPTION INFORMATION

TWIN CITIES 128/64 -

Cost for six issues sent by 3rd Class Mail - USA ONLY!

$24.00 - TC128/64 magazine only
$40.00 - TC128/64 magazine & disk

TWIN CITIES 128/64 -

Cost for six issues sent by 1st Class Mail for the USA or Air Surface mail for Canada, Mexico, and other countries.

Overseas delivery now takes only 7-14 days to deliver the issue into YOUR country's mail system.

$31.50 - TC128/64 magazine only
$48.00 - TC128/64 magazine & disk

LABEL INFORMATION

Line #1 User ID, last issue, subscription type
 Subscription type #1 = magazine 3rd class w/o disk
 Subscription type #2 = magazine 3rd class w/ disk
 Subscription type #3 = magazine 1st class w/o disk
 Subscription type #4 = magazine 1st class w/ disk
Line #2 = name Line #3 = street address
Line #4 = City, State, Zipcode
Line #5 = optional, used to list countries

Payment can be by check or money order. ALL PAYMENTS MUST BE IN US FUNDS! Overseas customers can usually get a money order in US funds at a Post Office or Citibank location. Sorry we do not take credit cards

TC-128/64 COMPANION DISKS
Our companion disks not only save you time typing but the disks come with at least one "value" program written just for TC-128/64 that makes the disk worth the price!

If at any time you are not happy with the magazine or disk, you can be given a credit for the remaining issues to be used towards purchases made from our catalog, which contains items ranging from $1-$300.

Make checks payable to "Parsec, Inc."

Mail your payment to:

PARSEC, INC.
PO BOX 111
SALEM MA 01970-0111
USA

Only $24.95

FOR THE C64 AND C128 IN 64 MODE

Fun Graphics Machine

THE **FUN GRAPHICS MACHINE** (**FGM**) IS AN "ALL-IN-ONE" GRAPHICS PROGRAM FOR THE **C64** THAT TAKES OVER WHERE **PRINT SHOP** AND OTHER GRAPHICS PROGRAMS LEAVE OFF. WHAT YOU CAN DO WITH **FGM** IS LIMITED ONLY BY YOUR IMAGINATION, AND **FGM** CAN EVEN HELP YOU STIMULATE THAT. FOLLOWING ARE JUST A FEW OF THE THINGS THAT **FGM** CAN BE USED TO DO:

SUPPORTS ALL CMD DRIVES AND RAMLINK

BUSINESS CARDS
CUSTOM LABELS
NEWSLETTERS
VIDEO TITLING
CALENDARS
DIAGRAMS
POSTERS
FORMS

SIGNS
CHECKS
OVERLAYS
BROCHURES
LETTERHEAD
CERTIFICATES
GREETING CARDS
DISK ENVELOPES

DRIVES MOST PRINTERS TO THEIR LIMIT IN GRAPHICS

THE **FUN GRAPHICS MACHINE** SUPPORTS IMPORTING GRAPHICS AND HI-RES SCREENS FROM MANY OTHER POPULAR GRAPHICS PROGRAMS. ONCE IMPORTED, YOU HAVE FULL CONTROL TO MANIPULATE AND PRINT THE SCREENS AND GRAPHICS. A FEW OF THE PROGRAMS YOU CAN IMPORT FILES FROM ARE:

SUPPORTS MULTIPLE DRIVES

KOALA
RUNPAINT
PRINT SHOP
ADV OCP ART
VIDEO BYTE II

HANDYSCANNER 64
COMPUTER EYES
PRINTMASTER
NEWSROOM
DOODLE!

THIS AD CREATED WITH FGM

ANY **GEOS** SCREEN (INCLUDING **GEOPAINT** & **GEOWRITE**) MAY BE CAPTURED SIMPLY BY RESETTING THE COMPUTER AND LOADING **FGM**.

NOTE: YOU DO **NOT** NEED ANY ADDITIONAL PROGRAMS TO USE **FGM**.

THE **FGM PACKAGE** INCLUDES A 140 PAGE MANUAL, TUTORIAL, DOUBLE-SIDED PROGRAM DISK, AND DEMO TUTORIALS------**$24.95**

FUN GRAPHICS MACHINE FULL KEYBOARD OVERLAY -----**$3.50**
KEYBOARD OVERLAYS FIT OVER YOUR KEYBOARD AND CONTAIN ALL OF THE **FGM** COMMANDS. AN OVERLAY CAN GREATLY REDUCE THE **FGM** LEARNING CURVE. PLEASE SPECIFY WHICH COMPUTER THE OVERLAY IS FOR (**C64, C128, SX-64**) OR **C64** WILL BE SHIPPED.

FUN GRAPHICS MACHINE DEMO: THIS IS A DEMO VERSION OF **FGM** THAT YOU CAN GIVE TO FRIENDS OR PUT IN CLUB LIBRARIES --**$2.00**

***THE BELOW DISKS SUPPORT THE FULL-BLOWN VERSION OF FGM ***

FGM FONT DISK: OVER 90 FONTS IN **FGM** FORMAT ---------**$5.00**

FGM CLIPART: HIGH QUALITY GRAPHICS FOR USE IN **FGM**
FGM CLIPART VOL-1: FOUR SIDES WITH 200+ GRAPHICS---**$8.00**
FGM CLIPART VOL-2: TWO SIDES WITH 100+ GRAPHICS----**$5.00**

FGM CALENDAR TEMPLATE: HI-RES SCREENS READY FOR YOU TO USE TO MAKE DAILY, WEEKLY, AND MONTHLY CALENDARS ----**$5.00**

FGM OVERLAY TEMPLATE: MAKE YOUR OWN KEYBOARD OVERLAYS. SPECIFY KEYBOARD (**C64, C128, SX-64**) OR **C64** WILL BE SENT. NOTE THAT OVERLAYS ARE NOT A BEGINNERS PROJECT! -----**$5.00**

FGM UPDATE DISK V6.4: THIS DISK UPDATES EARLIER **V6.0-6.3** FGMS TO **V6.4**. INCLUDES SOME DEMOS AND UTILITIES-----**$2.00**

SHIPPING AND HANDLING (PER TOTAL ORDER)----------**$3.50**
FOREIGN ORDERS: FOR **AIR MAIL** ADD ADDITIONAL AS FOLLOWS:
CANADA/MEXICO $1.00, AUSTRALIA $10.00, ALL OTHERS $5.00
U.S. FUNDS ONLY **SORRY, NO CHARGE CARDS**

The FGM Connection, P.O.Box 2206, Roseburg, OR 97470

FOR MORE INFORMATION CALL 503-673-2234

RIO COMPUTERS

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

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

VIDEOFOX

The Tool For Creative Video Buffs

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

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

ONLY $59.95

VIDEO DIGITIZER

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

ONLY ~~$249.95~~ 149.95!

HANDYSCANNER 64

The Worlds First Handscanner for the 64!

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

ONLY $299.95

PAGEFOX

3 Easy To Use Editors For Perfect Home Desktop Publishing

GRAPHIC EDITOR ~ TEXT EDITOR ~ LAYOUT EDITOR

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

ONLY $139.95

ACTION REPLAY V 5.0

THE ULTIMATE UTILITY/BACKUP CARTRIDGE FOR THE C64/128

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

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

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

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

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

MIDI 64 -Only $49.99

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

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

DIGITAL SOUND SAMPLER

Only - $89.99

THE ADVANCED OCP ART STUDIO

COMPREHENSIVE, USER FRIENDLY ART AND DESIGN SOFTWARE

Only - $29.99

ADV. MUSIC SYSTEM

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

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

Only - $29.99

MAKE THE MOST OF YOUR ACTION REPLAY

GRAPHICS SUPPORT DISK

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

Only - $19.99

SUPERCRUNCHER - ONLY $9.99
Turn your Action Replay into a super powerful program compactor. Reduce programs by up to 50%! Further compact progrms already crunched by Action Replays compactor

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

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

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

ORIGINAL COMMODORE

1750 (512K) RAM EXPANSION PLUG-IN

~~$99.95~~ → 49.95!

Commodore has just released 400 of these hard-to-find RAM expansion units that give you a full 512K of RAM at a sensational price. NOTE: The 1750 works with C64 and C128. A separate heavy-duty power supply is needed for the C64, which we will sell at a discounted price of $27.95. DON'T MISS OUT ON THIS ONE!

The Grapevine Group, Inc.
3 Chestnut St., Suffern, NY 10901
Order Line: 1-800-292-7445 or 914-357-2424
Customer Service Line: 914-368-4242
Fax: 914-357-6243

We Ship Worldwide

Hours: 9 – 5 ET M – F

KeyDOS ROM V2
from Antigrav Toolkit

KeyDOS ROM is a chip that contains 20 new function key definitions and 20 great utilities. **KeyDOS ROM** is easy to install in the empty ROM socket in any C128 or C128D. **KeyDOS ROM** can also be installed in the space for an empty socket in Commodore 1700, 1764, and 1750 RAM Expansion Units. All the **KeyDOS** function keys and utilities are available ***INSTANTLY*** as soon as you switch on your C128! Why waste time hunting for disks or waiting for utilities to load? Now you can have all of the following features built in - instantly at your finger tips!

- ✔ Simple installation in all Commodore 128 or 128D computers and Commodore RAM Expansion Units. Internal installation usually can be done with only a screwdriver. REU installations require some minor soldering.
- ✔ Compatible with almost all C128 hardware and software including JiffyDOS, The Quick Brown Box, CS-DOS, and even other function ROMs.
- ✔ 20 new **KeyDOS Function Keys** simplify disk access on multiple drive systems. New function keys perform commands on any disk drive with only one key press and without typing file names! Load or run programs, scratch files, open and close 1581 subdirectories, load C64 programs in C128 FAST mode, batch files - (execute commands from a file), type SEQuential files without disturbing memory - all with easy to use KeyDOS Function Keys.
- ✔ **Swapper** memory management leaves BASIC memory undisturbed! Use any KeyDOS function at any time. Your BASIC program will be waiting for you when you get back to it! You can even use Swapper to keep more than one program in memory at the same time.
- ✔ New **KeyDOS Utility** lets you select multiple files from a directory list for typing, printing, copying, renaming, and scratching. Works with most popular printers. View ASCII files with a complete 80 column true ASCII character set. View CBMASCII files. View Screen Code files such as those produced by SpeedScript, The Write Stuff, and other popular word processors. Copy files between drives. Convert text files between ASCII and CBMASCII or from Screen Code to CBMASCII. Full 1581 subdirectory support.
- ✔ **KeyDOS 1581 Visual Partitioner** lets you create 1581 partitions and subdirectories without complex commands or calculations! The display shows where space is available for subdirectories and will create them for you automatically!
- ✔ **RAMDOS** for Commodore RAM Expansion Units up to 2 MB.
- ✔ Throw away your old boots! **GEOS SuperRBoot** is the most reliable method of rebooting GEOS from an REU that has ever been devised! Save yourself frustration and hours of lost work when GEOS crashes or hangs up! SuperRBoot will reboot GEOS even if your disk drives have been renumber or shut off, and even if all of the 128's memory is lost! **ACTIVATOR** simplifies switching between GEOS and C128 native mode by restoring the reset switch and STOP/RESTORE key combination to their normal functions.
- ✔ View, edit, or disassemble disk sectors. Dump monitor output to printer or disk file. **Hexpert** advanced machine language debugger. BASIC Find/Replace utility. Recover BASIC programs after a NEW command.
- ✔ Renumber and reset disk drives. New COLLECT command protects autoboot sectors.
- ✔ Switch between two independent 80 column screens. Screen text editor and screen dump.
- ✔ Screen clock with calendar and 24 hour alarm. (If you have a SmartWatch clock chip connected to joystick port #2 for use with GEOS, a special version of the **KeyDOS ROM** is available that sets the time and date automatically. Specify the part number of your SmartWatch when ordering).
- ✔ **Video Manager** sets VDC video options - colors, interlace, screen height, TABs, cursor shape, and more!
- ✔ Utility disk with sample batch files, demonstration programs, and support utilities.
- ✔ Satisfaction Money Back Guarantee and free lifetime replacement.

*"The program is a definite MUST for all 128 users. I definitely want **KeyDOS** to be a permanent part of my computer."*
-K.H., Perris, CA.
"If you don't have a 128, then this is (yet) one more reason to upgrade."
- Jan/Feb *Kernel Chronicles*, Edmonton, Alberta

KeyDOS ROM, with 52 page manual, utility disk, and shipping in U.S., A.P.O., F.P.O., Canada and Mexico is $32.50 US. Shipping overseas is an additional $3.

Antigrav Toolkit, P.O. Box 1074, Cambridge, MA 02142

CMD Sets Pace for 1993 with New Products and Lower Prices

CMD Offers One-Stop Shopping to Commodore Owners

Effective May 1, 1993 CMD acquired all rights to RUN software and has purchased all items from their inventory. As result, CMD will be offering one of the largest selections of Commodore 64/128 software and hardware available today. CMD's decision reinforces its commitment to C-64/128 owners worldwide. It is our hope that we can offer *"One Stop Shopping"* to Commodore owners. If we don't have it...we can probably find it! Here is a partial list of products. For a complete list call or write for a free catalog.

RUN Mag.- back issues
RUN/ReRUN software
Abacus books & software
GEOS 2.0 & applications
Timeworks Software
Superbase & Superscript
SOGWAP-Big Blue Reader

Microprose games
Xetec Printer Interfaces
Electronic Arts Games
Skyles Electric Works
Dr. T's Music Software
Power Supplies
Miscellaneous Cables

Blank disks - 3.5" & 5.25"
... and MUCH More

FREE*
TC128/64 - 500 Questions Answered Book with any purchase over $50.00
*One per customer add $1.00 s/h

CMD Utilities — A Powerful and Unique Collection of Disk Utilities for Commodore and CMD Storage Devices

FCOPY+ A two drive file copier featuring 1541,71,81, REU and CMD device compatibility which is capable of copying any size PRG, SEQ, REL, USR File. ***NEW*** features include:
Scratch/Unscratch Files • Copy/Delete C128 Boot Sector • Disk/Partition UnNew • File Compare • Access DOS Commands thru Menus • Lock/Unlock Files • Format Disk/Partition • Create/Remove Sub Directories • Rename File/Partition/Header • Change Current Partition/Subdirectory

MCOPY A two drive whole disk/partition file copier which supports CMD storage devices and Commodore 1541, 71, 81 disk drives.

BCOPY+ A powerful backup/restore utility which backs up any CMD device or partition to a 1541,71,81 drive or CMD FD Series floppy drive. This new version incorpoates the ability to dump an entire CMD device to an HD Series hard drive.

FIND Searches specified partitions on CMD devices for files that match a user defined filename pattern. Includes handy printer/screen toggle.

DIR SORT Alphabetizing utility for 1541,71,81 drives and all CMD storage devices makes it easy to organize large directories.

MCOMPARE Two drive disk compare utility makes it easy to evaluate the accurancy of a CMD partition against a disk copy.

FOLLOW LINKS Helps to locate and remove corrupt files.

CONVERT 41<>71 Converts between 1541 & 71 formats. Allows increased storage on 1541 disks after MCOPY'ing to 1571 or partition.

ZAP REU/DACC Clears memory in REU or RAMLink DACC partition.

HD POWER TOOLS Allows editing of CMD Device Partition Table.

REBUILD PDIR Helps to recover partitions after completely recreating the system on CMD Devices.

Shareware Utilities Included:

DEDIT64/128, SuperDEDIT64/128 Sort or manually edit directories subdirectories and partitions on 1541,71,81 drives and all CMD Devices.

NEW LOWER PRICES

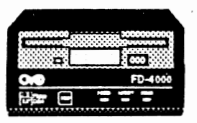
RAMLink	
RAMLink with 0 MB RAMCard	$199.95
RAMLink with 1 MB RAMCard	$229.95
RAMLink with 4 MB RAMCard	$329.95
Real-Time-Clock for RAMCard (Optional)	$20.00
RAMLink Battery Back-up (Optional)	$24.95
1 MB & 4 MB SIMM Modules	CALL
Parallel Cable (RAMLink to HD)	$14.95
RAMDrive	
RAMDrive 1 MB	$249.95
RAMDrive 2 MB	$299.95
HD-Series Hard Disk Drives	
HD-40, (Special Edition w/ 85 MB Drive)	$495.00
HD-100, (Special Edition w/ 170 MB Drive)	$595.00
HD-200, (Special Edition w/ 245 MB Drive)	$695.00
FD-Series 3.5", Floppy Disk Drives	
FD-2000 (800K and 1.6 MB Formats)	$179.95
FD-4000 (800K, 1.6 MB and 3.2 MB Formats)	$249.95
FD Real-Time-Clock Option	$20.00
Box of 10, High Density Disks (1.6MB)	$14.95
Box of 10, Enhanced Density Disks (3.2 MB)	$60.00
Single ED-Disk (3.2 MB)	$8.00
JiffyDOS (Specify Computer/drive model & serial number)	
C64-System (Computer & drive)	$49.95
SX-64 System (Computer & Internal 1541)	$49.95
C-128 System (Computer & drive)	$59.95
128-D System (Computer & Internal 1571)	$59.95
Additional Drive ROM's	$24.95
Software Protection	
geoMakeBoot (Makes bootable copies of GEOS 64/128)	$12.95
gateWay/64 or 128 (GEOS Desktop Replacement)	$29.95
geoCanvas (Alternative GEOS Paint Program)	$29.95
Collette Utilities (Handy GEOS Utilities)	$19.95
Perfect Print LQ for GEOS (Font Coll.1&2, 49 Fonts)	$49.95
Font Collection 3 (17 Fonts and 5 Borders)	$19.95
Border Font Collection 1 (24 Border Fonts)	$19.95
JiffyMON - ML Monitor for JiffyDOS/64	$19.95
CMD Utilities - Collection of Powerful Disk Utilities	$24.95
Miscellaneous CMD Products	
SwiftLink, RS-232 Interface (300 to 38,400 baud)	$39.95
SwiftLink Cable (DB9 to DB25)	$9.95
SID Symphony Stereo Cartridge w/ Player	$39.95

High Performance SCSI Hard Drive for the C64 & C128

HD Series Hard Drives are available in capacities up to 200 MB, are fully partitionable, and can emulate 1541, 1571, & 1581 disks while Native partitions utilize MSDOS-style subdirectories. HD's connect easily to the serial bus or parallel via RAMLink. Includes built-in JiffyDOS, SWAP feature and Real-Time-Clock. HD Series Drives offer superior compatibility with most commercial software including BBS, Productivity and GEOS. And with new pricing, HD Series drives offer the lowest cost/MB of any C64/128 storage device.

Power Backed Expandable RAM Disk and Interface

The fastest possible form of storage, RAMLink provides instant access to files and retains data while your computer is turned off. Easy to use and expandable up to 16 MB, RAMLink utilizes the same powerful operating system found in the HD. RAMLink also offers built-in JiffyDOS, SWAP feature, reset button, enable/disable switch, pass-thru port and RAM port for REU/GEORAM use. Ideal for those requiring maximum speed, expandability and compatibility with all types of software and hardware including GEOS.



High Capacity 1.6 and 3.2 MB 3.5" Floppy Disk Drives

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

JiffyDOS

Speeds up disk access by as much as 1500% while maintaining 100% compatibility with commercial software

- Speeds up Loading, Saving, Verifying, Formatting and Reading/Writing of PRG, SEQ, USR and REL files
- Built-in DOS Wedge plus 17 additonal features including file copier, text dump, printer toggle, and redefinable function keys

ROM upgrade installs easily into most computers and disk drives. Supports C-64, 64C, SX-64, C-128, 128-D, 1541, 1541C, 1541-II,1571, 1581 and more. 128 system supports both 64 and 128 modes and upgraded Kernal routines.

Ordering Info

UPS $15.00; 2nd-day $25.00; AK, HI, Canada $35.00
UPS $9.00;2nd-day $16.00,;AK,HI,Canada $20.00
UPS $5.50,;2 day $10.00,;AK,HI,Canada $15.00
1st Class Mail $2.00; Canada $5.00

MC, Visa, Money Order or Bank Check.COD shipments add $5.00
COD's may require advanced deposit. Personal checks are held for 3 wks.

Creative Micro Designs, Inc
P.O. Box E

CMD
Orders (800) 638-3263
Info (413) 525-0023
BBS (413) 525-0148
9 AM to 5 PM, EST, Monday thru Friday.

TWIN CITIES 128/64 CHECKSUM PROGRAMS By Michael Gilsdorf

If you decide to type in programs from Twin Cities 128/64 magazine, you should first type in and run TC128 Checksum (for the C128) or TC64 Checksum (for the C64). These programs check your typing by generating a two-letter checksum each time you enter a program line and press the RETURN key. The checksum is displayed in the upper left hand corner (home position) of the 40 or 80 column screen. To check for typing errors, compare the checksum on the screen with the one appearing in the magazine listing. If they're different, then you know you've made a typing error. The magazine listing will show the correct two letter checksum in front of each line number.

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

Also, both TC128 Checksum and TC64 Checksum are designed to make it easier for you to indent text or enter blank lines. To indent text, simply type the line number, space or tab over to where you wish the text to begin, and then begin typing. This feature will improve the readability of your listings by making portions of your program such as FOR-NEXT loops and DO loops stand out more easily. To enter a blank line, type a line number followed by at least two spaces (or tab) and a shifted character. When the program is listed only the line number will appear.

```
10 rem read data
20
30 for j=1 to 80
40     read a$
50     b$=b$+a$
60     if a$="." then j=80
70 next j
```

Additionally, TC128 Checksum has the ability to generate a checksum listing. (This feature is not provided with TC64 Checksum.) The listing will show the checksum along side each line number as the program is listed. To begin the listing, type a # in direct mode (without a line number) as the first character on a line. Do not include any additional BASIC commands on the line: otherwise they will be ignored. Once the listing begins, you can use the NO SCROLL key or STOP key to pause or stop the listing as desired. You'll find the checksum listing especially useful if you need to redisplay the checksums and double check the lines you've already entered.

```
km 10 rem read data
be 20
ob 30 for j=1 to 80
jn 40     read a$
hb 50     b$=b$+a$
jh 60     if a$="." then j=80
fm 70 next j
```

Also, should you decide to submit a program listing to Twin Cities 128/64 magazine for publication, you can use TC128 Checksum program to save a checksum listing to disk. To create an a SEQ file listing, type:

```
open 2,8,2,"0:filename,s,w": cmd 2
#
print# 2: close 2
```

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

```
open 2,4: cmd 2
#
print# 2: close 2
```

Both TC128 Checksum and TC64 Checksum programs are listed below. Be sure to save a copy to disk before running them. Once run they will automatically activate themselves.

PROGRAM NAME:TC64 CK SUMV1.0

```
1 print chr$(147);"tc64 checksum  v1.0"
2 print "by mike gilsdorf (c) jul 93 by parsec inc": print
3 for a=828 to 1001: read d: pokea,d : t=t+d: next
4 if t<>22612 then print "data error": end
5 poke 770,60: poke 771,3
6 print "tc64 checksum activated"
7 print
8 print "to deactivate, type:"
9 print "poke 770,131: poke 771,164"
10 :
100 data 162, 255, 134,  58,  32,  96, 165, 134
105 data 122, 132, 123,  32, 115,   0, 170, 144
110 data   3,  76, 142, 164,  32, 107, 169, 160
115 data   0, 169,  32, 198, 122, 209, 122, 208
120 data   8, 198, 122, 209, 122, 240, 250, 230
125 data 122, 230, 122,  32, 121, 165, 132,  11
130 data 160,   0, 132, 122, 230, 123,  32, 145
135 data   3,  56,  32, 240, 255, 152,  72, 160
140 data   0, 185, 227,   3,  32, 210, 255, 200
145 data 201, 146, 208, 245, 104, 168,  24,  32
150 data 240, 255,  76, 164, 164, 162,   0, 134
155 data 251, 134, 254,  24, 165,  20, 101,  21
160 data 133, 253, 177, 122, 240,  33, 170, 224
165 data  34, 208,   2, 230, 251, 165, 251,  74
170 data 176,   4, 224,  32, 240,  14, 166, 254
175 data 177, 122,  24, 101, 253, 133, 253, 202
180 data  16, 246, 230, 254, 200, 208, 219, 152
185 data 208,   5, 169,  45, 168, 208,  17, 165
190 data 253,  74,  74,  74,  74,  24, 105,  65
195 data 168, 165, 253,  41,  15,  24, 105,  65
200 data 140, 230,   3, 141, 231,   3,  96,  19
205 data  18,  32,  35,  35,  32, 146
```

PROGRAM NAME:TC128 CK SUMV1.0

```
1 print chr$(147);"tc128 checksum  v1.0"
2 print "by mike gilsdorf (c) oct 91 by parsec inc": print
3 bank 15: for a=3328 to 3583: read d: poke a,d: t=t+d: next
4 if t<>29208 then print "data error": end
5 poke 770,0: poke 771,13
6 print "tc128 checksum activated"
7 print "to list, type:  #": print
8 print "to deactivate, type:"
9 print "poke 770,198: poke 771,77"
10 :
100 data 162, 255, 134,  60,  32, 147,  79, 134
105 data  61, 132,  62,  32, 128,   3, 170, 240
110 data  14, 144,  15, 201,  35, 208,   7, 166
115 data  45, 165,  46,  76, 223,  13,  56,  76
120 data 212,  77,  32, 160,  80, 169,  32, 198
125 data  61, 209,  61, 208,   8, 198,  61, 209
130 data  61, 240, 250, 230,  61, 230,  61,  32
135 data  10,  67, 132,  13, 160,   0,  32,  89
140 data  13,  56,  32, 240, 255,  32, 129, 146
145 data  19,  18,  32,  78,  75,  32, 146,  27
150 data  81,   0,  24,  32, 240, 255,  76, 234
155 data  77, 162,   0, 134, 251, 134, 254,  24
160 data 165,  22, 101,  23, 133, 253, 177,  61
165 data 240,  33, 170, 224,  34, 208,   2, 230
170 data 251, 165, 251,  74, 176,   4, 224,  32
175 data 240,  14, 166, 254, 177,  61,  24, 101
180 data 253, 133, 253, 202,  16, 246, 230, 254
185 data 200, 208, 219, 152, 208,   5, 169,  45
190 data 168, 208,  17, 165, 253,  74,  74,  74
195 data  74,  24, 105,  65, 168, 165, 253,  41
200 data  15,  24, 105,  65, 140,  75,  13, 140
205 data 205,  13, 141,  76,  13, 141, 206,  13
210 data  96, 200,  32, 236,  66, 153,  20,   0
215 data 192,   3, 208, 245, 200, 169,  63, 141
220 data   0, 255,  32,  89,  13, 169,   0, 141
225 data   0, 255,  32, 129, 146,  78,  75,  32
230 data   0, 166,  22, 165,  23,  32,  35,  81
235 data  32, 181,  75, 166,  65, 165,  66, 134
240 data  97, 134,  61, 133,  98, 133,  62,  32
245 data 152,  85, 160,   0,  32, 236,  66, 133
250 data  65, 200,  32, 236,  66, 133,  66, 208
255 data 184, 197,  65, 208, 180,  76,  55,  77
```