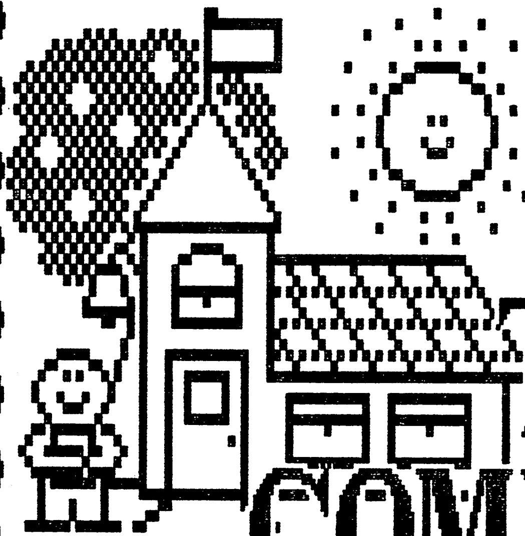
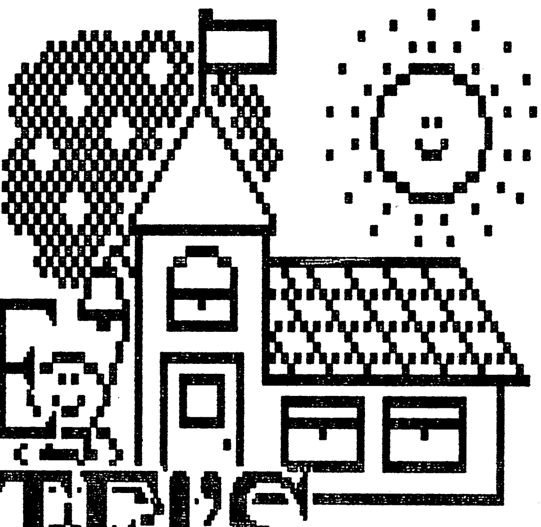
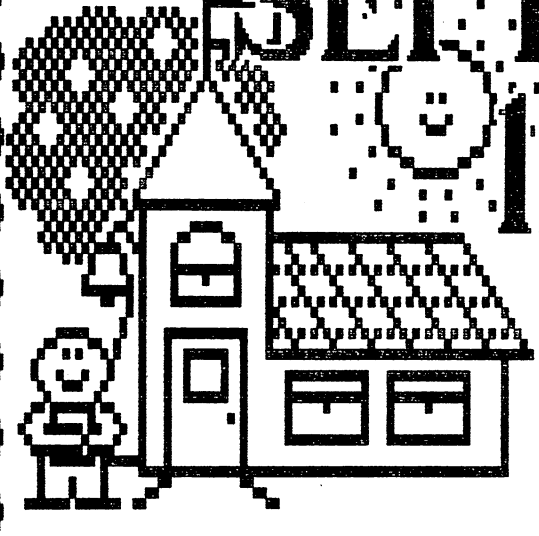
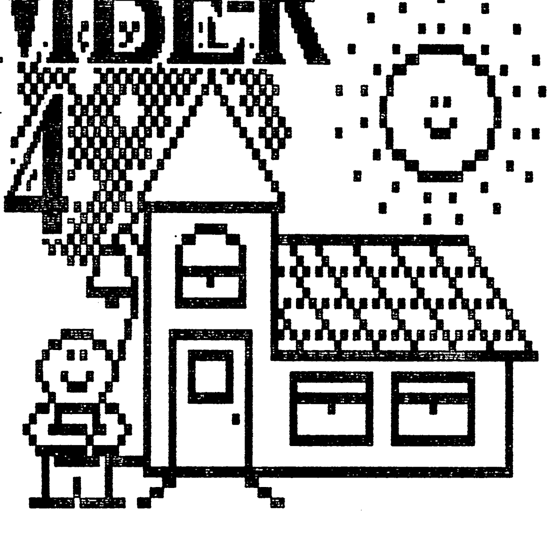



THE COMPUTE!'S GAZETTE DISK

SEPTEMBER 1994

```
*****  *****  *   *  *****  *   *  *****  *****    *  *****
*      *   *  ** **  *   *  *   *    *    *       *   *
*      *   *  * * *  *****  *   *    *    ***    *    *****
*      *   *  * * *  *      *   *    *    *               *
*****  *****  * * *  *      *****    *    *****       *****
```

G A Z E T T E   O N   D I S K   S E P T E M B E R   1 9 9 4

Disk Magazine

C-64 Users, Type: Load "Menu",8,1 and press <Return>.

C-128 Users ----: Enter 128 Mode; then type RUN "128 Menu" and <Return>

****

GAZETTE DISK *** September 1994

Features:

EXPLORING THE 6502
By Frank Gordon

Examine the orderly world of opcodes with this article and companion program.

Reviews:

4-SKAN, a scanner that works from your printer, reviewed by Don Radler.

Columns:

64/128 VIEW by Tom Netsel.
Piracy in the Computer Age.

FEEDBACK.
Comments, questions, and answers.

D'IVERSIONS by Fred D'Ignazio.
Paper Training Sparky the Dog.

MACHINE LANGUAGE by Jim Butterfield.
Counting Up/Down.

BEGINNER BASIC by Larry Cotton.
More Variables and Constants.

PROGRAMMER'S PAGE by David Pankhurst.
Screen Savers.

GEOS by Steve Vander Ark.
GEOS Isn't Perfect.

PD PICKS by Steve Vander Ark.
Titles, Slowpoke, Spooky Eyes.

128 Programs:

Going to the Dogs by J.J. Hromdik
A game that simulates a day at the dog races, including parimutuel wagering.

64 Programs:

Directory Reader by Justin Mahoney
Read disk directories without disturbing programs already in memory.

Component Selector by Robert Lindsey
Electronic hobbyists can use this utility to select components for power supplies.

Ketchem by William Snow
Ketchem is a fast-paced math game for th 64 and one or two players.

Bascan by Daniel Lightner
Search BASIC programs for strings with this scanning utility for the 64.

Going to the Dogs by J.J. Hromdik
A version for the 64 of the 128 dog racing game.

Titles (PD).
Make title pages for your video tapes with this program.

Slowpoke (PD)
Slow down your BASIC programs with this utility.

Spooky Eyes (PD)
Have fun with this amusing little demonstration.

Gazette, September 1994

DIRECTORY READER

By Justin Mahoney

Read disk directories without disturbing programs already in your 64's memory.

Have you ever wished for a short, memory-resident machine language routine for your 64 that would allow you to display the current directory to the screen without overwriting the BASIC program in memory? Or maybe you've wanted a routine that would do the same thing from a machine language program. Imagine: What if this routine would also allow the user to stop or pause the listing at any time?

Directory Reader does all of the above. It's a short BASIC utility that will construct an even shorter machine language routine at any specified address. This routine will perform the wished-for functions: It lists the disk directory and has both Pause and Stop features built in. Even better, when the ML routine is constructed, it's only 264 bytes long, taking up only two blocks of disk space. It can easily be loaded into memory during the execution of any program.

RUNNING DIRECTORY READER
The Directory Reader BASIC component is on the flip side of this disk. To use it, you should copy it to a work disk because it writes a machine language routine to disk at whichever address you specify.

When you run the program, it will ask for an address to begin the routine code. This can be any address in memory. The routine is assembled directly to disk, so addresses such as 2048 that would usually overwrite BASIC are still valid.

Next, the program will prompt you to insert a disk on which to save the code. This should be a disk with at least two blocks free. After pressing Return, the program will verify that the disk is seated properly in the drive.

The last prompt is for a filename. You can use any legal Commodore filename, up to the full 16 characters. After you press Return, the program will tell you that it is saving. When it is finished, the word "DONE!" will be printed onscreen, and the code is ready to be used.

USING THE ROUTINE
There are three ways to load the Directory Reader code: in immediate mode, from inside a BASIC program, and from inside a machine language program. In each case the routine will load at the address you specified in the creator program. After it is loaded, it can be accessed by opening a channel for the directory of the desired drive and accessing the routine. (It is not necessary to open #15 unless errors are anticipated.)

From the BASIC prompt, here's the syntax to use.

```
OPEN 1,8,0,"$":SYS xxxxx
(where xxxxx is the starting address)
```

From inside a BASIC program, add a line number.

```
50 OPEN 1,8,0,"$":SYS xxxxx
```

From inside a machine language program, use this code.

```
LOAD    LDA  #1    ;file number
        LDX  #8    ;device number
        LDY  #0    ;secondary address
        JSR  SETLFS
        LDA  #1    ;length of filename
        LDX  #<DOLLAR
        LDY  #>DOLLAR
        JSR  SETNAM
        JSR  OPEN ;open channel
        JSR  xxxxx
```

After each of the above three methods, the routine will begin displaying the directory, and the S and P keys become active to stop and pause the listing, respectively.

Note that the above methods exclude error-checking; this can be added. When the routine is finished, it automatically closes file number 1 and clears all open channels by calling the CLRCHN routine. This will not affect the operation of any BASIC programs, but machine language programmers should be aware of this. Remember that file number 1 must be used for this routine.

Justin Mahoney lives in Chino Hill, California.

Gazette, September 1994

GOING TO THE DOGS

By J.J. Hromadik

A board game for the 64 or 128.

Going to the Dogs is a board game for the 64. It can be played on the 128, but the animation isn't as crisp. The game simulates a day at the dog races, including pari-mutuel betting. There are ten races per game, and up to three players may participate. The game is easy to play; just follow the prompts.

There are 110 dogs in the kennels, with 7, 8, or 9 dogs entered in each race. The computer selects the racing card and the odds for each dog. No dog races twice in a game.

The game consists of a tote board that lists the entries with their odds for each race. Races are run at various lengths. The board lists any player messages and race results. Information about bets and payoffs are listed below the tote board.

Each player starts with $1,000 and may bet on three entries per race, with a maximum bet of $80 per entry. The entry is bet to win, place, or show. Each player in turn places bets when prompted by the computer.

To bet, move the cursor to highlight an entry and then move the cursor to highlight first, second, or third. Then move the cursor to highlight the amount of the wager. The highlight is moved by pressing any key. when Player's Choice is highlighted, press Return. This procedure is repeated for each player.

If you desire, the computer will do the betting on one, two, or three entries. Simply press the C key when prompted or after any complete wager on an entry.

Here are some rules to keep in mind.

* The odds affect the race and the payoff.
* The computer will accept a bet even if a player's pot is minus.
* Names of dogs may be changed to suit the player. Names are limited to 14 characters.
* Any momentary delay in posting the results is due to the computer's balancing the odds.
* The real winner is the one who has the most fun.

J.J. Hromadik lives in Ventura, California.

Gazette, September 1994

COMPONENT SELECTOR

By Robert Lindsey

Electronic hobbyists can use this utility to select component values for electronic power supplies.

For those of you who like to build your own power supplies for electronic projects, the program will select the rectifier and capacitor values for various transformer ratings. It also computes the DC output of the supply. It calculates values for full wave bridge, full wave center-tapped, and full wave bridge center-tapped power supplies.

When the program is run, it prompts for output to the screen or to a printer. The screen display gives a single supply listing, while the printer option prints a list of 50 outputs, incrementing up from the transformer voltage that the user inputs.

The program calculations are conservatively rated, and you should always round off values to the next higher standard voltage ratings when you get around to selecting the actual components.

ABOUT THE PROGRAM

I made use of subroutines for the repetitive functions. The subroutine at line 5000 generates the menu used for inputting values. Routines 6000 and 7000 generate the screen component labels. Routine 8000 creates the printer output and 9000 does the component calculations.

If you want the printer output to be shorter, in lines 1540, 2550, and 3550 change 50 to the total number of calculations you wish printed.

The program calculations are based on linear-type power supplies. Since the hobbyist may have several transformers around the shop, the screen output of the program treats the transformer as the variable unit. This lets you test the values for whatever transformer you have on hand to determine whether or not it will be satisfactory.

Warning: This article is not meant to teach power supply design and construction. If you plan to use this utility with actual projects, you should already be experienced in proper electronic safety, design, and construction techniques. Those subjects are beyond the scope of this article.

Robert Lindsey lives in Mechanicsburg, Pennsylvania.

Gazette, September 1994

KETCHEM

By William F. Snow

Ketchem is a fast-paced math game for the 64 and one or two players.

We all know that learning math facts is a necessary evil when trying to become proficient in higher math skills. Ketchem is a fun way to practice some fundamental addition and subtraction facts.

Ketchem is written in BASIC. To play it, simply load and run. Brief instructions are given on the first screen. You are then asked whether you want to play against the computer or another human and told to enter your name. You must then hit Return to begin the game.

Like some other popular board games, the players move along a path consisting of a series of spaces. Each player trys to arrive at the finish before his or her opponent. Unlike most board games, however, the number of spaces moved in Ketchem is not determined strictly by chance.

During each turn, you are presented with two numbers. You must quickly decide whether to add or subtract those numbers. The result of the operation will determine how many spaces your token will advance. You have only five seconds to make your decision by pressing the A key for add or the S key for subtract. If you do not make your decision in time, the computer will pick a random number determining the distance to advance.

It may seem that addition would always be the best choice, but this is not necessarily true. Along the path you'll discover holes and arrows. If you land on a hole, you drop through to the next level of the path, but if you land on an arrow, you are sent back one level. If you can manage to land on your opponent, the opponent is sent back to the starting position.

The children in my classroom really enjoy playing Ketchem. It is short, easy to play, and appealing because of strange sound effects. Everyone will have fun with Ketchem and also will improve his or her addition and subtraction fact skills.

William Snow is a teacher who lives in McHenry, Illinois.

Gazette, September 1994

BASCAN

By Daniel Lightner

Search BASIC programs for strings with this scanning utility for the 64.

Have you ever tried to find a specific command or SYS call in a BASIC program, but it eludes you? Wouldn't it be great to have something like the Search command found in most word processors to help to find a certain string buried somewhere in a BASIC program? That's what Bascan does.

With Bascan you can search through BASIC programs in memory for any given string of characters, but there's more. You can also use it to view a BASIC program on disk without interfering with the BASIC program that is currently in memory. How about another feature?

Bascan also allows you to send commands to the disk command channel or view the disk directory without disturbing the program in memory. With Bascan in memory BASIC programmers have a powerful aid to their skills and a handy way of looking at other programs without loading them.

Bascan is written in machine language. Its starting and stopping addresses are C000 and C87E, respectively.

USING THE PROGRAM

To use Bascan, type LOAD"BASCAN",8,1. When it has finished loading, type NEW and hit the Return key. Now you are ready to load a BASIC program.

To activate Bascan, type SYS49152. Immediately the Bascan menu appears displaying four options. The first is activated by pressing f1. This option allows you to search for a string in the BASIC program in memory.

Enter the word "PRINT" and Bascan will display all the print statements in the program. When Bascan finds what it is searching for, it will list that line to the screen and continue searching until it either finds another match or comes to the end of the program.

While Bascan is searching, it can be temporarily stopped by pressing any key; it can be restarted by pressing another key. You can abort a search while it's scanning by hitting the Stop key. You can abort when it is asking for input by typing an asterisk (*) and hitting Return.

After Bascan has finished searching, it waits for you to press a key to return to the menu.

Option 2 is activated by pressing f3. Here's a utility that BASIC programmers will appreciate. Bascan allows a person to look at any other BASIC program without disturbing the one that is in memory.

To test this option, place a disk containing another BASIC program in drive 8. When you're ready, hit f3. Bascan will prompt for a filename. Type the name of a program that is in drive 8 and hit Return. When Bascan finds the program, it will start listing it to the screen.

To stop the program listing, press any key; to restart it, press any key again. To exit to the menu, press the Run/Stop key. After it has finished reading the file, Bascan will wait for a keypress.

Bascan saves you the trouble of having to save the program that you're working on, load another just to look at a certain part of it, and then reload the original program. If you have to repeat this process several times, it can become time consuming. It can really get confusing if you're not scratching the old files. You may have a real mess on your disk.

Option 3, or f5, is for disk commands. Here you are able to send disk commands to the command channel. Disk commands should follow the same syntax as they would when opening the command channel to drive 8 from BASIC. Here are some of those commands.

COPY
To copy a file to the same disk, use C0: or C: followed by the new name, an equal sign, and the old name as in the example below.

```
C0:newname=oldname
```

Press Return to issue the command.

RENAME
The RENAME command is similar.

```
R0:newname=oldname
```

SCRATCH
The SCRATCH command is S0: followed by the filename to be scratched. Wildcards are allowed.

```
S0:filename
```

INITIALIZE
INITIALIZE is I0: followed by Return.

VALIDATE
To validate, type V0: and then press Return.

FORMAT
The FORMAT, or NEW, command is N0: followed by name, comma, and two letters representing a unique ID.

```
N0:disk name,ID
```

The FORMAT command will destroy all data on a disk, so be sure that you put the proper disk in the drive before pressing Return.

You can abort option 3 by entering a single asterisk.

Entering a dollar sign ($) will allow you to view the disk directory. Hold down any key to stop the listing. To restart, press any key. Pressing the Run/Stop key will abort the directory.

The last option sends you back to BASIC. Press f7.

Daniel Lightner lives in Sidney, Montana.

Gazette, September 1994

TITLES

Titles is a utility for use with a VCR. that lets you produce titles for your videotapes. Run the video cable from your computer to the Video In jack on your VCR and then record the title that you produce with this program. The program TITLES INST loads onscreen instructions before loading the actual program.

SLOWPOKE

Run your BASIC programs slower than normal with this little utility. Enter POKE 251,X to slow things down. Enter a number between 0 and 255 for X. A value of 50 is a good starting place.

SPOOKY EYES

This one is simply a fun little program that keeps an eye on you.

All of these public domain programs are discussed in detail in Steve Vander Ark's "PD Picks"column, found elsewhere on this disk.

Gazette, September 1994

64/128 VIEW: Piracy in the Computer Age

By Tom Netsel

Bill Clark in Lynden, Washington, sent in a letter recently that asks for guidelines for handling a user group's disk library. Over the years, users have abandoned their 64s or 128s and have donated their software to the club.

Disks and magazines from long-defunct publications are also available, but they present a problem for Clark's user group other than storage. Finding a place to keep everything is the simple part, deciding how to use the material is a tougher question.

"We are enjoined to neither sell nor give away copies of commercial disks," Clark writes. "That is easy. The following indicates what is not easy."

He then explains that should the group loan a disk, the disk might not be returned. That's stealing and the group loses.

If the group asks for collateral when it lends a disk, how much should it require? Should it be cash and if so, how much? What if the disk still isn't returned? Should the collateral cover the price of the disk? Should the group lend copies only and keep the originals in a safe place?

How can a group make sure that the borrower doesn't make a copy of the software. The closet copier cheats us all.

Clark presents a number of good questions that I'm sure many groups have faced at one time or another. So how does your user group manage its library? How do you handle missing software, users who don't return software, and the question of illegal copying?

If you'll send me your group's guidelines, I'll publish them and perhaps they'll help other groups who are experiencing similar problems.

Just to show that Clark takes software piracy seriously, he sent along a couple of short plays in three acts. The moral of these plays is not aimed at anyone in particular, but it should make many of us pause and think.

The first play is called Commercial.

Act 1
Rudy is a programmer for a major software vending company. he has a great idea for a new program, so writes it on his own time, and sells it to the company for whom he works.

The contract says Rudy will get $1 for every copy sold. The company develops the packaging, marketing, advertising, and distributing.

Act 2
Over 2 million 128 users see the advertising. Five hundred users buy copies from the company, so Rudy gets $500.

Enough buyers share their copies with other 128 users so that there are now 40,000 128 users enjoying the fruits of Rudy's labors.

Rudy should have received $40,000; he is short $35,500 of his due for his expertise and year-long labor in producing a valuable program. Rudy knows this.

Act 3
Rudy says, "Forget it!"

He takes up tennis in his spare time, hoping someday to meet Bill Gates and find a better opportunity for his programming skills. Numerous 128 users who received something for nothing at Rudy's expense will get no more. The well has dried up, and so has the 128.

Update
Rudy met Bill and got a job, but see the next play.

Here's a little number called Shareware, another play in three acts.

Act 1
Roddy, Rudy's brother, has the same good genes for programming. He works for an electrical supply firm. His 1985 128 was an immediate challenge, and he thrived on programming.

Using his 128, Roddy came up with an idea for a program that he thought would be welcomed by other users. He develops the program after many hour's of work and had a few friends Beta test what he had written.

Act 2
With all tests completed, Roddy puts his program on a local BBS as shareware. He asks users who like an duse his program to send him $5.

Roddy was on a major BBS on evening and saw his program there. He thought, "Great, maybe this will have all been worth it."

A conservative estimate is that 12,000 users had downloaded and were enjoying Roddy's program. His shareware receipts? $45.

You do the math this time.

Act 3
Roddy decided to quit wasting his time programming for his fellow users. He turned his talents to writing a program that was a big help to the company for which he worked. His program was so good, he got a

$500 bonus.

Beats the heck out of shareware. We all know what happened to the 128 and why.

Update
Same as Rudy.

Moral: You can't get something for nothing unless someone, somewhere, sometime gets nothing for something.

Gazette Disk, September 1994

FEEDBACK

BUG-SWATTER
I spent a lot of time typing in Ultimate ML Mon from the July 1993 Gazette, and I was distressed when I didn't perform as expected. The problem occurred when examining machine language programs that load like BASIC programs because the boot was still in memory.. I was getting errors in line 10 even though the program had no line 10. The boot program always listed after errors, but lines 10 and 20 had been partially overwritten.

I have created a new boot program for Ultimate ML Mon that performs a NEW before making the SYS call to start the program.

```
10 REM COPYRIGHT 1993 - COMPUTE PUBLICATIONS - ALL RIGHTS RESERVED -
JULY GAZETTE
20 D=PEEK(186):IF A=0 THEN A=1;LOAD "ULTIMON.L",D,1
25 POKE 646,PEEK(53281)
30 PRINT"[CLR][5 DOWN]NEW"
40 PRINT"[2 DOWN]SYS52234[6 UP]
50 POKE631,13:POKE632,13:POKE198,2
```

I hope this might help some other people who have experienced problems with this program.
BRUCE THOMAS
EDMONTON, AL
CANADA

MISSING DISKS
I am sorry to learn that the October 1992 and November 1993 issues of Gazette Disk are no longer available. I am concerned about the nonavailability of certain programs on disk that I wanted to obtain. These had to do with SpeedScript updates and now I am forced to type them in. I am particularly interested in SpeedSpeller 128 and SpeedSpell. Can you offer any ideas on finding them?

Also, do you have any plans of offering an update to the SpeedScript disk with the programs that have appeared since the last disk was compiled?
LOUIS MCNICOLL
SALEM, OR

I am not sure who you contacted, but those disks are still available from our Greensboro office. We may be temporarily out of a disk, but we do keep a master from which to make more copies. The disks you mentioned are $11.95 each This includes shipping and handling. You can order them by writing Gazette Single Disk Sales, 324 West Wendover Avenue, Suite 200, Greensboro, North Carolina 27408.

SpeedScript has been our most popular disk product. Since the last SpeedScript collection was released, we have published a number of improvements or updates. If we feel that there is sufficient interest in such a disk, we will be happy to offer it to our readers.

MORE 128 PROGRAMS
I am a new subscriber to Gazette Disk. My major complaint is the dearth of 128 information. I rarely use 64 mode unless forced to. The 128 is a much richer machine.

Another observation is that you seem to believe your subscribers are eternally stuck at the novice level. Perhaps this is a gross error on your part and will cost you subscribers.
JOHN LOGUE
ADAMSVILLE, PA

Several readers have expressed an interest in getting more 128 programs, but we can only publish the programs that are submitted to Gazette. As it stands now, we get about one 128 program for every seven or eight 64 programs. Several contributors submit 128 programs and we most always purchase their submissions. So if you program in 128 mode, we'll be happy to consider purchasing your best efforts. Since there's little competition, your odds of making a sale are much greater.

Not responding to subscribers' needs is the biggest mistake any publication can make. We've recently published several articles on machine language that may not appeal to many novice programmers. But if we find that the majority of our readers want such programs, Gazette would be amiss if it didn't publish them.

That's why the recent readership survey is so important. It gives us a good idea as to who our readers are. To those of you who have taken the time to send in your surveys by mail, fax, and modem, you be sure that we will pay close attention to what you tell us. Thank you for your replies. We'll have a report in the "64/128 View" next month.

WHERE'S THE GEM?
In the March 1994 column "64/128 View" you mentioned a spreadsheet program called GemCalc. You said it is (was?) available on the disk Gazette Productivity Manager. I can find no mention of this disk. I think you should ad an advertisement of your own, listing the prices of disks and magazines that are available.
JAMES HOOD
SALT LAKE CITY, UT

A couple of years ago, the manager of the Gazette Special Disks changed the name of the Productivity Manager to Gazette Power Pak. This disk sells for $16.95, which includes handling and shipping.

On it you'll find 64 and 128 versions of GemCalc, a full-featured spreadsheet plus 13 sample templates. It also contains Memo card, a compact database; Financial Planner to help you decide whether to rent or buy, how much life insurance you need, how much you should save, and other financial information.

As for other special Gazette Disks, be sure to see the Advertisement menu on this issue. We have included several products that are still available.

MORE FILES ON DISK
In the February "Feedback," Jeff Peterson asked how he could put more than 144 files on a disk. You might want to pass along the following information about a product called 1541/1571 ROMDISK. It lets you save up to 1000 files in read-only format. For more information about this and other products, send a self-addressed, stamped envelope to Chessoft Ltd, 723 Barton Street. Mt. Vernon, Illinois 62864.
JOHN MENKE
MT. VERNON, IL

Gazette, September 1994

D'IVERSIONS: Paper-Training Sparky the Dog

By Fred D'Ignazio

The other day I attended a speech given by a nationally renowned demographer. At the end of the speech I had the opportunity to talk with him for a few minutes. I introduced myself as a technology writer, and the demographer nodded and said he used a computer every day. His profession tracks large groups of people, their movements, their aging, and their behavior. He said that he considered the computer to be his single most valuable tool.

On the other hand, he was alarmed by the pace at which new, more powerful computer chips were being introduced. "All I need," he said, "is enough power to crunch my numbers and store and manipulate some data. And I'm a demographer. Who could possibly need more computer power than I?"

As a multimedia enthusiast and computer educator I was stunned. It took me a few minutes to summon enough courage to respond to his challenge. "Excuse me, sir," I finally said. "If you get in my car, I can drive you to a local elementary school and show you some fifth graders doing a multimedia history project for our college museum. They're desperate for more computer power."

"What could they possibly be doing?" the demographer asked.

"They're part of a multimedia detectives project which gathers nontraditional resources to research historical mysteries. As part of their research on the Civil War, they're digitizing recorded voices from Civil War soldiers and ex-slaves. They're scanning photographs passed down over generations by their families. They're videotaping live reenactments of dramatic events and personalities from the war. They're orally narrating eyewitness accounts of the war written by women, blacks, Southerners, and soldiers. They're digitizing artifacts from the war uncovered in family attics. They're interviewing local historians and pulling in pictures and sounds from libraries on the Internet. They're...."

"Enough!" said the demographer, smiling. "You made your point. Maybe I should come see what these fifth graders are up to."

MULTIMEDIA CUISINARTS

We adults, like the demographer, live in a world of text (numbers and letters), and we are tickled pink with the way the computer zips along, shuffling our words, sentences, and numbers. We're almost unaware, however, that a new age of knowledge is dawning in which computers will be required to push around digitized movies, voices, paintings, and symphonies as well as words and numbers.

Knowledge processors of the future will have to be multimedia cuisinarts that take images, sounds, and numbers and slice them, dice them, blend them, and puree them. To do this in realtime, they'll have to be far more powerful than the wimpy little word processors most adults are using today.

A NARROW TRICKLE OF TEXT AND TALK

In my multimedia speeches and demonstrations, I spend just as much time running up and down the auditorium aisles, Geraldo-style, as I do pressing buttons and switches up onstage. Since I am probably 75-percent clown, it seems natural in my presentations to turn my body into a comic, visual metaphor.

"It's so hard," I tell audiences, "for us adults to see things through the eyes of our children. We grownups are hooked on words. And, as the older and presumably wiser human beings in any room, we're great at frontal lecturing. We stand in front of young people and become a stream of words, spoken one at a time, dribble, dribble, dribble, pointed at their young ears.

"We assume that if enough young faces are pointed back at us, the stream of words is flowing between their ears and into their brains. We conclude that learning has taken place, but we're fooling ourselves. As good teachers already know, teaching isn't talking, and learning isn't listening---especially when your learners are all fish."

At that point I jump off the auditorium stage and go running up and down the aisles, arched forward, my hands folded together like the prow of a ship. "Our children," I say as I run, "are fish swimming through a sea of electronic media. This is their world of knowledge, their habitat. Each morning they're tossed through their classroom door into our world of words. No wonder they thrash and struggle! They can't breathe! They're like fish beached on a dry, arid shore. We try to help them, but all we can do is offer them this narrow trickle of text and talk."

I point my "ship" down a new aisle and run even faster, my hands pointed forward. "Text and talk," I say, weaving back and forth, "Text and talk. We think we're nourishing our children, and all the while they're suffocating."

CULTURAL BLINDNESS

We're not doing this on purpose. Most of us adults are not naturally mean, despite what many kids think. We really are people of good will, but we may be terminally blind.

As with any cultural transformation, the inhabitants of the old culture (the world of printed words) can't see the new culture coming. And the inhabitants of the new culture (electronic media) can't understand why most of their world is so foreign to the older persons they see everywhere around them.

Let's face it, we big people love books. We have spent our lives in the company of books. If you added up all the books we've stuck our noses into, you'd be amazed. Even worse, add up all the inches of text we've followed, line after line, page after page, as we've read books over 20, 30, or more years! We've spent our lives in book school, learning this simple equation: KNOWLEDGE = BOOKS.

School is the center of this theory of knowledge. The specialists of book-centered knowledge teach in the schools. Therefore, their methodology is straightforward: If you want to know something, find it in a book.

ENTER SPARKY THE DOG

And what are books made of? Paper! This is where I whip out a newspaper and throw it onto the floor. I fall to the floor and begin happily sniffing the newspaper, nuzzling it and talking to it in dog language. It's clear that I really love this newspaper!

As I'm scurrying around on the newspaper, I continue talking. "I'm an author," I say between barks and snorts. "That means I love words. I adore paper. In fact, you might say I'm paper-trained."

At this point I act as if I'm being led away from my paper on a leash. I resist the leash and gaze back longingly at the newspaper. I whimper and yip pathetically as I am dragged away from my paper. "I can't stand being away from paper," I say between growls and moans. "If I have to leave the world of paper, I get anxious and uncomfortable, like Linus being separated from his beloved security blanket."

I pretend I yank my head so hard that the leash snaps. I am now 20 feet away from my newspaper on the far side of the stage. Joyously I scamper on all fours back to the paper. I plop down on the paper and wag my tail against the paper. "Ahhhh," I say with a big doggy grin on my face. "Paper...mmmmm...I am so relieved."

I AM SO CONFUSED!

I jump up. I'm a human again. "I may be the silliest Sparky the dog in the room," I say, looking around the auditorium. "But I'll bet I'm not the only one who is paper-trained. This paper-training is shaping up to be a serious disability in the world of the future because knowledge is packaged in new nonpaper formats. We book lovers may feel very strange in a world where knowledge no longer comes on paper, neatly and politely, one word at a time. Instead, knowledge is crammed inside a shiny silver platter or whizzes onto our TV sets and computer screens from libraries and databases around the world. This is a brave new world for Sparky the dog!"

Then I show a video from the MCI Corporation which talks about the Internet and hypermedia libraries of the near future. An actor dressed as a Renaissance scholar lights a candle and enters into the darkened library while classical music plays in the background. He wants to look up information about Columbus's voyage to explore a new world. A modern woman, the librarian, tries vainly to help him, talking about hypertext and multimedia archives stored in "infinite digital preservation in realtime." The poor man shakes his head, bewildered, and says he'd love to understand the new scheme of knowledge, but he fears he will be in his grave before he learns to navigate through this new world.

ESTHER WILLIAMS OR ARNOLD SCHWARZENEGGER?
We are this old man. All our paper-training has left us unprepared for the new ways in which knowledge will be packaged, dished up, and devoured. We have to decide really soon whether we are really in love with books (the comfortable old medium) or with the ideas, the life stories, the treasures found inside.

There is hope. The trick is to take the first baby step into this brave new world of our children. Or is it really a step?

I ask the audience if they ever heard their parents or grandparents talk about the Hollywood swimming star Esther Williams. A Florida educational TV producer, Jane Matheny, challenged me to transform my paper metaphor of children swimming through a sea of media into a physical realworld metaphor.

One spring morning I met with Jane and her camera person at a Florida poolside. I stripped down to my swim trunks and revealed my not-so-Schwarzenegger torso underneath. While I was disrobing, I was to talk about the sea of media which represents the world of knowledge in the future.

DO-IT-THE-HARD-WAY FRED
While I was spouting words, I was in my medium, but then I ran into a problem: I was supposed to keep talking and also jump into the pool! It's kind of like walking and chewing gum. This is not a hard proble for Esther Williams or many other people, but for a person nicknamed "Do-It-the-Hard-Way Fred," it was a fearful challenge.

After 17 film takes and nine painful belly flops, the producer said she'd had enough. With some skillful editing, the video segment eventually aired on Florida public television. The shoot at poolside was supposed to take only half an hour, but the producer hadn't counted on all the times I would get water up my nose while I was pretending to be a child swimming through a sea of electronic media. Choking and coughing I would rise from the water like a whale breaching, and we'd have to do the whole thing all over again, starting with another belly flop.

It was almost the end of me, but the piece was a success. In it I asked teachers to come with me and jump into this new sea of electronic knowledge. We may almost drown, and we may have to leave our paper high and dry on the shore. But the time has come for us to decide. In the multimedia world of the future, who will be our role model? Will it be Willy, the whale who leaps to freedom, or good old Sparky the dog?

BEGINNER BASIC: More Variables and Constants

By Larry Cotton

I ran across an interesting program the other day, one written for one of those "other" computers. It's called S. I. Plus and was copyrighted back in the software Stone Age of 1988 by a company called Geocomp Corporation. What does the program do? For one thing, it tells me that my car gets about 1800 furlongs per ferkin, and that I am about 1.933012E-016 light-years tall. Yep, it's a very thorough units-conversion program.

The next time you hear the old proverb "Give him an inch and he'll take a mile," you'll be able to add some zing to it by converting it to "Give him in a barleycorn and he'll take a parasang". (A barleycorn's about 1/3 inch; a parasang's about three-and-a-half miles.)

Always on the alert for fodder for this column, and remembering that we're in the middle of a discussion on constants and variables, I thought we might have the makings of an interesting (albeit simpler) BASIC units-conversion program.

Let's think about the program's structure. Obviously, we'll need a menu; I like white characters on a dark blue screen, so here's the code for that.

```
10 PRINTCHR$(147):REM CLEAR SCREEN
20 POKE53280,6:REM BLUE BORDER TO MATCH SCREEN
30 POKE646,1:REM PRINT EVERYTHING WHITE
```

What's involved in units conversion? We need words such as "foot," "inch," "pound," and "ferkin" and conversion factors, such as 16 and 9 (for 16 ounces per pound and 9 gallons per ferkin). These should be set up early in the program as constants. For instance, since there are 25.4 millimeters in one inch, 25.4 should be set up as a constant.

Constants like these are best stored in arrays. Last month, we touched on arrays, which are just groups of numbers or words with numbered tags like C(11) or E$(3), so they can easily be accessed and manipulated. In line 40, we'll reserve computer RAM for three one-dimensional arrays: C(1) through C(16), E$(1) through E$(16), and M$(1) through M$(16).

```
40 X=16:DIMC(X),E$(X),M$(X)
```

You have to use DIM to reserve space when an array will contain more than ten elements. Think of arrays as rows of labeled pigeonholes. A pigeonhole on the left could be labeled C(1). Later in the program, it will be loaded with a units-conversion constant (since it won't vary while the program runs). The rest of that row of pigeonholes will contain units-conversion constants with the names C(2) through C(16).

Likewise, arrays E$(1) through E$(16) and M$(1) through M$(16) will be loaded with string constants such as FOOT, INCH, MILLIMETER, and so on. (We may use abbreviations.)

This simple program will deal only with conversion of lengths from metric to English and vice versa; however, the principles can be used in conversions of area, volume, weight, pressure, or whatever. In fact, a main menu where the user can pick length, area, and volume can then lead to several submenus for each set of conversions.

Time to load the three arrays. We'll do that with a FOR-NEXT loop and a READ command.

```
50 FORN=1TOX:READC(N),E$(N),M$(N):NEXT
```

As N increases from 1 to 16, this line reads data from lines further on in the program. We need some conversion data to read, so here it is.

```
320 DATA .0000254,IN,KM,.0254,IN,M, 2.54,IN,CM,25.4,IN,MM
330 DATA .0003048,FT,KM,.3048,FT,M, 30.48,FT,CM,304.8,FT,MM
340 DATA .0009144,YD,KM,.9144,YD,M, 91.44,YD,CM,914.4,YD,MM
350 DATA 1.609344,MI,KM,1609.344,MI,M, 160934.4,MI,CM,1609344,MI,MM
```

Where did those line numbers come from? I've already written the whole program, so I know what they'll be. Normally, you would put data in some high-numbered lines and renumber them later. The data is arranged in groups of three: conversion factor, English unit, and metric unit.

Here's a simple menu.

```
60 PRINT"(1) IN TO KM (17) KM TO IN
70 PRINT"(2) IN TO M
(18) M TO IN
80 PRINT"(3) IN TO CM (19) CM TO IN
90 PRINT"(4) IN TO MM (20) MM TO IN
100 PRINT"(5) FT TO KM (21) KM TO FT
110 PRINT"(6) FT TO M
(22) M TO FT
120 PRINT"(7) FT TO CM (23) CM TO FT
130 PRINT"(8) FT TO MM (24) MM TO FT
140 PRINT"(9) YD TO KM (25) KM TO YD
150 PRINT"(10) YD TO M (26) M TO YD
160 PRINT"(11) YD TO CM (27) CM TO YD
170 PRINT"(12) YD TO MM (28) MM TO YD
180 PRINT"(13) MI TO KM (29) KM TO MI
190 PRINT"(14) MI TO M (30) M TO MI
200 PRINT"(15) MI TO CM (31) CM TO MI
210 PRINT"(16) MI TO MM (32) MM TO MI
```

When users reach this screen, they're asked to choose a menu number. The number just happens to be an index to the arrays. For example, C(5), E$(5), and M$(5) all relate to converting feet to kilometers. Can you begin to see the power of arrays?

```
220 INPUT"[DOWN]WHICH NUMBER";N
```

Check for a valid response.

```
230 IFN<1THENPRINT"[UP][UP]":GOTO220
240 IFN>32THENPRINT"[UP][UP]":GOTO220
250 IFN>16THEN290
```

Line 250 will send program control to line 290, which begins handling menu options 17-32, metric to English conversions. These conversions are somewhat more complicated; we'll tackle them next month.

Line 260 will begin the English to metric conversions, which are simpler to understand.

```
260 PRINT:INPUT"[DOWN]HOW MANY "E$(N);
```

Let's say the user wants to convert seven inches to centimeters. That is menu option 3, so the user enters 3 in line 220, making N equal to 3. Since N isn't greater than 16, control will go to line 260, where it first prints a blank line and the words "HOW MANY."

After "HOW MANY" is printed, the computer looks up E$(3), finding "IN" (for inches) and printing it next. Note carefully the semicolon just after E$(N). Next, line 270 will gather the user's input and calculate the conversion.

```
270 INPUTQ:A=Q*C(N)
```

Q is the number of units to convert, be they English or metric. In our example, Q is 7 and N is 3. So C(3), which is 2.54, is multiplied by 7 to get the answer A. Now we print the answer.

```
280 PRINT:PRINTQ;E$(N)" ="A;M$(N):END
```

First, the program prints a blank line. Next, we print the value of Q. If a semicolon were not used, the computer would attempt to look up the value of QE$(N), which doesn't exist. Be very careful with semicolon placement.

The last items in line 280 are an equal sign, followed by the answer A and M$(3), which is CM for centimeter. Thus the printed answer line should look like the following line.

```
7 IN = 17.78 CM
```

Next month, we'll go the other way and complete the other half of the conversion program.

MACHINE LANGUAGE: Counting Up/Down

By Jim Butterfield

Loops are one of the most fundamental program structures. A popular type of loop is the counting loop, where the action is repeated a fixed number of times. We'll talk at some length about the way you can set them up and whether it's best to count up from 0 or down to 0.

We'll use a simple program to illustrate the various methods. The program will print the message HELLO several times, with each output line using a different counting method. To round things off, the program will go through an addition loop; in this case, multibyte numbers will be added using a loop to go from byte to byte.

PROGRAM COUNT

The illustrative program, Count, is located on this disk. It pokes the machine code into place and then runs it. Here, we'll go through the code, commenting on the various approaches.

As for counting, my recommendation is to count up to avoid complexity. Some programmers are down-counting fanatics, and we'll show those methods, too.

Occasionally, it's important to know how many bytes are in a loop. You're not likely to run out of memory space in a program, but a smaller loop often runs faster. On rare occasions, speed is important. For these reasons, the actual machine code will be shown here.

USELESS LOOP: CLEAR COUNTER

The first loop in our program sets both bytes of a counter to 0. The counter is at address hex 2080 and 2081. Remember, the two-digit values on the left are the hex machine code; on the right is our symbolic assembler coding.

```
    ; set counter to 0
    a2 01              ldx  #1
    a9 00              lda  #0
    9d 80 20    lp0    sta  cntr,x
    ca                 dex
    10 fa              bpl  lp0
```

We are using a simple loop which counts the value in X downward. We do not benefit from a loop in this case, since two STA (STore A) instructions would do the job faster and more compactly. But if our counter were ten bytes in length, rather than two, the loop would make a lot of sense.

SIMPLE UP-COUNTING

The message HELLO, complete with following space character, is stored at $2070 (symbolic address hifwd). Counting up is easy: We start at 0, tracking the "H" in HELLO, and count each letter as we print it with a call to $FFD2. When the count reaches 6, we will have printed all the characters and may drop out of the loop.

```
     ; method 1 - counting up
     a2 00              ldx  #0 ; start at 0
     bd 70 20     lp1 lda  hifwd,x
     20 d2 ff           jsr  $ffd2
     e8                 inx       ; count up
     e0 06              cpx  #6   ; until 6
     d0 f5              bne  lp1
```

It's simple. The programmer is unlikely to get the count value muddled. The letters of the message are arranged in forward order, which makes the message easy to code.

But some advocates of down-counting say that they can make the loop shorter, and thus faster.

SIMPLE DOWN-COUNTING
Suppose we arrange our HELLO message backwards (OLLEH, with a space at the start). You'll find this reversed message stored at address 2077. We could then count downward to save a couple of bytes and gain a little speed.

Even though we plan to output six characters, we must load X with a value of 5. That's because we'll be using the 0 value of X, too.

```
     ; method 2 - counting down
     a2 05              ldx #5 ; start at 5
     bd 77 20     lp2 lda hibak,x
     20 d2 ff           jsr $ffd2
     ca                 dex      ; count down
     10 f7              bpl lp2  ; til neg!
```

It's short, but perhaps not so sweet. The coding looks clumsier. Here's another disadvantage: Since we're testing X for a positive condition (BPL is Branch PLus), we can count no higher than 128. We can partly fix these problems, however, by using "address adjustment."

DOWN-COUNTING REVISITED
It was noted that reverse string hibak was at address 2077. By using the next lower address, 2076, we can overcome the 128 limit. The programmer is also allowed to use the real string length of 6.

Even though we specify address 2076, the loop never goes that low. The smallest indexing value will be 1, so address 2077 is the last one accessed.

```
     ; method 3 - counting down (2)
     a2 06              ldx #6 ; start at 6
     bd 76 20     lp3 lda hibak-1,x
     20 d2 ff           jsr $ffd2
     ca                 dex      ; count down
```

```
    d0 f7          bne lp3 ; til zero!
```

Counting down seems to be pulling ahead. It's shorter and faster. But wait: We can rehabilitate counting up.

SON OF UP-COUNTING

If we really need to squeeze a couple of bytes and a few microseconds, we can still do it by means of counting up. We'll just have to study some new concepts.

A value of $FF often means the same as decimal value 255. But, if you want it to do so, $FF can represent a value of negative 1. Think of it in terms of a tape counter showing 9999.

You can't really have negative-value indexing except with 0 page addressing. You can get that effect, however. Here's how.

Suppose we have an address we're interested in, such as $1234. And suppose we want to index that address by negative 1, with $FF in the X register. A little thought will show that adding $FF to $1134 gives the desired result of $1233. Now, $1134 is exactly $100 below $1234. So, if we subtract hex $100 from the reference address, the value in X will become a "negative index." You may need to think about this and try a little more arithmetic, but it works.

OK, so we want to work addresses below the end of our HELLO string. That string is at hifwd or 2070. The end of the string is at hifwd+6, or 2076. And hex 100 bytes below that are hifwd+6-$100, or $1f76. Thank heavens most assemblers will do this work for us.

Here comes our up-counting negative-indexed loop.

```
    ; method 4 - counting up (2)
    a2 fa          ldx #$fa ; neg 6!
    bd 76 1f   lp4 lda hifwd+6-$100,x
    20 d2 ff       jsr $ffd2
    e8             inx      ; count up
    d0 f7          bne lp4  ; til zero
```

It's just as short and fast as the counting-down method. It can handle a full 256-iteration loop. And it leaves the text in a more readable form.

If you don't need those bytes and microseconds, a conventional up-counting loop will do nicely.

MULTIBYTE ARITHMETIC

If you wish to use a loop to step through the individual bytes of a multibyte number, go ahead. You should, however, be careful with arithmetic operations such as addition, subtraction, and the rotate/shift instructions.

These instructions use the Carry flag to carry data from one byte to the next. If your looping method should change that flag, you'll have a problem. The Carry flag is most often changed this way by a compare instruction: CMP, CPX, or CPY. Avoid these instructions or be prepared to do extra coding to preserve the Carry flag.

The loop shown here uses down-counting and BPL to do its addition job. There are no compare instructions.

```
    ; add 7 to counter
    a2 01             ldx #1
    f8                sed
    18                clc
    bd 80 20     lp5  lda cntr,x
    7d 7e 20          adc incval,x
    9d 80 20          sta cntr,x
    ca                dex
    10 f4             bpl  lp5
    d8                cld
```

The code to output the BCD number calculated above will not be shown here. For your information, here are the string and variable definitions.

```
     hifwd     .asc  'hello '
     hibak     .asc  ' olleh'
     incval    .byte 0,7
     cntr      *=*+2
```

CONCLUSION
Counting upward makes life easier, but it's not the only way to do things. Any way you can get your loop to work is valid.

Gazette, September 1994

PROGRAMMER'S PAGE: Screen Savers

One of the hottest and yet oddest fads to hit the IBM world in recent times is that of screen savers. Originally designed to stop an unchanging display from burning itself onto your monitor's screen, screen savers have exploded into a plethora of Star Trek scenes, flying toasters, and pistol-packing penguins.

They trace their origins to a problem as old as computing. Back in prehistoric days, there was an addictive game called Pong. It was very simple--so simple, in fact, that the screen rarely changed. Many people enjoyed playing it, and many left it running so they could have a quick game when they wanted. The problem was, after a weekend on the TV, the Pong image was permanently burned into the screen. The unchanging image eventually damaged the phosphor coating of the TV screen, leaving a ghost image of the game.

Although the problem of burn-in has existed for many years, nothing much has been done about it until recently. Now we have such a variety of programs purporting to save our screens that the major choice in computing isn't what spreadsheet or word processor to use, but what display to have on the screen when we're not working.

BUT WHAT ABOUT US?

It isn't fair that the IBMs and compatibles should have all the fun. Although IBMs may have a tad more processing power, 64s need their screens protected just as much. With this need in mind, Martin Fensome of Richmond, British Columbia, Canada, has thoughtfully provided a screen saver for us. It's on the flip side of this disk as SCRNSVR1.

Although the program listing isn't much to look at (being mostly machine language), its results are. Like other screen savers, this one waits in the background, waiting for you to press a key. If nothing is typed on the keyboard after a given period of time (typically 10 or 15 minutes), the screen saver starts up.

Martin's routine sets the screen background to the text color, making the text disappear. At this point, any keypress (including Ctrl, Shift, or the Commodore key) returns the display to normal.

To use, just load and run it. The delay before the display changes can be set from .1 to 18 minutes. Set it for .1 minutes, or six seconds, when you want to see the effect right away; for normal use, a range of from 5 to 10 minutes is suitable. To disable the program at any time, type SYS 886, or hit Run/Stop and Restore. Enable it again with SYS 880.

A second screen saver, adapted from Martin's, simply blanks out the screen--no colors, but also nothing to burn in. It's called SCRNSVR2. This program is also on the flip side of this disk. Remember to load and run both of these screen savers manually.

HOW THEY WORK

Although there are various ways to program a screen saver, one of the easiest is to patch into the interrupt program. Sixty times a second, the computer stops normal work and executes this section of machine language, which checks the keyboard, adjusts the clock, and performs various other chores. Afterwards, it resumes normal processing. What screen savers do is patch themselves into this routine and get executed at the same time.

Usually, they just keep count of the quiet time at the keyboard. But if there is a long enough pause with no typing, they spring into action, changing the screen color, blanking it, or whatever. Then, the routine waits for a keypress to reverse the effect and return to normal.

Since this interrupt routine gets called 60 times a second, there isn't much time for fancy work. To give ourselves more time, we have to work a little harder. The computer keeps track of what it was doing before the interruption. What we can do is fool the computer into thinking it was running our screen-saver program before. This means that when the interrupt is finished, it will execute our screen display. And when the screen effect is stopped (by someone pressing a key), we fool the computer into thinking it's right back running the original program again. This is the method used by the following program.

```
100 REM  FIVE SCREEN-SAVER EFFECTS
105 REM
108 IFPEEK(789)=192THENSYS49152
110 PRINT"POKING";:FORJ=49152TOJ+403: PRINT".";:
READX:C=C+X:POKEJ,X:NEXT
115 PRINT"[DOWN][DOWN]":IF C<>52401 THEN PRINT"ERROR IN DATA!":STOP
120 INPUT"MINUTES TO WAIT
(.1-18)";X:IFX<.1ORX>18THENPRINT"INVALID":GOTO120
125 X=X*60*60:Y=INT(X/256):X=X-Y*256: POKE49155,X:POKE49156,Y
130 PRINT"[DOWN][down]WHICH SCREENSAVER EFFECT TO LOAD:"
135 PRINT"[DOWN]1-INVERSION EFFECT
140 PRINT"2-RANDOM EFFECT
145 PRINT"3-INCREMENT EFFECT
150 PRINT"4-HORIZ.SCROLL EFFECT
155 PRINT"5-TERMITES![DOWN]"
160 INPUTX$:X=VAL(X$):IFX<1ORX>5THEN PRINT"INVALID":GOTO130
165 Y=49920:IFX>1THENFORI=1TOX-1: FORJ=0TO-1STEP0:READJ:NEXT:NEXT
170 READX:IFX>=0THENPOKEY,X:Y=Y+1: PRINT".";:GOTO170
175 SYS 49152:PRINT"[CLR][DOWN][DOWN] SCREEN SAVER INSTALLED![DOWN]"
180 PRINT"USE SYS 49152 TO ALTERNATELY ACTIVATE
185 PRINT"AND DEACTIVATE IT[DOWN]"
190 PRINT"POKE 49159 WITH A VALUE TO CHANGE EACH
195 PRINT"SCREENSAVER'S EFFECT
200 PRINT"[DOWN][DOWN][DOWN][DOWN] [DOWN]NEW[UP][UP][UP]":END
205 :   IRQ SCREENSAVER CALLER
   [DATA statements from 210-315]
320:   INVERSION EFFECT
```

```
   [DATA statements from 325-335]
340:   RANDOM MESS EFFECT
   [DATA statements from 345-365]
370:   INCREMENT TO BLACK EFFECT
   [DATA statements from 375-390]
395:   SIDEWAYS SCROLL EFFECT
   [DATA statements from 400-455]
460:   TERMITES!
   [DATA statements from 465-650]
```

This program runs from the Gazette Disk menu. When run, the program pokes two sections of code into memory: the IRQ interrupt patch (lines 205-315) and a screen-effect routine (one of the five sections from line 320 on). The IRQ section sets up memory, starts up the display, and recovers when the screen-effect routine ends; the screen effect gets to do the fun stuff. A menu lets you choose which of the five included screen displays you want to use.

Following the selection, you enter a time delay, and the program is started with SYS 49152 (line 175). SYS 49152 alternately activates and deactivates the screen saver, patching it into the interrupt or removing it.

To see the effects best, leave some text on the screen, such as a program listing. The effect can also be varied for some of the displays by poking different values to location 49159.

The first four routines were written especially for use with the program, but the fifth is adapted from a submission. Donald Klich of Mount Prospect, Illinois, wrote a fascinating screen display involving programmed termites. When it starts up, two or three termites start moving. They obey two simple rules: If the next square to move to is colored, blank it and turn right; if it is blank already, color it and turn left. Keep watching it, and you'll be amazed at the patterns that emerge. And if you watch long enough, you'll even see little termite wars!

A PROGRAMMING CHALLENGE
As has already been mentioned, the machine code is written in two parts. The first, the IRQ patch, does all the countdowns, housekeeping, memory management, and so on, and then calls the screen-saver effect. After, it restores memory and recreates the original screen. Since it does most of the work, the actual routines for the effects can be quite small. All they have to do is fiddle with the screen.

Written in this way, a screen routine can be created and tested separate from the interrupt routine, which makes things much easier. After the effect has been tested, it's added to the screen-saver program. Just about any screen effect can work, including ones you may have already written. The only requirements are that they start from location 49920 (hex $C300) and that they loop endlessly (the IRQ patch will take care of stopping them).

The screen effect can't overwrite anything in memory, of course, but there are some places that are fair game. Locations 2-42 and 200-255 in page 0 and all of screen and color memory are restored after the effect is ended and so can be used. Also, the VIC chip is restored to original values, so colors, graphic modes, and so on can be changed with impunity.

What I want are submissions with screen effects for the 64. Remember the guidelines, and send me your code. Anything interesting is welcome, with the more off-the-wall and exotic effects preferred. When we get enough good effects, I'll present them in this column.

In addition to the fame and money, you'll have the deep satisfaction of knowing that your display will help entertain thousands of 64 users who should be doing something useful with their lives instead of looking at a screen saver. So get cracking! Send your screen savers (or other tips) to Programmer's Page, COMPUTE's Gazette, 324 West Wendover Avenue, Suite 200, Greensboro, North Carolina 27408. We pay up to $50 for each tip published.

Gazette, September 1994

GEOS: GEOS Isn't Perfect

By Steve Vander Ark

Don't let the title of this piece cause panic, I haven't gone over and joined the Cardassians or anything. I still think GEOS is the best thing ever to happen to the Commodore 64 and 128 computers. As I mentioned last month, I'm always quick to argue in favor of GEOS when I'm online on GEnie, but it's because of those discussions that I've started to think about the flaws some people find in the system. So, I figured it was time to address some of these concerns.

The single biggest complaint I've heard about GEOS is its speed. GEOS doesn't have any. It's slow to load, slow to scroll through geoPaint documents, slow to update a geoWrite page before going to the next page, and so on. This is absolutely true. Don't believe it? Try changing something on a page while working with a large geoWrite document and then going to the next page. The pause while the system updates the whole document is so long that you could probably eat lunch while you're waiting. The first time this happened to me, I was sure the system was locked up.

The system takes quite a while to boot as well. What makes this hard to defend to a non-GEOS user is that a Commodore without GEOS is ready to go as soon as you turn it on--providing you don't mind typing in your commands instead of pointing and clicking. So why wait around just to get things started? If you then need to load files into a RAM device during boot up, the problem is worse yet.

The reason for the slow performance is simple: The system has to wait for the disk drive. When you're loading GEOS, for example, a number of small programs are loaded into the memories of both the computer and each disk drive. Some of these, the ones called Auto-Execs, are run and then removed from memory after doing things like setting your preferences. Obviously, your system is only going to do this as fast as your disk drive can find and send the required files.

GEOS uses the disk drive even more creatively once you've passed the boot process. The disk drive becomes additional memory, available for storing information which couldn't possibly fit in the Commodore's tiny amount of RAM. When a desk accessory pops up on the screen, the image of what was in that spot on the screen gets saved to the disk drive. When you're using most GEOS programs, they have many more routines than will fit into the computer's memory at one time. The disk drive stores these routines until you call for them. Next time you select the Text tool in geoPaint, watch your disk drive light as the system loads into memory the program code for that routine. All this means that GEOS gives a 64K machine far more than 64K of power. GeoPublish, for example, is almost 100K, the price you pay for all those extra capabilities.

So detractors are right when they complain that GEOS is awfully slow at times. There are Commodore programs that will do a lot of the same things but without the wait time. So why use GEOS?

Well, first of all, it's important to note that there actually are no programs that can do everything the entire GEOS system can do. Sure, you can find a great word processor or a good spreadsheet for the Commodore, but they won't work as a team with your other programs. And there are a lot things you simply can't do without GEOS, such as true desktop publishing.

Second, there is no need for any serious GEOS user to put up with the wait time. There is a simple solution to all that hanging around while the disk drive does its thing: Get a faster disk drive. What I'm talking about here is more than just another mechanical drive, the kind that you have to put a disk into. No matter which of those you get, even one of the new superpowerful FD drives from Creative Micro Designs, you will still have to wait while it spins the disk and moves the head around looking for files. There's a better answer.

GEOS is designed to use a RAM device as a pseudo disk drive, and a RAM device doesn't make you wait at all. It has no mechanical parts to chug around, no electric motors to spin. A RAM device operates at the speed of the electrons rushing through its circuits, which is pretty darn fast. This means that your Text tool in geoPaint loads in less than a second. Updating pages in geoWrite takes mere seconds instead of the minutes it takes with a mechanical drive. And booting takes just 12 seconds flat.

Wait a moment...booting? Sure! A RAM device such as CMD's RAMLink, which saves the information stored on it even when your computer is off, can be configured to boot GEOS from its RAM disk drive. In order to do this, you'll need to be using Gateway, CMD's file manager, but since that's one of the best file manager software packages around, you'll get a lot of benefits besides fast boot times.

There is one complaint that I have no answer for. In order for GEOS to run at these kinds of speeds, you need to shell out some money. There's no way around that. A RAMLink with 1MB of memory will set you back $229.95, and that's without such niceties as the realtime clock chip or a battery backup for those thunderstorm nights. The Gateway program is $29.95. Even a regular RAM expansion unit, without extra features, costs around $100.00 these days.

All I can say is it's worth it. A GEOS system running at top speed is a joy to use. When you include a lot of excellent shareware/public domain programs such as the ones I feature in this column regularly, there is very little you can't do with your Commodore.

Gazette, September 1994

PD PICKS: Titles, Slowpoke, and Spooky Eyes

By Steve Vander Ark

Oh, I'm having fun now! And get this: I'm not playing games! Nothing is blowing up; no grim legions of vicious monsters are trying to blast me to smithereens. Nope, I'm having fun with utilities.

I know. That sounds like a low-budget public access television show (which might actually be kind of neat), but it's really just my way of living up to the boss's expectations. You might remember that a few months ago my boss at Gazette told me to start writing up a few more utilities. Well, he's going to love me this month. I have two beauties that I think a lot of you will like, plus a little bonus that's not exactly a utility but....

OK, OK, here's the list. Once again they come from Jim Green's excellent disks of shareware and public domain programs. So here's a big thanks to him for sending me the disks.

TITLES
By David Rose

When I bought my first VCR, way back in the late seventies, the price for a single one-hour videotape was well over 20 bucks. I remember standing in the store, figuring out how much it would cost me to get all 79 episodes of "Star Trek" on video. I remember deciding that I'd never be able to afford it. Now, at three dollars for a two-hour tape, I can easily afford to record all the episodes of not only the original series but also "Star Trek: The Next Generation" and "Deep Space Nine" as well.

I don't tell you this to brag about my video collection. I mention it because it shows just how commonplace video has become in our society. Videotapes cost so little because stores sell so darn many of them. And they sell so many because practically everyone owns a VCR.

That's why I think this program will be a helpful addition to your collection of utilities. What it does, quite simply, is create title screens that you can use on your videotapes. It allows you to specify the text you want displayed on the screen and then makes it look a bit fancier than just plain old Commodore text. With very little trouble you can then record that image onto your tapes to create beautiful titles.

It's really easy to record the image from your computer screen. My VCR has a jack marked "Video In," so I hook up my video cable to that. I have to press a switch to set the VCR to accept what it calls "Aux Input" in order for it to take its signal from that jack. Then, when I see an image on the screen that I want to capture on tape, I just start recording. It works great. It's so easy, in fact, that my third grade students do it for class projects.

Of course, any screen image can be used for a title, but the Titles program makes the whole process quick and easy. You type in the text you want displayed and specify which font to use from the four provided. Then your message appears in oversized letters on the screen. You can adjust the colors of the screen border or background as well as the color of the text by pressing the function keys. You can also use the function keys to change the font you're using. There are several other nice features to make things easier. For example, you can force the program to keep several words together on a line by separating them with a shifted space. This keeps the word-wrap function from splitting things awkwardly.

The program also has a built-in return loop, by which I mean that it lets you quickly and effortlessly go back and create a new title when you've finished with one. I always appreciate that in a program which is likely to be used several times in a row.

The documentation for Titles is included on this disk in the TITLES INST file. Since that file will also load and run the Titles program, it's on the flip side of this disk along with the main Titles program and its associated fonts. There are four font files included, plus one more called MENU.FONT, which the program also needs, although it doesn't show up as an option for making titles.

SLOWPOKE

Slowpoke, by an unknown author, is a tiny little program that does a simple little job. It slows down your computer. You can control how much slower than normal you want it to run.

But hold on. The 64 operates at only 1 MHz, which is ridiculously slow by today's standards. Why in the world would you want to slow it down further? There could be a number of reasons. For one thing, it forces the LIST function to place text on the screen at a much more leisurely rate, which is handy if you're trying to read a long directory as it goes by. Do you have a BASIC game that runs just a little too quickly? Slowpoke will apply the brakes.

You can control exactly how much to slow things down. Once installed, you activate Slowpoke from the keyboard by typing POKE 251, X, where X is some number between 0 and 255. The larger the number the slower your BASIC program will run. I don't recommend anything much over 100, only because a program or listing becomes tediously slow. The suggested value of 50 seems to be about right.

You might not be able to think of a use for Slowpoke right off the bat, but don't forget that you have it. I think you'll find that it really comes in handy somewhere along the line.

SPOOKY EYES
By PerryM4

I'm not saying much about this one. Just load and run it, and then sit back and watch. Oh, it may be just slightly rude now and then, but in a nice way. Anyway, the graphics are very well done, and the animations are effective. Enough said.

Gazette, September 1994

EXPLORING THE 6502

By Frank Gordon

Examine the orderly world of opcodes.

The 6502/10 microprocessor launched four of the earliest home computers: the VIC-20, Commodore 64, Apple, and Atari. It has 151 documented operation codes and the operating system (the Kernal on the 64) or any high-level language like BASIC must be translated into the 8-bit voltage patterns of these opcodes to access the microprocessor.

Most opcode listings are alphabetical and do not reveal these bit patterns. In the two programs that are associated with this article, you have the option of listing the codes in numerical order in binary, hexadecimal, and decimal with the mnemonic in the following form.

```
    00000000 $00 0 BRK.
```

If you wish an alphabetical listing, you also have that option from the Opcode menu. With both programs, you have the option of sending the output to the screen or printer.

USING OPCODES

As given, Opcodes will display a complete listing of the 151 opcodes in numerical or alphabetical order. After you run one program, you have the option to run the other, return to the Features menu, or quit to BASIC. You can explore various bit patterns by adding selection rules to Opcode. After you run the opcodes in numerical order (OPCODES.N) select quit to break out of the program and then list line 200. It reads GOSUB 210. Change it to read as follows, and then type RUN and press Return.

```
    200 Y=3AND A(X):IFY=1 THEN GOSUB 210
```

This gives all of the opcodes for ORA, AND, EOR, ADC, STA, LDA, CMP, and SBC. Each of these instruction mnemonics is identified by its three high bits as illustrated in the following partial list.

ORA	000xxx01
AND	001xxx01
EOR	010xxx01
ADC	011xxx01
STA	100xxx01
LDA	101xxx01
CMP	110xxx01
SBC	111xxx01

There are eight addressing modes for each instruction (except STA which has no immediate mode). Bits 2 to 4 (indicated above by xxx) define the addressing modes for each instruction above as follows.

```
    xxx00001  XXX (15,X)
    xxx00101  XXX 15
    xxx01001  XXX #15
    xxx01101  XXX 1500
    xxx10001  XXX (15),Y
    xxx10101  XXX 15,X
    xxx11001  XXX 1500,Y
    xxx11101  XXX 1500,X
```

The instruction and addressing mode combined will define the opcode. Now change line 200 to read as follows.

```
  200 Y=31 and A(X): IFY=13 THEN GOSUB 210
```

When the program runs this time, it will select opcodes with the same addressing mode xxx01101 XXX 1500 from this series, giving the same addressing mode from ORA to SBC.

```
    00001101 ORA 1500
    00101101 AND 1500
    01001101 EOR 1500
    01101101 ADC 1500
    10001101 STA 1500
    10101101 LDA 1500
    11001101 CMP 1500
    11101101 SBC 1500
```

There are many possibilities. Feel free to experiment with different groupings. Here's another example.

```
    200 Y=31 AND A(X): IFY=16 THEN GOSUB 210
```

This line will display all of the Branch instructions. These share the bit pattern xxx10000.

```
    00010000 BPL ADDR
    00110000 BMI ADDR
    01010000 BVC ADDR
    01110000 BVS ADDR
    10010000 BCC ADDR
    10110000 BCS ADDR
    11010000 BNE ADDR
    11110000 BEQ ADDR
```

UNDOCUMENTED OPCODES
In COMPUTE! magazine (October 1983), the article "Extra Instructions"

by Joel Shepherd shows how to create new undocumented opcodes. These codes are discussed further in two articles in COMPUTE'S Gazette (March 1993): "Secret 6502 Opcodes Revealed" by Randy Thompson on page G-20 and "Strange Opcodes" by Jim Butterfield on page G-18.

It becomes clearer how these undocumented codes combine two operations into one when you examine the binary forms. First, note that none of the 151 documented opcodes have the form xxxxxx11. You can check this by changing line 200 to the following.

```
200 Y=3 AND A(X): IFY=3 THEN GOSUB 210
```

Run the program again and note the absence of any output.

Now let's look at LDA and LDX.

```
10101101   173   $AD   LDA 1500
10101110   174   $AE   LDX 1500
```

When these are combined as shown in the following line, both the accumulator and the X register are loaded simultaneously with a value from memory.

```
10101111   173   $AF   LAX 1500
```

By running these programs, you should see that the bits that make up the 6502 opcodes are not simply selected from some random jumble but are actually derived from a pattern that's coherent and orderly.

Frank Gordon lives in Orono, Maine.

Gazette, September 1994

4-SKAN

A photo scanner that fits on your printer

Reviewed by John Elliott

No 64/128 scanner will transfer text to a word processor. It is possible, however, to move a two-dimensional drawing or photograph into the digital world of the 64/128. Until recently there were only two methods of loading a two-dimensional drawing or photograph into a 64 or 128. One was the $300 Handyscanner 64, which provides a high-resolution image. The higher the resolution setting, though, the more carefully the scanner must be drawn over the page.

The other approach requires a video camera to capture a still image of a page which can in turn be captured by a digitizer such as the $100 ComputerEyes. This cartridge, while still available, is no longer in production. Then, a third-party conversion program was needed to convert the image into most graphics formats.

Now, there's a $70 device which might be the scanner we've been waiting for. The new 4-SKAN consists of a fiber-optic cable whose one end connects to the computer's user port while the other attaches to the printer's printhead. The photo or image to be scanned is then inserted into the printer, just like a piece of paper. A photocell at the printer end registers the amount of light on the image as the printhead moves across the page. As the image moves through the printer, it is scanned, and the digitized information is sent to the computer.

That my Epson Homewriter 10 is a 1525 clone means that the photocell will register the amount of light every 1/10 inch as it crosses the page. While I am delighted with my results, printers with adjustable linefeeds will stop more frequently on each line and provide about twice the resolution I achieve.

The head of the fiber-optic link must rest on the printhead as close to the picture to be copied as possible. I tape the end of the cable to a groove in the printhead. I place a desk lamp about one foot from the printer to act as an effective light source. With the software that comes with the scanner, I can control how long the head and photocell stop at each point across the page. I can also assign a fixed number of dots to each sample taken.

The maximum size that 4-SKAN can digitize is an area 9.6 inches by 8 inches. The smallest object is a bit less than a 2-inch square. An image of a person's head on the cover of TIME magazine scanned at the minimum size resulted in a silhouette with no detail in my scan. A head at least 1/3 the size of a TIME page shows as much detail as the original photo. The heavier lined illustrations from a child's coloring book reproduce well. Pencil illustrations with thin lines do not register at all.

I had a problem with my equipment because my printer would miss lines that are less than 1/10 inch thick. This may be why solid objects, such as faces, provide the best results. Photocopiers which enlarge allowed me to blow up a 3-inch by 3-inch head and shoulder image to full page in size. My scan of the photocopy produced a full range of gray scales in what I found to be typical 4-SKAN soft focus. A copier's brightness control also let me make adjustments to how light or dark I wanted the image printed from 4-SKAN.

In attempting to digitize my signature, I have had to make accommodations to my 1/10-inch limitation. The writing from a pen or pencil would not register. A felt-tipped pen with letters 2 inches high produced a recognizable, although somewhat blocky, image. I got best results using a Magic Marker.

The enclosed printing program allows me to print various images from a saved scan by changing the exposure, contrast, and light ratio. The standard image displays six levels of gray. I can also choose an image font that resembles a woodcut or one that resembles a print by numbers which shows different numerals for each shade of gray. If you have a printer with variable linefeed and higher resolution, three additional fonts are also available.

This higher resolution would permit scanning of objects smaller than two inches square and allow for greater detail in larger images. Printouts would also have access to additional graphic fonts. An optional program which allows variable-sized printouts and smooths the rough edges of blocky pictures is also available for use if your printer can use variable linefeeds. This program also controls stretching of the width or height.

The 4-SKAN's designer has added a feature to this separate program which creates perforations around stamp-sized images. When printed with a single color ribbon on an envelope, for example, I'm told the results would puzzle postal authorities. Fan clubs which publish monthly stamps of their favorite stars should find this option useful.

An optional program which I found essential for my 1525 clone converts 4-SKAN files to Doodle format. While my 4-SKAN file may reach 200 blocks, Doodle's size depends on the height of the image. This could range from 8 to 37 blocks.

I changed the Doodle prefix from "dd" to "rph" and then loaded the image into RUN Paint where I was able to reverse the vertical or horizontal axis of the image, add additional images, and type anywhere on the screen. Fun Graphics Machine also uses Doodle's format without conversions. with a conversion program such as Grafixlink, you can convert Doodle images to most other graphic formats.

Until now, the only way I could fill a page with an image was to use a full page of geoPaint or a shareware program which quadruples the size of a Doodle image. With 4-SKAN, I can create 9.6- by 8-inch images, and unlike GEOS and Doodle, it allows a full range of grays. The result is soft focus, but when converted to Doodle, some shades are lost in favor of sharpness.

The designer of 4-SKAN described some GEOS documents he received that had 4-SKAN images printed between paragraphs of the documents. The Write Stuff word processor (Illustrator version) allows a similar approach once the 4-SKAN format conversions have taken place.

My main use of 4-SKAN is to create refrigerator art of full-page scans. Although I cannot claim the skills of a painter or photographer, I think I am becoming a good darkroom developer using the tools that come with 4-SKAN.

To get excellent results on your first try, use a full page black-and-white or photocopied photograph of a face. Good results will come with other subjects and sizes but will take experimentation. Also, expect to wait for your scan. While a small scan might take five minutes, a full page could take two hours using the default settings. You don't have to babysit the printer, though. You can read a book or cook supper and do your own version of parallel processing.

A $5 sample disk is available that will enable you to print and convert a sample 4-SKAN file to Doodle. It also contains the directions for scanning. It'll also let you discover whether your printer has variable linefeeds that allow for higher resolution than I was able to get with mine.

Despite my printer's limitations, I purchased 4-SKAN.

KALTEK
Adjuntas, Puerto Rico 00601-0971
4-Scan System - $69.95
Demo Disk - $5.00

Gazette, September 1994