THE COMPUTE!'S GAZETTE DISK

DECEMBER 1994

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

G A Z E T T E   O N   D I S K

D E C E M B E R   1 9 9 4

—— H A P P Y   N E W   Y E A R ——

Compute!'s Gazette Disk
December 1994
Received Disk: December 12, 1994

Printout on: Panasonic KX-P1180
Created By: Christopher T. Ryan

****

Loading Gazette Disk:

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 *** December 1994

Feature:

Wax Up Your Commodore; We're Going Internet Surfing

By Nick Rossi

You've all heard about the information superhighway, now learn about this huge network of computers and how you can tap into it with your 64 or 128.

Columns:

64/128 VIEW by Tom Netsel
The Internet may be a good place to find Commodore information online now that Q-Link is no longer available.

D'IVERSIONS by Fred D'Ignazio
Multimedia on a Shoestring

MACHINE LANGUAGE by Jim Butterfield
Simple String Sort

BEGINNER BASIC by Larry Cotton
Tracking Holiday Costs

PROGRAMMER'S PAGE by David Pankhurst
Winter Shopping

GEOS by Steve Vander Ark
Writing Patterns

64 Programs:

Bounce by Maurice Yanney
This is a multilevel arcade-style game for the 64.

Jump Start by Jim Hug
This is a programming shell for the 64 and for use with MetaBASIC.

Latin Study Guide by Frank Gordon
Use this study guide to help learn about the conjugation of Latin verbs and the declension of Latin nouns.

Tracktask by Joe Lawrence
Run two programs at the same time with this utility and cut and paste between them.

MetaBASIC by Kevin Mykytyn
This utility adds 32 new debugging and testing commands to Commodore 64 BASIC, working by itself or in conjunction with a machine language monitor/assembler.

Holiday Greetings
A Christmas demo from the early days of the Commodore 64.

Gazette, December 1994

BOUNCE

By Maurice Yanney

An arcade-style game for one player and a 64

Bounce is a game of timing and tact. As you release balls in trying to fill the containers, you must dodge those balls which did not go into the containers, but hurry -- time is running out!

Each level of Bounce contains a series of containers, each requiring a ball. The object is to release balls in a manner which will enable the balls to reach the empty containers. When a ball hits a wall, it bounces back in the opposite direction. To make things difficult there are moving columns, and there is a limited amount of time.

Bounce is written entirely in machine language, but it loads and runs like a BASIC program.

PLAYING THE GAME

Bounce is an easy game to learn to play; however, mastering it will require a lot of practice. The object is quite simple: move about the screen releasing balls in an attempt to fill the containers. To move use the keyboard I, J, K and M keys (to move up, left, right, and down), or use a joystick (in either port). Using a joystick will give you the added capability of moving diagonally.

To release a ball, press the space bar or the joystick button. Only ten balls can be moving at a time; attempting to release more balls will be unsuccessful.

Once a ball is released it will continue moving until it either fills a container or it makes contact with your player. Balls move from left to right and right to left until they hit a wall or a filled container. Once a ball hits a wall or a filled container, it will either move up or down a row and proceed in the opposite direction. When a ball hits another ball, the balls reverse direction (both left/right and up/down). If you are hit by a ball, a player is lost.

On the first level there are six containers which need to be filled before you can go to the next level. There is also one moving column on the first level. Each new level adds another moving column and six more containers. If all the containers are not filled before time expires, a player is lost.

On higher levels you are permitted to move further and further right but if you are hit by one of the moving columns, you lose a player.

The speed of the game picks up as you go to higher levels. So, while you will be able to move faster, the balls and columns will also be moving faster. On higher levels the time also decreases at a faster rate.

Each filled container results in 500 points. At the end of each level bonus points are awarded based on the amount of unused time. For each 10,000 points an extra player is awarded. You begin with four extra players.

You can pause the game by holding down the Shift key or by pressing the Shift Lock key. To continue playing, release the Shift key or press the Shift Lock key again.

Gazette, December 1994

JUMP START

By Jim Hug

A programming shell for the 64 for use with MetaBASIC 64

Jump Start is framework code written to be the starting point for almost any new BASIC project. It virtually eliminates the normal start-up headaches associated with new programs, such as designing the main panel and subpanels, selecting the colors to be used, deciding how the supporting code should be accessed, and so forth.

Jump Start provides the mechanics for a menu display, option selection and analysis, and branches to certain line numbers for placing subroutines. Jump Start works best when used in conjunction with MetaBASIC 64. First, let's discuss the assumptions:

Let's use MetaBASIC 64 as the supporting development tool. The FIND facility is invaluable in developing and testing a new program; the KEY definitions are useful and so are the great functions of MERGE, DELETE, NUMBER, SCRATCH, and TRACE. MetaBASIC 64 first appeared in COMPUTE!'s Gazette in February 1987. It also is available on Gazette's Power Tools disk.

Use predetermined line numbers, as explained below. Using a range of line numbers for different sections of code provides a level of standardization that eases the design and maintenance of any program.

Let's assume that many of your new programs will be menu-driven.

Here's a layout of our framework.

Generally, the BASIC line numbers have been allocated in groups of 100 for the first 1000 lines and then in groups of 1000 for each of the main menu options. Within the main option section, such as 1000 - 1999, I try to stay within a 100-line limit for each subroutine, that is, 1000 - 1099, 1100 - 1199, and so on. This also helps achieve modularity in the program structure.

LINES

Lines 0001 - 0010 are used for canned SAVE and VERIFY commands which ease the problems of version control. This also reduces the number of keystrokes required for these activities. Refrain from using line 0, because it will cause editing and execution problems.

Line 0011 - 0039 are allocated for initial setup of MetaBASIC, usually a standard set of KEY definitions, which can be changed as required for a given program.

Lines 0100 - 0199 are allocated for numeric variables, their initialization and definition. In larger programs it pays to keep track of the way the variables are used.

Lines 0200 - 0299 are allocated for string variables, initialization, and definition. (Lines 200 - 239 are for option text on the main menu.)

Lines 0300 - 0399 are allocated for array dimensions (DIM) and definitions.

Lines 0400 - 0499 are allocated for the main menu display, selection analysis, and branching to the handling section.

Lines 0500 - 0999 are allocated for initializing subroutines at 0500, 0600, 0700, 0800, and 0900. These are usually reserved for array initialization at program startup.

Lines 1000 - 8999 are allocated at 1000 lines each for the code corresponding to each of the options 1 - 8, thus allowing for 100 lines of BASIC code at 10-line intervals, or ultimately 1000 lines of BASIC per selection; that's usually enough. Note that these routines end with WAIT653,1 preceded by PRINT "SHIFT TO RETURN." This proves to be a neat way to momentarily stop the program execution, giving the user a polite chance to catch up with all information that's been displayed or to simply have time to see a displayed message before it disappears. To save space, make a subroutine at 900 and simply enter GOSUB 900 when you need a controlled pause.

Lines 9000 - 9999 are set up with the code to determine your remaining free space. It is necessary to display the code, blank out the line number, and press Return to execute the command.

Lines 10000 - 10999 are loaded with the command to print a listing of the current program. Again, it is necessary to display, delete the line number, and press Return to execute. However, with MetaBASIC I have set f2 for a LLIST command and supply the appropriate line numbers.

NOTE: If menu options 9 & 10 (0) are needed for more selections, then by all means use them; they're set to go.

Lines 11000 - 11199 contain a print-screen routine. I use it for troubleshooting a difficult program when its needed.

Lines 11200 - 11270 have a directory-display routine.

Lines 12000 - 12120 contain a subroutine to open and read files.

Lines 12300 - 12390 include a subroutine to format and align decimal numbers.

HINTS
The option selection analysis results in an ON GOTO command. The convention expects GOTO 400 as a final statement of each of the main

target routines.

I store several useful routines on different disks, but it's easy to append them as needed. This code was stripped from existing programs using MetaBASIC and then renumbered using line numbers in the 30000 range so that it could be easily appended to new programs using the MERGE function. After the routines are added to a program, they should be renumbered to suit that program and the section of code where they are actually used.

For example, let's assume that you are working on a routine for option 4 (BASIC lines 4000 - 4999) and you need an array processing routine that's on another disk. Using the MERGE function of MetaBASIC, you can load the array processor at 30000 - 30099 and then change the 30000 range numbers to those that will be appropriate for the routine at 4200 - 4299. Then you should delete the lines 30000 - 30099 to save space or to make room for any other archived routines you want to merge.

Any variables in the newly loaded routine should be appropriately mapped or changed to suit the existing code. Once this is done, a pretested routine is usually ready to go in a half hour or less. Since you may have made changes to it, some testing is required, but the basic structure should be intact and workable. Watch out for branches such as GOSUBs that suddenly do not have their original target addresses.

In statements 1 - 10, I have placed the following BASIC instructions.

```
1 GOTO 100
2 SAVE"SKELETON PRGC",8:REM AA,AB
3 REM
4 REM
5 REM
6 VERIFY"SKELETON PRGC",8
10 SYS4096*9: POKE40341,7
```

Statement 1 is used to branch around the "housekeeping" code (1 - 99).

Statement 2 is a canned SAVE command which I have found most useful. By changing the last character of the program name immediately after the load, you then have a way to store your latest work to disk. All you have to do is list the first ten statements, delete the number 2 in BASIC line number 2, and press Return. This will save your program in immediate mode.

When the SAVE command completes, the computer automatically positions the cursor at the VERIFY. By blanking out the line number on line 6 and then pressing Return, a VERIFY is performed. With practice you can start the VERIFY before the disk stops rotating, saving the mechanical startup wear and tear on your disk drive.

Statement 10 will re-enable MetaBASIC after you break out of the program by pressing Run/Stop-Restore. I do this quite often during the development process.

You can use the same technique for the KEY definitions in statements 11 - 31 as used for the SAVE/VERFY commands mentioned earlier. Simply list lines 11 - 31, blank out the line number on each KEY statement, press return with your cursor on that line, and MetaBASIC will accept each command.

SHELL SORT
The Shell Sort routine located at line number 13000 was originally a type-in from COMPUTE!, December 1982, in an article by C. Regena, Cedar City, Utah.

The routine has been very useful in the many routines which require making order of chaotic character strings. The routine can easily be adapted to numerics or strings. There may be faster sorts, such as branching straight into a machine language subroutine, but this has been a most utilitarian routine in all my projects.

Just follow the comments and be careful to change or map the listed variables to correspond to the section of code requiring the sort, and the routine can serve you well.

SUBMENU
The five-option submenu is included to provide a quick way to subsection code within a given main menu choice. When using this routine, you should remember to have the options return to the beginning of the suboption menu choice, not to line 400 which is the main menu.

Load MetaBASIC into your computer, enable it, and then load Jump Start. You'll then be ready to use the program as a jump start to writing functional code for all your future programs. Just remember; there is no substitute for taking the time to decide what you want to code, before you actually code it.

Gazette, December 1994

TRACKTASK

By Joe Lawrence

Run two 64 programs at the same time and cut and paste between them.

Computers have come a long way in the last few years. One very helpful feature is the ability to run several applications at the same time and be able to copy and paste from one to the other. Now try this same feature on a 64.

With this utility, you can load one program and at the same time load another. You can copy subroutines, data, or whatever or edit one while running the other. You can run a calculator program, for example, and be able to copy the results.

This program for the 64 requires a disk drive and a joystick connected to port 2. Please understand that this is not represented to be totally new software but is simply an application of two previously published utilities that have been slightly modified to perform a useful task.

The Multitasker (November 1986) let you load and run two programs at the same time. The Trackmouse (December 1985) enabled cursor control using the joystick. Tracktask combines the two features and permits copying directly from one program to the other.

The machine language part resides in memory at address

C000 to C3AF.

Load Tracktask.boot to get the program running properly. Type RUN and follow the prompts on the screen.

```
SYS 49936  Press Enter
SYS 49152  Press Enter
Press f1 to initialize memory
SYS 49155  Press Enter
```

The message MULTITASKER ENABLED will appear on the screen.

The function keys have been programmed to the following modes.

f1 Initialize (use at start-up only)
f3 Switch between Blocks
f5 Multitask ON/OFF
f7 Block identify

The joystick (Port 2) will control cursor movement, up/down and right/left. The fire button is now the copy button. To get familiar with the system, press f3 followed by f7 a couple of times. Notice the two totally different screens. Now press the Clr/Home key on both screens. Type the following sample program into one of the Blocks.

```
100 FOR I=1 TO 10000
110 PRINT I;
120 NEXT I
```

Press f3 to switch to the other Block. Now type in just this one line.

```
100 FOR I=10000 TO 1 STEP -1
```

Now move the cursor to just below the 1 in 100. Tap the fire button. This locates the point to which text will be copied. This is indicated by a nonflashing Block under the 1 and the flashing cursor beside it.

Now press f3 to switch Blocks. Move the cursor to the start of the second line. Now holding the fire button down, move the cursor to the end of the line. Notice that the line is highlighted in reverse video.

Now press f3 again. See that the line has been copied over. Press Return to enter the line into program memory. Repeat the procedure to copy the other line.

Type RUN in both Blocks. One program will count up from 1 while the other will count down from 10,000. Use the f3 key to switch back and forth and see each program in action. Each program runs separately.

Pressing F5 enables the multitasking function. When multitasking is on, both programs run at the same time. When multitasking is off, the program run is suspended while viewing the alternate Block. The function key operation was a feature of the original multitasking program.

Gazette, December 1994

LATIN STUDY AID

By Frank Gordon

Use this study guide to help learn about the conjugation of Latin verbs and the declension of nouns.

At one time, Latin very nearly was Western civilization's universal language for scientists and scholars. Newton's "Principia Mathematica" was in Latin, along with many other works on science and mathematics.

Even today, Latin is very much alive in medicine, nursing, and law. For examples, see any law dictionary or textbook on anatomy. Latin is also used in botany for classifying plants.

Each of the above disciplines is replete with Latin or words of Latin origin. An argument could be made for specialized prelaw or premedical Latin courses, but this Latin Study Aid, along with a dictionary and grammar guide, can help you explore Latin on your own.

Latin Study Aid consists of two BASIC programs. The first, Conjugations, lets you choose between the active and passive moods and then shows the principal parts verb forms from which their present, past, and future tenses are formed. The second program, Declensions, shows the declensions of Latin nouns. A menu program on the flip side of this disk lets you run either of these two programs. If you prefer, you can manually load and run whichever program you desire.

CONJUGATIONS

Load and run Conjugations. First select the active or passive mood. Then a menu will appear, giving several model types. Select a desired type, and enter the stem, dropping the "o," "eo," or "io" suffix. Enter MAND for "mando" (I command), for example, PLAC for "placeo" (I please), or MITT for "mitto" (I send). The forms for the present, past, and future tenses will rapidly appear on the screen.

The data statements provide the endings, and in some cases, as in lines 235 and 255, parts of the original stem are replaced. The menu can be added to or changed for other verb types or tenses. Make any required changes in lines 35 and 95-115 to give new values for variable Z, and be sure to properly dimension A$(X). Declensions can be modified similarly.

DECLENSIONS

Declensions also has a menu. Because the noun form and its stem are often different (as with rex, reg and corpus, corpor) you are asked to enter both.

SUGGESTIONS

These programs can serve as starting templates which can be modified for other grammatical forms. With a little work you could modify them for other languages such as Esperanto, which is designed to be very regular in its endings.

Also, the conjugation and declension forms can be altered to conform to your particular textbook.

Gazette, December 1994

MetaBASIC 64

By Kevin Mykytyn

This utility will change the way you program. It adds 32 new debugging and testing commands to Commodore 64 BASIC, working by itself or in conjunction with a machine language monitor/assembler.

An Introduction to MetaBASIC 64

"MetaBASIC 64" commands use English mnemonics, so you don't have to memorize a lot of SYS numbers. And if you forget the new words, you can either refer to this article or type HELP.

BASIC programmers have 12 new commands at their fingertips. For writing programs, AUTO, KEY, and UNNEW are available. You can use CHANGE, DELETE, FIND, RENUM, and VCHANGE to examine and alter programs. And DUMP, SPEED, TRACE, and TROFF help during debugging sessions. If you're writing in machine language, you can use some of the BASIC problem solvers, as well as MEMORY, MONITOR, NUMBER, and @. To control MetaBASIC 64, you have DEFAULT, HELP, INT, and QUIT. Disk commands include BSAVE, CAT, DLIST, ERR, MERGE, READ, RESAVE, SCRATCH, SEND, and START. Finally, there are LLIST if you have a printer and TERMINAL if you have a modem.

Special Notes

Always type NEW after loading MetaBASIC 64. One feature that works automatically is LIST Pause. When you're listing a program, hold down CTRL, SHIFT, or the Commodore key to temporarily halt it. RUN/STOP-RESTORE is available in both program mode and direct mode. But if you want to interrupt any of the utilities like RENUM, use the RUN/STOP key by itself (not RUN/STOP-RESTORE).

The commands work only in direct mode; you cannot add them to programs. Also, you're limited to one MetaBASIC command per line (although you can still use multistatement lines inside your programs). Unlike ordinary BASIC commands, there are no abbreviations. You must type the entire MetaBASIC 64 command. If it seems to be working incorrectly, make sure the syntax is correct.

Machine language (ML) programmers should remember that MetaBASIC 64 occupies memory locations $9000-$9FFF. The 4K that begins at $C000 is available for programs like Micromon as well as for your own ML programs. Be sure to load and run MetaBASIC 64 before loading any other program.

To use MetaBASIC 64, follow these steps:

1. Load the program with LOAD"METABASIC",8,1.

2. Type NEW.

3. Activate the program with SYS 36864.

The program uses 4K at the top of BASIC memory (which leaves you with 35K for your programs). The first thing it does is move the top-of-BASIC pointer down to protect itself from variables. After the SYS, it may seem that nothing has changed. But MetaBASIC 64 is active, and you now have 32 new commands to help you write and debug programs.

MetaBASIC 64 Commands

Here's an alphabetical list of the new commands and how to use them, with examples. In the descriptions of syntax, MetaBASIC 64 commands and mandatory parameters appear in boldface. String parameters appear in italics. Optional parameters appear in normal print.

If something is described as a disk command, it won't work unless you have a disk drive. However, some of the ML programming aids can be useful in BASIC and vice versa.

@
Use: ML programming (see also MEMORY)
Syntax: @ starting address, number, number...

This works like POKE, except it allows you to put a series of numbers into consecutive memory locations. For example, if you want to change border and background colors to white, you use @53280,1,1. The first 1 goes into 53280; the second into 53281. If you add more numbers, separated by commas, they are POKEd into the next locations, 53282, 53283, and so on.

You can also use this in conjunction with MEMORY. First, display the contents of a series of locations using MEMORY. Then change the information there by putting @ before each line you want to change. Move the cursor to the numbers you want to change, change them, then press RETURN.

AUTO
Use: BASIC programming
Syntax: AUTO starting line number, increment

AUTO can take some of the drudgery out of writing a program. It automatically numbers a program, starting at the first number and incrementing by the second. Separate the numbers with a comma. If you do not specify a starting line number or increment, numbering will start at 5 and increment by 5 for each additional line. If you specify only a starting line number, then that value will also be used for the increment. After you press RETURN over a line, the next number is automatically printed. The current line number can be changed by using the INST/DEL (delete) key and replacing it with another number. Press RUN/STOP to escape from AUTO.

Example: AUTO 100,10 starts at 100 and increments by 10.

BSAVE
Use: disk command (see also RESAVE)
Syntax: BSAVE "filename", starting address, ending address + 1

BSAVE (Binary SAVE) saves a chunk of memory to disk, from the starting address to the ending address. Put the program name inside quotation marks and use commas to separate the name, starting address, and ending address. It's important that you add 1 to the actual ending address. You can use this command to make backups of machine language programs, as long as you know the starting and ending addresses. BSAVE can also function to save sections of screen memory, custom character sets, or high-resolution screens.

The numbers should be in decimal. If you need to translate from hexadecimal to decimal, see NUMBER (below).

After you BSAVE the contents of an area of memory to disk, you can load the data back in with LOAD "filename",8,1.

Example: BSAVE"METABASIC 64",36864,40805 makes a backup of MetaBASIC 64. To copy the first five lines of screen memory (locations 1024-1223) to disk, BSAVE "SCREEN",1024,1224. Screen memory does not include color information--that is stored in color memory and has to be handled separately.

CAT
Use: disk command (see also DLIST, READ)
Syntax: CAT

Anytime you want to look at the entire disk directory, use CAT (for CATalog). The BASIC program currently in memory will remain undisturbed. To see specific portions of the directory, see DLIST.

CHANGE
Use: BASIC programming (see also FIND, VCHANGE)
Syntax: CHANGE @old string@new string@, starting line, ending line

CHANGE searches through the program in memory, changing every occurrence of the old string to the new one. The strings can be up to 30 characters long@10,@ and must be bracketed by the commercial at sign (@). All lines in which changes are made are listed to the screen.

The first format will change BASIC keywords and variable names. The second format should be used to change strings. If you omit the line numbers, CHANGE affects the whole program. If you want to change only one section, add the starting and ending line numbers, marked off by commas.

Example: CHANGE @X@QQ@, 1,200 changes the variable X to QQ in lines

1-200. To change the name Charles to John throughout the program, enter CHANGE @"CHARLES"@ "JOHN"@.

DEFAULT
Use: MetaBASIC 64 command (see also INT, QUIT)
Syntax: DEFAULT border color, background color, text color, device number

When you press RUN/STOP-RESTORE, the screen always reverts to the default colors of light blue characters on a dark blue screen. And several commands like LOAD and SAVE default to tape. DEFAULT lets you change these values to whatever you prefer.

If you have a disk drive, you can change the device number to 8. If you have a second drive addressed as device 9 and want to use it for SAVEs, change the default to 9. If your 64 is hooked up to a black-and-white TV or monitor, change the character and background colors to a more readable combination. Note: If you change the default device number to 1 (tape), you will be unable to use any of the new MetaBASIC disk commands. To disable the DEFAULT device number setting and go back to normal, use the MONITOR command below. Also, the TERMINAL command will not operate properly after DEFAULT has been used to change the device number. If you use DEFAULT, be sure to issue a MONITOR command before trying to use the TERMINAL command.

Example: DEFAULT 1,1,0,8 changes the border and background colors to white, the character color to black, and the device number to 8. If you press RUN/STOP-RESTORE, you'll see black characters on a white background. And you'll be able to save to disk by typing just SAVE"filename" (without adding a ,8).

DELETE
Use: BASIC programming
Syntax: DELETE starting line-ending line

DELETE removes a range of lines from your program. Separate the starting line number from the ending number with a dash (-). Use this command with extreme caution because it is impossible to recover deleted program lines.

Example: DELETE 200-250 erases all lines with line numbers in the range 200-250, including lines 200 and 250.

DLIST
Use: disk command (see also CAT, READ)
Syntax: DLIST "filename"

This command lists a BASIC program from disk to the screen, without affecting what's currently in memory. The program name must be enclosed in quotation marks. DLIST enables you to look at a program before using MERGE or SCRATCH.

It also allows you to read portions of the directory. DLIST "$0:A*" displays all disk files beginning with the letter A.

Example: DLIST "BASICPROGRAM" reads the file named BASICPROGRAM from disk and lists it to the screen.

DUMP
Use: BASIC programming
Syntax: DUMP

Use DUMP to examine the current values of all nonarray variables in a program. If the program is running, press RUN/STOP and type DUMP. To resume, type CONT.

ERR
Use: disk command
Syntax: ERR

ERR reads the disk drive error channel and displays the DOS error number and error message from the drive. Use it when the red light on the disk drive starts blinking to determine what caused the problem.

FIND
Use: BASIC programming (see also CHANGE, VCHANGE)
Syntax: FIND @string@, starting line, ending line

This allows you to find any word, variable, or other string within a program. Each line containing the search string is listed to the screen. If you wish to search just one section of the program, add the starting and ending line numbers, separated by commas.

If you're trying to find BASIC keywords (like PRINT or REM), use the first format. It also works for variables and numbers. The second format should be used when you're looking for strings or items inside quotation marks.

Example: FIND @A=@ searches for lines where variable A is defined.

HELP
Use: MetaBASIC 64 command
Syntax: HELP

Whenever you're unsure of the commands available in MetaBASIC 64, type HELP for a complete list.

INT
Use: MetaBASIC 64 command (see also DEFAULT, QUIT)
Syntax: INT

Some features of MetaBASIC 64 are interrupt-driven. If you reset the interrupts (with the MONITOR command), the function keys and the SPEED function may no longer work. INT puts the MetaBASIC interrupts back in place.

KEY
Use: BASIC programming (see also INT)
Syntax: KEY key number, "command or string"

This command adds a lot of flexibility to MetaBASIC 64, allowing you to define each of the eight function keys as a different command or string. (However, because of a minor bug in MetaBASIC, any definition you assign to the f8 key will be garbled whenever you use the RENUM command.) The command, up to ten letters in length for each key, must be inside quotation marks. There are two special characters: The back arrow acts as a carriage return, so you don't have to press RETURN after BASIC commands. Also, the apostrophe (SHIFT-7) counts as a double quotation mark.

Using KEY, you can load other utilities you may own and SYS to them with a tap of a function key. Or you can do a one-key RUN or LIST. If you want to permanently define the function keys and screen/text colors, you can use KEY and DEFAULT to set up the desired configuration and then save a copy of your customized version of MetaBASIC using BSAVE "METABASIC",36864,40960. The definitions will be saved along with the program. If the interrupts are accidentally reset, you'll have to use the INT command to reenable the KEY function.

Examples:
KEY 1,"@CLR@LIST100-" clears the screen and lists line 100 and up whenever you press f1 (the back arrow means RETURN will happen automatically). You can also abbreviate LIST with L SHIFT-I.

KEY 7,"DATA" could be useful with automatic line numbering (see AUTO) if you're writing a program with a lot of DATA statements. After entering a line, press RETURN and you'll see the next line number. Then press f7, and the word DATA automatically appears.

KEY 2,"VERIFY@*' " defines f2 to print VERIFY"*" plus a RETURN (note the apostrophes have been changed to quotation marks). If you've used DEFAULT to change the device number to 8, pressing f2 will automatically verify the program most recently saved to disk.

LLIST
Use: printer command
Syntax: LLIST starting line-ending line

This command lists a program, but the listing is sent to a printer rather than to the screen. Line numbers are optional. The syntax for LLIST is identical to the regular LIST. As written, LLIST does the equivalent of OPEN 4,4,4 to open a file for output to the printer. Some printers may require a different secondary address (the last number in the OPEN statement), OPEN 4,4,7, for example. To change the secondary address, POKE the desired value into location 40341. If you are using a printer with a different device number (5, for example) or a plotter (device 6), you can change the device number for LLIST by POKEing the desired value into location 40339. To make the changes permanent, follow the instructions for saving a new copy of MetaBASIC given above in the discussion of the KEY command.

Example: Use LLIST 10-20 to list lines 10-20 to the printer.

MEMORY
Use: ML programming (see also @)
Syntax: MEMORY starting address-ending address

You can examine any section of memory with this command. Use decimal numbers (not hex) for the starting and ending addresses. The values in memory are displayed, six bytes per line, in decimal. In addition, the equivalent ASCII characters are printed in reverse to the right (if there's no corresponding ASCII character, a period is printed).

If you omit the ending address, MEMORY 43 for example, you'll see the contents of two bytes (43 and 44). This makes it easier to look at two-byte pointers, like 43 and 44, which point to the beginning of BASIC memory. To change memory, you can use the @ command, described above.

Example: Enter MEMORY 41374-41474 and you'll see the first few error messages in BASIC ROM (note that the ASCII value of the last character is always added to 128). Or, load a BASIC program, and type MEMORY 2048-2148 to see how programs are stored in memory.

MERGE
Use: disk command
Syntax: MERGE "program name"

MERGE reads a program from disk, lists each line to the screen, and adds the line to the program in memory. If the programs have common line numbers, the program on disk takes precedence. Say both programs contain a line 250. The line 250 from the disk program will replace line 250 in memory.

Before using this command, you may want to use DLIST to make sure you're merging the right program. And if there are conflicting line numbers, you can use RENUM to renumber one of the two programs. If you want to merge just part of one program, use DELETE to eliminate the unwanted lines.

MONITOR
Use: ML programming (see also INT)
Syntax: MONITOR

If you have a machine language monitor in memory, you can enter it with MONITOR (providing that it is enabled when a BRK instruction is executed). To use MetaBASIC 64 with a monitor, you must load MetaBASIC 64, type NEW, and activate the program with SYS 36864. Next, load the monitor, type NEW, and SYS to the starting address for the monitor (which will set up the BRK vector to point to the monitor).

MONITOR does several other things, as well. It changes border, background, and text colors back to their default values (light blue on dark blue). It also resets the default device number and sets interrupts to normal, which disables the function-key definitions (see KEY) and SPEED command. You can get them back with the INT command.

NUMBER
Use: ML programming
Syntax: NUMBER $hexadecimal number

NUMBER allows you to convert back and forth between decimal and hexadecimal (hex). Put a dollar sign ($) in front of hex numbers. In addition, the number is displayed in low-byte/high-byte format (in decimal) and in binary (preceded by a percent sign).

```
Examples: NUMBER $100
  256
  0 1
  %100000000

  NUMBER 34
  $22
  34 0
  %100010
```

QUIT
Use: MetaBASIC 64 command
Syntax: QUIT

This resets all vectors and disables all MetaBASIC 64 commands. The one thing it does not do is restore the top-of-memory pointer. MetaBASIC 64 is still protected from BASIC. Reactivate MetaBASIC with SYS 36864.

READ
Use: disk command (see also CAT, DLIST)
Syntax: READ "sequential filename"

READ allows you to examine sequential disk files. The information in the file is displayed to the screen, without altering whatever program is in memory.

In the rare case that you want to use the BASIC READ statement in direct mode (to see if all DATA statements have been read, for example), you can precede it with a colon.

RENUM
Use: BASIC programming
Syntax: RENUM starting line, increment

This command renumbers the entire BASIC program in memory (you can't renumber just part of the program). The first line of the renumbered program will be given the specified starting line number. If you omit the starting line number, the renumbered program will begin with line 10. The increment value specifies how much the starting value will be incremented for each succeeding line. If no increment value is provided, the value defaults to 10.

In addition to renumbering BASIC lines, all references in GOTOs, GOSUBs, ON-GOTOs, ON-GOSUBs, IF-THENs, and so forth are taken care of. One word of caution: GOTO is covered, but GO TO (with a space in the middle) is not. Use FIND before renumbering to look for occurrences of GO TO.

Example: RENUM 100,20 renumbers a program, starting at line 100, counting up by 20s.

RESAVE
Use: disk command (see also BSAVE)
Syntax: RESAVE "filename"

The built in save-with-replace disk command (SAVE "@:filename") first saves the program and then scratches the older version, so there must always be enough free space on the disk for the new version of the program. This can cause problems if you don't have enough available space. The save-with-replace command is also sometimes unreliable and should be avoided.

RESAVE reverses the order; first it scratches the old version of your program from disk, and then it does a regular SAVE, solving both of the above problems.

SCRATCH
Use: disk command
Syntax: SCRATCH "filename"

SCRATCH does the same thing as OPEN 15,8,15: PRINT#15,"S0:filename": CLOSE 15, but it's easier to type. It scratches a file from the disk. If you have just inserted the disk into the drive, it's a good idea to initialize it first (see SEND). You can use wildcards to scratch more than one program; for example: SCRATCH "A*" will get rid of all files beginning with the letter A. However, you should use such commands with care to avoid accidentally deleting important programs.

Example: SCRATCH "SPACEGAME" removes the program named SPACEGAME from the disk.

SEND
Use: disk command
Syntax: SEND "command string"

This is a convenient way to send disk commands to channel 15. SEND"I0" initializes the drive, SEND"V0" validates the disk, SEND "R0:newname=oldname" renames a disk file, and so on. For more information about disk commands, see the 1541 user's manual.

SPEED
Use: BASIC programming
Syntax: SPEED number

SPEED changes the rate at which the 64 prints to the screen. The number supplied with the command must be in the range 0-255. The higher the number, the slower the printing speed. Try typing SPEED 255 (the slowest you can make it) and then list a program. You can get back to normal with SPEED 0. If it doesn't work, try using INT (see above) to correct the interrupts. SPEED is useful when you're using the TRACE command.

START
Use: disk command
Syntax: START "filename"

If you forget where a machine language program begins, put the disk in the drive and use this command. This can help when you have forgotten the SYS that starts a program. If this command returns the value 2049, the file you are checking is probably BASIC rather than machine language (or it at least has a single line of BASIC, as does "SpeedScript").

Example: START "METABASIC 64" should display 36864 on the screen.

TERMINAL
Use: modem command
Syntax: TERMINAL

If you own a Commodore modem (and it's plugged into your 64), TERMINAL transforms your computer into a 300-baud "dumb" terminal you can use to talk to standard-ASCII bulletin boards or information services like CompuServe. You can't change any of the default parameters, nor can you upload or download text or programs.

To return to BASIC, press the £ (English pound) key; do not press RUN/STOP RESTORE. A note of caution: Memory locations 52736-53247 ($CE00-$CFFF) are used for buffers, so any program in this area will be overwritten.

TRACE
Use: BASIC programming (see also TROFF)
Syntax: TRACE

If you're debugging a BASIC program, TRACE helps you see what's happening. As each line is executed, its line number is printed on the screen. Use the SHIFT or CTRL keys to temporarily halt the program. SPEED controls the speed of execution, and TROFF turns off TRACE.

TROFF
Use: BASIC programming (see also TRACE)
Syntax: TROFF

This command turns off the TRACE function.

UNNEW
Use: BASIC programming
Syntax: UNNEW

You may never need this command, but it's nice to have it available. If you accidentally type NEW and you want to retrieve the program, use UNNEW to get it back.

VCHANGE
Use: BASIC programming (see also CHANGE, FIND)
Syntax: VCHANGE @old string@new string@, starting line, ending line

VCHANGE (Verify CHANGE) works just like CHANGE, except you get to choose whether or not each change is made. Each line containing the old string is displayed, with each occurrence of the string marked with a filled-in circle. If you press Y, the change is made. Press N to skip to the next occurrence of the old string.

Gazette, December 1994
e11e11e11e11e11e11ere11e11e11e11e11ere11e11a11erre11eZerre11e@11@11111@11@11@11
@111111111@11@11@11@11@11@

Holiday Greetings

As this year draws to a close, Gazette Disk would like to wish you a happy holiday and a properous new year.

To help celebrate, we thought we'd include this program that shows off the graphic and music capabilities of the 64. This demo is from the very early days of the 64. You can get some idea of how old this program is by the promotional material on the final screen and the price that is presented. We think you'll like it.

Best wishes and see you next year.

—Tom Netsel, Gazette editor

64/128 VIEW: On the Internet

By Tom Netsel

By this time QuantumLink should have turn off its modems, ending a great era of Commodore telecommunications. If you ever logged on then you know what a fantastic resource it was for 64 and 128 users.

Of course, over the past few years GEnie has attracted many Commodore users to its service. It has a large number of programs and files for downloading. That library combined with chat areas, and commodore-specific message bases provide a good alternative to Q-Link. But there's still another alternative that I think you'll find interesting. That's the Internet

Unless you've been marooned on an island without TV, newspaper, or radio, you've undoubtedly heard about the Internet, but experiencing it may be another matter. Until fairly recently this enormous collection of computers hooked together across the world was accessed primarily by businesses and universities. But that is changing rapidly, and now many individuals are logging on--even Commodore users.

This collection of computers is indeed worldwide, and the items of interest you can find on it are just as widespread as are its users. The whole network dwarfs any single commercial service, but this variety is one of the network's drawbacks. It is so huge and so varied that it can at first be intimidating. There is no user-friendly graphical interface to guide you through it's passages such as we had on Q-Link, and finding files and other items of interest can be a challenge, but it's well worth checking out.

Of course since the network is a loose collections of computers scattered around the world, there is no central headquarters, no 800 number to call. You have to locate an access in your own community, often through a university, business, or commercial access provider. Here in Greensboro, North Carolina, for example, our local newspaper has started offering access for about $10 per month. Online services such as CompuServe and AOL also offer access.

In any event, I think you'll find the feature article on this disk interesting. Nick Rossi, the man behind Novaterm and no slouch at Commodore telecommunications, has written an outstanding introducing the Internet with the Commodore user in mind. He covers many of the topics that you'll want and need to know before you tap into this worldwide net.

Since the article is so long, I have broken it into two parts, both on this disk. So be sure to check out parts one and two of "Wax Up Your Commodore; We're Going Internet Surfing!"

After you get a taste of what available and how to find items, you are bound to have many more questions. There have been a number of books written about the Internet, but one interesting one arrived at the office just last week. This one is called Simple Internet by Jeffrey M. Cogswell. Cogswell got his start programming on a PET and VIC-20.

Simple Internet is a beginner's guide that is written from the viewpoint of Archie Finger, an Internet illiterate hired to solve the mystery of a missing person online. You travel the information superhighway with Detective Finger as he uncovers electronic clues that prompt him through the various net tools and features. The paperback book is disguised as a humorous detective-novel parody, so it's fun to read as well.

This 175-page guide should be in the computer section of most bookstores by now, or you can order it for $16.95 by calling Waite Group Press at (800) 368-9369.

By the way, as several of you already know, you can reach me via the Internet here at Gazette headquarters by sending e-mail to TomNetsel@aol.com. That's just one of the many features of the Internet that is so appealing. Once on the net, you can send e-mail to sites and individuals practically anywhere in the world for the cost of a local telephone call.

Gazette, December 1994

FEEDBACK

USING AN FD-2000 WITH 1581 PROGRAMS

One of the many additions I have made to my Commodore 64 system over the past 10 years has been an FD-2000 3.5-inch drive from Creative Micro Designs. The drive is mainly used with GEOS and Big Blue Reader for easy access to Laser Printing from GeoPublish.

Recently, I planned to update my Gazette Index disk using my FD-2000 for a one-disk index instead of three 1541 disks. I soon realized that it would be much easier to use if the files were organized in a logical manner.

The FD Utilities Disk that came with the drive has a good selection of files but it is missing a Directory organizer program. I tried all the Organizing programs I had and couldn't get any to work. Eventually I used Dirmaster (PD Picks in February 1994 Gazette) under GEOS to sort the files into the desired order. This worked but I don't want to boot GEOS to organize non-GEOS disks on my FD-2000.

I had tried 1581 Alphabetizer (February 1989 Gazette) and 1581 Sorter (July 1989 Gazette) only to be told the drive wasn't a 1581. Checking the listings (both are written in BASIC) I saw where the drive type was checked and it was easy to fix.

What the programs look for are the version numbers of the drives' DOS, and if the last two numbers aren't 81 the programs wouldn't work. To find the version number of your FD drive DOS use the @ command on your JiffyDOS-equipped computer right after turning your system on and you should see something like the following.

```
73,CMD FD DOS V1.34,00,00.
```

My DOS is version 1.34, so I added a check for the last two digits 34. Here are the original lines from 1581 Alphabetizer and my revised lines.

ORIGINAL

```
60 OPEN15,DV,15,"UI":INPUT#15,A$,B$,C$, D$:CLOSE
15:IFRIGHT$(B$,2)="81"THEN110
70 PRINTTAB(9)"↓DEVICE"DV"ISN'T A 1581"
```

REVISED

```
60 OPEN15,DV,15,"UI":INPUT#15,A$,B$,C$, $:CLOSE
15:IFRIGHT$(B$,2)="81"THEN110
65 IFRIGHT$(B$,2)="34"THEN110
70 PRINTTAB(9)"↓DEVICE"DV"ISN'T A 1581 OR FD DRIVE"
```

The same sort of easy fix can be made to 1581Sorter as the original and revised lines show here.

ORIGINAL

```
170 OPEN15,DV,15,"UI":INPUT#15,A$,B$, C$,D$:CLOSE15
180 IFRIGHT$(B$,2)<>"81"THENPRINTTAB (WD-11)"┤ ┐DEVICE"DV"┘ ISN'T AN
81":GOTO100
```

REVISED

```
170 OPEN15,DV,15,"UI":INPUT#15,A$,B$, C$,D$:CLOSE15
180 IFRIGHT$(B$,2)="81"THEN190
182 IFRIGHT$(B$,2)="34"THEN190
185 PRINTTAB(WD-11)"┤ ┐DEVICE"DV"┘ ISN'T AN '81 OR FD Drive":GOTO100
```

If you have the ROM upgrade from CMD (from 1.28 to 1.34) this should work fine. If these modifications don't work you may have a different ROM version (I'm not sure about the ROM version of an FD-4000). Check your drive manual on how to read the error channel or use the JiffyDOS method described above.

You could also add this line to the Alphabetizer and Sorter programs.

Alphabetizer
```
62 PRINTRIGHT$(B$,2):END
```

Sorter
```
175 PRINTRIGHT$(B$,2):END
```

Run the program and note the number that appears on the side of your screen above READY. These are the two numbers to use instead of 34 in lines 65 and 182 of the revisions. Delete line 62 or 175 as required and save your new copy of the program.

With these programs modified they work just fine on disks formatted in CMD format or on 1581 sized partitions. As always, though, when using a program that writes to your disk, especially re-writing the directory, try this on a backup. Do not expereiment on a disk that is full of irreplaceable material.
I have saved the new versions of these programs onto the FD-Tools disk that came with my drive so they are handy when I need them.
BRUCE THOMAS
EDMONTON, AB
CANADA

USER GROUP UPDATE
Here are some Commodore user groups that have recently changed addresses.

Hudson Valley Commodore Club
c/o G.T. Gallagher
620 Plainfield St.

Kingston, NY 12401

Ottawa Home Computer Club
c/o Zembrund Colynuck
1743 Camiti Crescent
Orleans, ON
Canada K4A 1M2

BACK ISSUES
During the mid 1980s, you had a graphics program called Construction Set. Can you tell me where I can purchase a copy of this program on disk for my 64?
G. GRAHAM
BRONX, NY

Construction Set by Fred Kang was published in the December 1985 issue of Gazette. Back issues of the Gazette Disk prior to June 1990 disk sell for $3.50. Issus from September 1990 to the present sell for $11.95. Those prices include shipping.

They are available by writing to Gazette Customer Service, COMPUTE Publications, 324 West Wendover Avenue, Suite 200, Greensboro, North Carolina 27408.

OTHER BASIC
I attempting to convert some simple Commodore programs for use on my IBM, I've come across a printing command called LPRINT. Is there an equivalent command in Commodore BASIC?
BRAD SMALLEY
JESSUP, NC

Some versions of BASIC include the statement PRINT for printing text to the screen and the statement LPRINT for sending text to a printer. Printers were at one time called line printers so this may account for the term LPRINT.

Commodore does not use the LPRINT term, but it can make programming easier when QBASIC or one of the other dialects that support its use.

For example, you could use the following lines in QBASIC.

```
10 PRINT "THIS LINE PRINTS ON THE SCREEN.
20 LPRINT "THIS LINE PRINTS ON THE PRINTER.
```

On the Commodore, you first have to open a channel to the printer and then use PRINT# to send that line to the printer. Here's an example.

```
10 OPEN 1,4,7
10 PRINT "THIS LINE PRINTS ON THE SCREEN.
20 PRINT#1, "THIS LINE PRINTS ON THE PRINTER.
30 PRINT#1: CLOSE1
```

The three numbers after the OPEN statement are the logical file number, the device number, and the secondary address. The logical file number can be any number up to 127, and the device number for printers is usually number four. The secondary address of seven means that the letters should appear in uppercase and lowercase. You then have to close the file with line 30. Naturally you could delete lines 10 and 30 when converting.

LPRINT may seem easier to use, since you don't have to open files and specify addresses, but PRINT# has some advantages. With it you can specify which peripheral gets the information. In addition to printers, you can write files to tape, disk, or modem.

LISTING TRICK
Some years back you wrote about a little trick to foil people from listing a BASIC program and it also printed a warning message. I think it had to do something with a REM statement, and it printed a message to the screen. I can't seem to locate it, so would you mind printing it again?
BERT WIGMORE
ORLANDO, FL

If you put a REM statement on a line followed by a Shift-L, that will prevent the program from listing beyond that line, but it won't let you print a message. Try this one and see if it is what you want.

Enter a low line number followed by REM. Press Shift-2 twice and delete the second quotation mark. Now press Ctrl-9 to turn on reverse mode, followed by a Shift-M and Shift-S. You should see on your monitor a reverse backslash and a reverse heart. You can now turn off reverse mode by pressing Ctrl-0. enter the message that you want to appear onscreen.

On the next line, enter REM Shift-L. Now whenever anyone tries to list your program, the screen will clear, and your message will appear.

Gazette, December 1994

MULTIMEDIA ON A SHOESTRING

By Fred D'Ignazio

I travel around the U. S. training teachers and students how to use so-called "state of the past" equipment commonly found around most schools to create a multimedia publishing center for students. Among the types of equipment are the following.

* Record players
* Tape recorders
* VCRs
* TVs
* Musical keyboards
* Pocket cameras
* Music stands (used as a copy stand for videotaping and photography)

In weekend workshops, while we are sitting on the floor in the gymnasium or library in our jeans, we assemble these little consumer-electronic "leftovers" into a RAP station (for Research, Authoring, and Publishing). We put the station on a small wheeled cart, and the teachers have a portable publishing studio which can be wheeled around a classroom or up and down the halls of a school building.

Before I arrive on the scene, I send the teachers a Scavenger's List. Teachers select a team of students, and they begin a scavenger hunt, snooping around the school's cobwebbed closets and dusty cupboards trying to turn up the items on the list. When the hunt is finished, they drag everything into a gym or a media center.

On the morning of the workshop there are heaps and clumps of stuff lying around the floor. The magic of the workshop is the transformation process by which the humble clumps of stuff become creation stations and multimedia centers for student authors.

AT THE CORE: THE COMPUTER

At the core of our multimedia center is a computer. Not just any computer, of course. This computer has to be one of the oldest, most neglected, and antiquated relics you can imagine. Just in the last year I have taught multimedia scavenging workshops to teachers who have created a multimedia center using the following vintage machines.

* Apple II+
* IBM PCjr
* Atari 400
* TI 99/2A
* Radio Shack TRS-80
* Radio Shack Color Computer
* Commodore PET
* Commodore 64

What is this? How can these old computers become multimedia machines?

According to the manufacturers, multimedia PCs must have at least 8 megabytes of RAM, a 200-megabyte hard disk drive, a Motorola 040 or PowerPC processor (or a Pentium or 486). They must feature Super VGA resolution, have an on-board sound card, and come equipped with QuickTime (Macintosh) or Microsoft Video for Windows.

Wrong! If you do multimedia the wrong way, even these machines aren't powerful enough.

I've taught workshops on PowerPCs, Pentium-based PCs, and Unix machines on a 100-megabit per second network, and multimedia can turn these sleek Ferraris of the computing world into sluggard sloths and snails. The gargantuan sound files, high-resolution color images, and multimegabyte movie files can slow a speed freak's dream machine to a crawl.

The Achilles Heel of multimedia is digitizing. Digitizing means converting all other media into computer format. Images, sounds, pictures, movies, and so on can all be digitized and saved as files on a computer's hard disk. But digitized media sucks up most of your machine's processing power and greedily gobbles up much of its computer memory.

The trick is to create multimedia without digitizing. What you have to do is turn digitizing upside down and instead learn how to "undigitize."

UNDIGITIZING

Enter the antique computers. Many of these computers have a lovely almost unknown feature which most of today's modern multimedia computers lack: a video jack. This little hole or port on the back of computers such as the VIC, the 64, and the Apple II makes it possible to connect the computer to a VCR or (using an RF modulator) directly to your TV. On the other hand, most newer computers (Macs and PCs) have only an RGB jack on the back. This means a computer user must purchase an RGB-to-video scan converter box for $200 to $400 to transfer a computer image from a tiny computer screen to a large-screen monitor.

The first thing I do when I show up at a scavengers workshop is to get the teachers to plug their old computers into the video-in jack of their VCRs. One end of a standard AV cable ($1.98 from Radio Shack) goes into the monitor jack on the back of the computer. The other end goes into the video-in jack on the VCR. The VCRs are usually already cabled to the TV, so a few minutes into the workshop we are copying slick computer-generated titles and graphics from a colorful paint program running on the older computer directly onto a blank videotape running on the VCR.

To add sound, sound effects, and narration we take a second AV cable, plug one end into a humble little audio tape recorder and the other end into the audio-in jack of the VCR. The voice or music from the audio recorder merges with the graphic titles and pictures from the computer, and, voila!, you have an electronic slide show that can make students' ho-hum history reports and book reports spring to life.

We are so successful turning our primitive PCs into multimedia stations because we refuse to digitize---out and out refuse! Digitizing is a surefire way to give your computer instant constipation. So we don't do it. We run the video out of the computer onto videotape and not back in. We completely bypass the computer with the audio by going directly from the audio tape recorder to the audio-in jack of the VCR.

This method works because even with all the hype about the digital desktop, the desktop studio, and desktop media, most people have to take all that media that they painstakingly digitized and then undigitize it in order to publish it. Publishing media on a computer is impractical because there are no universal "media players" to play your stuff on someone else's computer as easily as popping a tape cassette into a friend's Walkman or boom box.

And what if you don't want to publish just still graphics and computer animations on your videotape? What if you want to do movies? Simple. When you get to the point where you want to insert a movie, press the Pause-Record button on your VCR, take the video cable out of video-in jack on the back of the VCR, take the audio cable out of the audio-in jack on the back of the VCR, and replace them with the audio and video cables from your camcorder. Turn on your camcorder until you see what you want to record through the viewfinder, then un-pause the VCR. You have just begun splicing live video into your videotape publication.

If your friends who are multimedia geeks sneer at you because you aren't digitizing movies directly into the computer, tell them you have higher production standards than they do. They are reducing the quality of their final videos by bringing them into the computer at a capture rate of from 10 to 20 frames per second (normal NTSC video is approximately 30 frames per second). Their color palette isn't millions of colors as in real TV but is probably a wimpy 256 colors. And their movies are usually small postage-stamp sized boxes because few computers can create and play back full-screen digitized movies. So why detour movies into a computer to lower their quality just so you can reroute them back out of the computer into your VCR? Doesn't it make more sense to just send the movies directly onto a recording VCR so you can maintain their high quality?

MOLDY MOPEDS ON THE INFO SUPERHIGHWAY

In our multimedia-publishing workshops we are just beginning to hook our computers to modems so we can travel by phone lines out onto the information superhighway. Our idea is to use computer networks to connect kids to other kids, to do online research, collaborative authoring, and global publishing---all on our modest little computers.

This is a tall order! But we think it can be done. The key is to find people who are using old Commodore machines as surfboards on the Internet ocean and mopeds on the Internet superhighway.

Can you help us? What sort of software do you use? Do you just exchange text files, do chats, and send E-mail, or can you upload and download multimedia files? What should we do? How do we get started?

I can hear the crash of waves and the roar of 26-wheel behemoths rolling down the highway. We're at the entrance ramp ready to go.

Is there anybody there? I believe that you exist. I feel you're out there somewhere. If you see this column, please contact me at America Online (explorer00@aol.com) or the Internet (dignazio@msen.com). I would like to publish your stories and your tips in an upcoming Gazette column. (For more on the Internet, see Nick Rossi's feature story in this issue.)

Gazette, December 1994

BEGINNER BASIC: Tracking Holiday Costs

By Larry Cotton

Season's Greetings! Back in the July issue of Windows magazine (written for those other computers), Enis Moran asked readers whether they had ever been frustrated by the fact that some computer programs (called applications or just apps) lack some specific feature that would make life easier.

He cited examples such as expense reports, resolving a complicated monthly account, and contract information. He then encouraged his readers to write a custom program precisely designed for the job that needed to be done.

Deja vu! Looks like those other computers have the same problem we 64 and 128 owners occasionally have: Often, sophisticated apps are overkill. We just need to sharpen our BASIC skills and write our own short programs to do only the small tasks immediately at hand. Last month we learned how to adapt a few time-worn formulas from the math and financial books to do just that. Let's wind up this topic with one more illustration.

Let's say you're trying frantically (and probably unsuccessfully) to keep up with the money you're spending on Christmas gifts. You could go to the trouble of loading a spreadsheet, assigning column widths, stating that certain columns contain string (literal) values and that others contain numeric (dollar) values. But it's just as easy, and a lot more fun, to write a short BASIC program tailored to track your mounting holiday expenses and to maintain a permanent record on disk. And it's a program that can be used year after year.

As an example, Listmaker, on the flip side of this disk, lets you quickly create a list of Christmas gifts and their costs and saves the list to disk. Although it's loaded with REMs, here are some highlights.

The program contains almost no error-checking. This is something you can get away with if you're writing a simple program primarily for your own use. I would not, however, recommend skipping error-checking when writing BASIC programs for others to use.

Line 20 illustrates the use of variable arrays, which we've covered in the past. Suffice it to say we use two of them: the names of the Christmas presents (camera, wallet, whatever) and what they cost. They are I$() and A(), respectively. Line 30 assigns a name (XMASLIST) to the list to speed up disk activity. There's no provision for giving the list a name once the program runs.

You do have an opportunity to load a previously saved list (lines 40-60 and the subroutine at 170-230).

Lines 80-160 make a loop that collects the item names and costs. C is the item counter which increments for every pass through the loop. Line 150 introduces variable T which is the "bottom line"--what you've spent on gifts so far. It starts life as 0 and is increased by the item costs A(). Everything else in this loop should be self-explanatory.

Lines 170-230 and 310-360 are essentially parallel subroutines, which respectively load and save the Christmas list. They use a simple technique that opens a file on disk (arbitrarily numbered 1) and then either loads data from, or saves data to, that file. The disk drive knows which to do by the third number in the OPEN statement: 0 means load, 1 means save.

Just before a disk save, the old file must be scratched (erased) if it exists. Line 300 takes care of that. Incidentally, the file XMASLIST will look like any BASIC program and will have a PRG extension.

In both subroutines, a FOR-NEXT loop is used to either read (load) or write (save) the arrays on disk. The advantages of array variables should be obvious. As X (the index to the arrays) increases, the variables I$() and A() are copied from computer RAM to the disk or vice versa. The two other nonarray variables (item count C and the total dollars spent so far, T) are loaded or saved before the FOR-NEXT loops begin.

After a disk load, the list is printed to the screen in lines 240-270, followed by the total spent so far. Incidentally, when entering item names, try to limit the number of characters per item to eight so the cost column will be printed neatly. As your list grows to more than one screen, you can hold down the Ctrl key to slow the printing.

Since Listmaker writes to and reads from a disk, you won't be able to use it with Gazette Disk, which is write protected. Simply load Listmaker manually from the flip side of this disk, save it to a work disk, and then run it.

Why not jump in and modify Listmaker to your own needs? You may want to add another column such as whom the gift is for. Or you may want to add a quickie routine to send the list to your new ink-jet printer that Santa brought a few days early. Enjoy the holidays!

Gazette, December 1994

MACHINE LANGUAGE: Simple String Sort

By Jim Butterfield

Sorting is an important operation when it comes to processing data. Sorting strings is generally more important than sorting numeric values, but sorting in BASIC is slow and has certain pitfalls. In this session, we'll use machine language to help with the speed problem.

First, we have two subjects to discuss. One is about sorting itself, a much misunderstood subject. Then we'll have to take up the problem of how BASIC and machine language can communicate string information usefully--not as easy as you might think. And we'll use a gimmick called fixed-length strings to help sidestep some of these intercommunication problems.

SORTING SUBTLETIES

Most sort demonstration programs perform their work on a set of single-field items, such as sorting a bunch of numbers or a list of names. Unfortunately, very little useful data comes in this form. Real-world data comes with multiple fields, where each record contains several items, such as first name, last name, street address, city, and so on.

When you sort this kind of data, you must choose a key field to be sorted (city, for example). That's where you do the comparison. But when you move the sorted record to a new position, you must move the whole record, not just the key field.

That's not an easy job for BASIC. And when you use machine language, the task of keeping track of all those fields seems insurmountable.

LINKING STRING DATA

BASIC strings are stored rather haphazardly in a memory pool sometimes called the "heap." BASIC keeps track of each string by means of a three-byte "descriptor." The first byte of this descriptor tells the length of the string; the second and third bytes give the location where the string is stored.

As BASIC creates and destroys strings, the memory heap becomes messy, and there may eventually be a need for "garbage collection" to take place. This event is dreaded by 64 users, since the computer freezes for considerable time (sometimes minutes).

To avoid this process, the rule in BASIC is to leave strings alone if you can. Don't change them, copy them, or edit them.

In machine language, the rules are even more strict. Your program can get into trouble if you create strings without understanding garbage collection. And the 128 has a completely different garbage collection scheme than that of the 64. The 128 uses "back pointers" which can crash your system if they're mishandled.

A SOLUTION

One way to get around the problems outlined above is to use an "index" array which points to the strings in question. Your program moves the index numbers, not the strings. This numeric index value neatly sidesteps the complexities of playing with strings.

Our approach this time will be different, however. It will take us back to a technique used in the earliest days of data processing: fixed length records.

Chances are you have seen punched cards, once very popular for data. The Hollerith card was most common, with 80 characters of information. Data was arranged by columns: "First name" might start in column 1, "last name" could start in column 15, and so on.

We'll use a similar scheme for our database. Each record will be a string of exactly the same length; the various fields will start in fixed positions.

This scheme of fixed-length records lets us swap two strings in memory without worrying about complexities. After all, if they are the same size, the data strings will exchange neatly. This is true of both 64 and 128 strings, but keep in mind that all 128 data is kept in "Bank 1."

The sample data file, FALLS, contains information on selected waterfalls around the world. Each record is 30 columns long: The name of the fall starts in column 1, its country starts in column 15, and its height occupies columns 27 to 30.

You can make up your own data files, of course, using your own arrangement of fields. Be sure that each record is exactly the same length. Maximum record size is 255 characters.

THE PROGRAM

The program SORT64/SORT128 reads in a set of records from the file FALLS, and puts them into an array. Then it asks you which field you would like to sort.

Based on your choice, the program pokes two values into memory. At address 40, it pokes the first column of the selected field, minus 1. At address 41, it pokes the last column of the field. Thus, to sort file FALLS by country, it would poke values 14 and 26.

Then, a SYS command calls the brief machine language sorting program, located in the cassette buffer. When BASIC resumes, the array has been sorted as requested.

The results are printed on the screen, but you are free to modify this BASIC program to write to the printer or to a disk file. The only rules are that the array must be dimensioned to a size at least one bigger than the number of records, the array must be the first one (if you have others), and the records must all be the same length.

MACHINE LANGUAGE

The sorting method used is the "bubble sort." That's one of the slower methods, but I think you'll find it plenty fast for the job. You're welcome to try your hand at coding fancier sorting methods; the fixed record size will help no matter what method you use.

Here's how the bubble sort works. The program scans through the strings, looking for adjacent pairs that are out of order. If it finds any, it swaps them and keeps going. It's worth noting that the program spots the last item of data by length. When the string length is wrong, the scan stops.

After a scan, the program checks an activity flag. If nothing happened during that last scan, the data must be sorted, and the program exits. Otherwise, back it goes to repeat the process.

Three pointers are used in zero page. DSCR walks through the array's string descriptors, starting at item 0 and bumping along three bytes at a time. From the descriptors we extract PTR1 and PTR2, pointers to the two strings to be compared.

You'll find the comparison and swapping routines to be quite straightforward. The source code is well commented.

128 BONUS

Sort128A is a straight rewrite of Sort64. The program is sited in the 128 cassette buffer. A few zero page locations needed to be relocated. There is one main problem: All the data (descriptors and strings) are located in Bank 1, but our program is in Bank 0. Data must be fetched, stored, or compared using Kernal functions INDFET, INDSTA, and INDCMP, which can reach across to another bank. The program does a lot of this kind of work, of course. Thus, it runs much more slowly than the 64 version. Even kicking the 128 into FAST mode doesn't recover the lost time, although the program is still much faster than BASIC.

Sort128B takes another approach. If the data is in Bank 1 and crossing banks costs time, why not put the ML program into Bank 1? No problem, except for a small job: We must make room by shifting the start-of-variables pointer. Suddenly, our program recaptures its original 64 speed. And, of course, FAST mode can make it really sing.

To avoid disk clutter, I have not included source code on disk for the SORT128 programs. But the built-in machine language monitor on the 128 makes it easy to disassemble the code, and the program logic closely follows that of Sort64.

SUMMARY

When messing with BASIC strings, be aware of the dangers involved. It's nice to see an easy way to sidestep the pitfalls, as we do here.

PROGRAMMER'S PAGE: Winter Shopping

By David Pankhurst

This marks my first column since moving. As many of you may know, I was a resident of Montreal, Quebec, which is located due north of New York. Now, however, I've moved across Canada and am a new native of Winnipeg, Manitoba. Besides having a reputation for being windy and cold, Winnipeg holds the distinction of being almost exactly on the east-west geographical center of North America. For these and many more reasons, I am delighted to have made the move and look forward to residing here a good long time.

Winnipeg's (and Canada's as a whole) being noted for cold weather brings home to me the significance of this column's date. As I type this column in September, it isn't cold--but it will be. And when you read this, winter weather will be in full force.

Another thing in full force will be Christmas. Arguably the most popular celebration in existence, it's even popular with nonbelievers--and especially merchants. Even if you don't celebrate Christmas, this is a time when bonuses and extra spending cash come in and sales pop up, so I think it's safe to say that gift-giving is well neigh universal at this time of year. So what better way to spend time than curled up in front of a nice warm 64, reading a column about buying stuff? This month, we'll review gift suggestions, as well as where and how to get those deals. Besides prodding the gray cells into new areas of exploration, it'll also serve as a reminder of all we Commodore users have to play with. You might want to print out this column and highlight some of the gift suggestions that you'd like to receive. Then make sure a spouse or other family member "accidentally stumbles over it."

WHERE TO FIND IT
Although Commodore items may be sparse at the local department store, chances are excellent that you can latch on to something by way of the mail-order route. There are a great many companies out there ready to provide that special gadget, and they're just a phone call away. The ads on this disk will clue you into who carries what, as well as provide ordering information. If you're new to this and nervous, here are a few tips to help you ease into ordering:

* Start with a company you've seen advertising awhile and a product you're familiar with.

* If you feel uncomfortable giving your credit card information over the phone, pay with a money order or check.

* Be patient.

The last suggestion comes from personal experience. The one thing I've learned about packages is that they always show up the day after I have a major seizure because they haven't arrived yet. A little patience will preserve the blood pressure and your sanity.

Local stores offer another route to finding gifts, although there are fewer places to search for that special item. If you happen across a Commodore product, chances are the people in the store will be happy to sell it at a discount. From personal experience, I can vouch that this is an excellent way to build up your collection of goodies. Prices are lower than for other makes of software because of the prevailing attitude that "If it ain't IBM, nobody wants it." Try making an offer on a slow-moving item. I like to decide how much I would pay for it and make that my ceiling price. The worst that can happen is that you leave it on the shelf. And when you're bargaining, don't appear desperate. If you announce "I'm not leaving without that!" you probably won't, but you will be leaving without a full wallet..

Secondhand goods provide great value for the money. One avenue to search is the classified ad section of your local newspaper. Don't overlook any small paper or shopper that offers free ads. Free ads are best for Commodore goods, since the resale value is often too low to justify paying for an ad.

Other venues include flea markets, garage sales, and user group sales. With secondhand goods, though, it's always buyer beware. Ask to see the item in action; remember to press all the buttons and keys. Put everything through its paces before you hand over the cash. This way, you'll have a great bargain with only a little extra effort.

Who doesn't have friends with an abandoned Commodore lying on a shelf? Persuade them that they'll feel better clearing out the junk they no longer want and you'll be happy to help them out--it can be their gift to you. If nothing else works, try mentioning to whomever cleans the house the advantages of a few fewer boxes cluttering up things!

WHAT TO GET

Now you know where to look, the question is what to get. You may already have several ideas in mind, but let's consider a few alternatives, starting with software.

Think of your favorite game. Are there others like it out there? Or perhaps ones you've always wanted to try? Don't limit yourself to one type or genre of game. If you're into shoot-'em-ups, try a simulation or perhaps a strategy game. Treat yourself to a program you've heard about but never tried.

To get an idea of what's best, check back issues of Gazette for reviews. Some retailers offer grab bags, so you can try a variety of games for one price.

For a real change, think about trying productivity software, such as spreadsheets and word processors. If you're already using them, try a later version or a different brand. Although the learning curve for some programs may put you off, taking on the challenge will help you develop flexibility and make learning the next program even easier.

How about a new bit of hardware? There are dozens of cartridges, chips, and devices available to speed up your disk drive, your printer, or to make life easier. Buy a Rampack and see the advantage of running programs like GEOS. Or buy yourself a second computer or peripheral. If you purchase a second drive, you'll find instant use for it with various software packages. And if you pick up another 64 or 128, you'll have a spare should the first one break. Ditto for the power supply, since it has a reputation for going on the fritz.

As for me, I'm looking forward to getting one of those new quiet printers. (My wife is too!) Or maybe invest in a color monitor or a 128 if you own only a 64. But whether it's an improvement, a spare, or an addition, there's bound to be something out there to please you.

Computer supplies are an area to consider. Miscellaneous items like paper, disks, and ribbons are always needed, since they're always being filled or used up. Try colored or specialty paper and colored disks--even format them for that special touch.

Another thing to look into is memberships. Join a user group. Connect with that online service you've been wanting to try. And for the best investment of money, give or get a magazine subscription. Besides entertaining, a subscription also provides a permanent library of useful info and programs. But then you already know its value; you're reading this!

Gazette, December 1994

GEOS: Writing Patterns

By Steve Vander Ark

Last month, I described how a GEOS program written in either machine language or in geoBASIC is event driven. That means, as you may recall, that the program is arranged around the command MAINLOOP, which is where the program will wait for an event. The programmer's job is to tell GEOS what to do when the various possible events take place, such as a mouse clicking on an icon. Here's the way an event-driven GEOS program is laid out.

    Initial screen and icons drawn
    Mainloop, waiting for events
    List of routines that will happen when each event happens

Since we're going to be using geoBASIC to write a real GEOS program, our program will follow that same pattern. For the sake of example, we'll write a simple program that will display, when called from the menu, any of the 32 built-in system patterns on the screen. Here's our program outline, then, all laid out.

```
@opening
Clear the screen and put a title on it.
Put an icon on the screen that will start our program.

MAINLOOP (The program is going to stay here until something happens.)

@mainscreen
(This is what will happen when our icon is clicked, an event.)
Clear the screen.
Place a set of nice drop-down menus up in the left hand corner.
Go back to MAINLOOP to wait for more input.

@choose
(This is what will happen when the user clicks on the menu choice to
choose a pattern.)
Open a dialog box that asks for the user to input a number from 0 to
33.
Set the pattern to that number.
Draw a rectangle with that pattern.
Go back to MAINLOOP and wait for something else to happen.

@random
(This is what happens when the user clicks on the menu choice for a
random pattern to be chosen.)
Choose a random number between 0 and 33, since there are 34 patterns
available.
Set the pattern to that number.
Draw a rectangle with that pattern.
Go back to MAINLOOP and wait for another event.
```

@quit
(This is what happens when someone chooses the Quit menu option.)
Clear the program screen.
End the program.

@drawRect
(This is a set of instructions for the program to use during @choose and @random to draw a rectangle on the screen.)
Draw a rectangle that pretty much fills the empty space on the screen.
Hold on that for a few seconds.
Clear that rectangle by drawing a new one over top of it using the background pattern.
Go back to the point from which @drawRect was called.

@whoDoneIt
(This is called for when someone clicks on the Program Info menu choice.)
Display the dialog box that gives the author information.
Go back to MAINLOOP to wait for another event.

As you can see, the program will just sit there at MAINLOOP most of the time. As you can also see, all the various parts of the program are labeled with a word or words preceded by an @ symbol. Those labels identify the various sections of the program so they can be jumped to when necessary. For example, when the user clicks on Program Information in the drop-down menus, the program knows to go to a label called @whoDoneIt and perform the commands listed there. That's an important part of a geoBASIC program. Every event that occurs will need to have associated with it one of the labeled parts of the program. You do this when you design the menus and icons themselves, using the built-in editors.

There are five of these editors, all of which are a delight to use. They are very interactive; you can change a label for a menu choice, for example, or add options to an already designed menu or icon with a few simple mouse clicks. There are editors for drop-down menus, dialog boxes, and icons, as well as for sprites and bitmaps. These tools really make creating a GEOS-style, graphics intensive program almost, well, simple! The components you create using these editors are inserted into the actual geoBASIC program with simple, intuitive commands.

Now, before I list the actual program code for geoPatterns, I'd better mention a few cautions. First and foremost, do not, I repeat, do not use the Update menu choice in geoBASIC. It has been found to be buggy, and it will corrupt your program code. As you enter this data, make sure you back up every so often.

To update properly, all you have to do is run the program. There are a number of other known bugs and problems, but this Update bug is the most dangerous because the manual clearly tells you to update, as you might in geoWrite.

Another thing to remember as you look over the program and perhaps type it in is that the various menus and dialog boxes it calls for need to be defined. They aren't part of the code; they're separately defined pieces stored on your disk along with the program itself. Before you can expect geoPatterns to run, you'll need to use the editors to define some terms. The menu, for example, which is called in line 530, is defined as having the name "patrn" with the following attributes set in the editor.

```
    Number of submenus: 3
    Name of the submenu:
      pattern     items: 2
      quit        items: 1
```

(The third submenu is the GEOS menu which is not changeable in the editor.)

Items under "pattern" were defined as "choose pattern" and "random."

Item under "quit" was defined as "quit."

When those items are clicked on in the sample menu at the top of the screen, the editor asks for the labels to attach to them. I assigned them to these labels in my main program: @choose, @random, and @quit.

See how that works? As I worked on this menu in the editor, a sample of the menu appeared at the top of the screen, and I used it to set the attributes for the various menu selections. It's all intuitive and a real joy to work with. The editor for the dialog box that I used to give the author info, referred to by the program as "who," was just as easy to use.

You'll find the completed geoPatterns program on the Gazette disk for this month. Rather than type in the program listing you see below, just boot up geoBASIC and load the program using the name geoPattern. That way you'll have all my defined objects as well. Oh, and by the way, if you don't have geoBASIC, you won't be able to do anything with this program. It's not that geoBASIC programs can't be turned into stand-alone applications--they can. But if I'd done that to my program, you wouldn't be able to edit it at all, and altering other people's programs, adding features and changing things here and there, is an excellent way to learn to program. While you're at it, maybe you can fix the clumsy coding I've included, the erratic line numbers, and inelegant ways of doing things. I would really love to see a few more people get excited about geoBASIC. After all, just look at what a novice programmer like me can do!

Here's the program listing.

```
10 REM geoPatterns
20 REM by Steve Vander Ark
30 REM GEOS column
40 REM November, 1994
```

```
50 REM
100 @opening
110     CLS
120     PATTERN 10
130     RECT 80,50,290,170
140     PATTERN 1
150     RECT 60, 30, 270,150
160     PATTERN 0
170     RECT 61,31,269,149
310     BITMAP "open",17,60
320     WINDOW 65,55,269,149
330     PRINT:PRINT:PRINT:PRINT"       by Steve Vander Ark"
340     ICON "start"
350 MAINLOOP
500 @mainscreen
510     WINDOW 0,0,312,199
520     CLS
530     MENU "patrn"
550     RETURN
600 @choose
610     DBSTRN "Enter pattern number (0-33)",Y$
620     IF VAL(Y$)>33 THEN 610
640     PATTERN VAL(Y$)
650     GOSUB @drawRect
690     RETURN
700 @random
710     Y=RND(34)
720     PATTERN Y
730     GOSUB @drawRect
790     RETURN
800 @quit
810     CLS
820     END
900 @drawRect
910     RECT 30,30,300,180
920     FOR Z=1 TO 500
930     NEXT Z
940     PATTERN 0
950     RECT 30,30,300,180
990     RETURN
1000 @whoDoneIt
1010     DIALOG "who"
1020     RETURN
```

WAX UP YOUR COMMODORE;
WE'RE GOING INTERNET SURFING!

By Nick Rossi

You'll be able to send a fax...from the South Pole! You'll be able to get married...at a cash machine! Or receive open heart surgery...in a taxi! We hear wild promises of things to come on the information superhighway during every commercial break these days. But what's with all the fuss about the superhighway as it's called today: the Internet? Why is Bill Clinton (president@whitehouse.gov) on the Internet? Why is Ted Koppel (nightly@nbc.com) on the Internet? Why is Adam Curry (formerly of MTV) (adam@metaverse.com) there, too? Why do so many people talk about the Internet--or use its trendy nickname, the 'Net--even if they don't understand it? Are they all just trying to sound cool?

The Internet, in case you've missed the news, is a huge network of thousands of mainframe computer systems at universities, corporations, and institutions around the world. Each of these sites, as the computer systems are called, hosts anywhere from a single person to thousands of users. The Internet has no hub or central computer; sites just transmit data to each other through a variety of network connections.

When you connect to the Internet as an end user, you're not actually tapping into the flow of network data traveling between host sites (although a few individual power users do). Instead, you dial into an Internet provider, which might be a BBS or host system that is connected to the network data stream and provides you with access to the services that use the network.

There's an important conceptual hurdle to understand about host machines: They are typically mainframe computers using the Unix or VAX/VMS operating system. As a user, you can run applications on the mainframe, just as you run application programs on your home computer. The only difference is that the mainframe can handle many users and many programs at the same time. Each user can run a program, and the mainframe makes sure the output of the program that you run gets directed to your screen. Therefore, when you use an Internet service, you're basically running a program that's installed on the mainframe.

The various Internet services are akin to a BBS: There are message areas (called newsgroups in Internet jargon), download areas (called FTP sites), and real-time chatting (such as IRC, MUD, and MOO). But unlike a BBS, the Internet's sheer immensity guarantees that you'll find information about anything your brain desires, whether in a newsgroup, an FTP site, or in one of the hundreds of public information servers world-wide. And you'll interact with people from all parts of the world.

Hold the phone! We Commodore users have always been suspicious of phrases like "bigger is better." That's why we haven't totally upgraded like our friends and neighbors, who are cruising around in shiny new 486-DX2/66 clones with twin-carb 540-megabyte hard drives and platinum-tipped triple-speed CD-ROM drives. But when we look down at our Commodore 64 or 128, we still tend to feel like David Spade in that other commercial, trying to merge on the information superhighway in a '72 orange hatchback with fog tires and a brown door. As Commodore BBSs disappear and QuantumLink counts the days until its demise, it seems like the big, nasty corporations are demolishing our comfortable neighborhoods to make room for the great interchanges of the superhighway.

All of this can be traumatic for the longtime Commodore user! However, the best therapy is to find a new electronic home on the Internet. Trust me; there is a thriving Commodore community on the Internet, and its members are doing all sorts of neat 8-bit things. (As I said, there's information about everything on the Internet, and Commodore is no exception.) There are Commodore newsgroups, Commodore FTP sites, Commodore realtime chat areas, and there's even a World Wide Web page just for Commodore users. Don't worry; I'll explain that term later.

In fact, the Internet is the perfect place for Commodore users. Since there are fewer of us today than there were years ago, there may not be enough in any given area to form user groups and provide mutual support. On the Internet, however, we can interact with Commodore folks from all over the world. (There are regulars from the U.S., Canada, Australia, and all over Europe.) On a global scale, there are enough of us to perpetuate the species. Since our programs are small and our needs are simple, we don't take up too much bandwidth, and we can always find a niche on some big host machine for our files.

HOW TO GET ONLINE

So all you have is a 1670 modem? Don't worry; that isn't too slow for the Internet. Nobody's going to get behind you on the superhighway and flash their lights at you for going too slow.

Internet providers generally come in two flavors: local BBSs that have evolved into host machines with Internet connections or online networks, such as America Online, CompuServe or Delphi, that have added Internet connectivity to their services. In either case, they gladly welcome users at any baud rate, although some systems restrict the use of 300 bps.

Once you get hooked on that flow of information that comes from the Internet, you'll probably want a faster modem. You may or may not be aware of this, but 14,400 bps modems are the standard these days. That speed--about 6 times faster than 2400 bps and 12 times faster than 1200 bps--is just about right for using Internet services effectively, simply because there's so much information to scan through, and you can only get it as fast as it appears on your screen.

Due to the combined efforts of programmers and hardware designers over the past several years, Commodore computers are now capable of operating at up to 38,400 bps. Creative Micro Designs sells a cartridge called SwiftLink that provides this capability. You can connect any high-speed Hayes-compatible modem with an RS232 serial port to the back of a SwiftLink. Several terminal programs for both the 64 and 128 are SwiftLink compatible, including Novaterm 9.5 for the 64 and Desterm v2.1 and Dialogue v2.2 for the 128. If you can swing the cash, I highly recommend getting a 14.4-kbps modem, a SwiftLink, and one of the terminal programs for heavy Internet usage. (Make sure you get comfortable with the Internet first before you make this investment!)

The hardware is the easy part. The real trick is finding an Internet provider at a cost you're willing to live with. If you're a college student or faculty member at any small college or big university, your school will definitely have an Internet connection, and you may be able to use it for free. Or, you may work for a large company with Internet access, in which case you can use the Internet through your account at work. If you're not a student or an employee, you may be able to finagle an account through a friend from a local university or company. The rest of us, however, have to cough up the cash.

All of the major online networks--CompuServe, Delphi, America Online, and so on--provide basic Internet services. They all charge basic monthly and/or hourly rates, and use of Internet services is included. On these systems, you can get the help of online support staff.

There are a few wide-area systems that provide full Internet access which go beyond what the major networks provide, as you'll see shortly. Netcom and CERFNet are two of the largest. They're generally comparable in price to the major online networks.

A lower-cost option to the major networks is a local BBS that has grown to provide Internet access. Its rates are often lower than the big guys', and it still has a BBS atmosphere, often having an active local message base that's not connected to Internet. Some of the BBSs provide limited Internet access, but others give you the full range of services. (If the BBS is run on a Unix computer, it's likely that it can provide complete Internet access.)

People on local BBSs are generally more available to help you with problems. In Seattle, for example, I use a system called Eskimo North, which was a local BBS long before it was an Internet provider. For as little as $8 per month (depending on the payment plan), I have unlimited Internet access on a friendly, growing system with no hourly charges.

You should check local newspapers, computer newsletters, and local bulletin boards to find out if low-cost local Internet access is available in your area.

INTERNET BASICS

So once you have an Internet account, where in the world is John Q. Internet User?

The Internet consists of host machines (sites) and users (people). As you might expect, every host has a name, and every user on a host has a log-in name. A user's E-mail (electronic mail) address combines the user's log-in name with the host name. It therefore uniquely identifies the user within the Internet network. Some typical user names are listed below.

nrossi@hmcvax.claremont.edu
voyager@eskimo.com
romeo@tolsun.oulu.fi

The first example is the E-mail address I had in college, and the second example is my current address. Anyone on the Internet can send me mail by typing this address for the message recipient (from within his or her favorite mailer program). The format is standard although it may look confusing at first glance. My user name, nrossi, is followed by the At symbol (@), and that is followed by the name of the host where I had an account, hmcvax.claremont.edu. Capital letters are usually not necessary.

The host name also has a format. It usually consists of two or more fields (words) separated by periods. The final field is a general classification for the host. In the U.S., it is almost always one of the following.

edu	- educational institution
com	- commercial/company
org	- nonprofit organization
gov	- government
mil	- military

Outside the U.S., the final field represents the originating country.

au	- Australia
ca	- Canada
de	- Germany
it	- Italy
ru	- Russia
fr	- France
jp	- Japan
uk	- England
fi	- Finland

The first field identifies the name of a specific host computer. The middle fields identify groups and subgroups within the network. For example, "claremont" identifies the group of computers at the Claremont Colleges, and "hmcvax" picks out a particular mainframe computer at Harvey Mudd College. You can often guess where a person is located from the names of the middle fields.

There's a handy book at the local bookstore called "The Internet White Pages." It contains E-mail address listings for every Internet user in the world. Caveat: It doesn't cover E-mail gateways, where you send

E-mail from one network to another. For instance, from an Internet account you can send E-mail to someone on FidoNet, a nationwide network of BBSs, but "The Internet White Pages" doesn't list FidoNet addresses.

NEWSGROUPS

So now you know how to address Internet users, but where do all these alleged users hang out?

Internet providers give you access to the most popular Internet service: newsgroups. A newsgroup is simply a networked message base. It's just like the ones you use on a BBS, except that every Internet user in the world can read and post to them. You use a program called a newsreader (located on the host machine) to read and write messages.

Every newsgroup has a name, and newsgroup names follow a format much like host names.

alt.urban.folklore
alt.religion
rec.video.production
rec.video.games
comp.sys.ibm.pc
seattle.forsale

The first field is a general classification that indicates what areas of the Internet receive the newsgroup. Several classifications (alt, rec, and comp, to name a few) are groups available to the entire world. Many regions have their own classifications; for example, newsgroup names that begin with "seattle" generally aren't available outside the Seattle area.

The "comp" hierarchy deals with computers, and "comp.sys" is for computer systems, newsgroups for specific computer types. So where are all the Commodore users? Simple: Just add up the names to get comp.sys.cbm.

The comp.sys.cbm newsgroup is the stomping ground for Commodore users on the Internet. All Commodore activity on the Internet is spawned by discussions and interaction there. It's like our clubhouse. We get all kinds of Commodore users in comp.sys.cbm, from people who just pulled the 64 out of the closet to serious Commodore gurus like Fred Bowen, engineer for the late Commodore Business Machines. Authors of major Commodore software hang around, as well as new programmers, enthusiasts, and just about anyone who cares about the future of Commodore computers.

Comp.sys.cbm averages about 50 to 60 messages per day. Questions are asked and questions are answered, software projects are born, ideas are exchanged, and most people have fun.

Notice that there is a five-part message labeled comp.sys.cbm: FAQ.

FAQ stands for frequently asked questions, a common term on the Internet. Many newsgroups have a corresponding text file (often called a FAQ file) that lists answers to questions commonly posted to the newsgroup. These files are an attempt to reduce unnecessary message traffic by keeping people from asking the same questions over and over. It's considered proper etiquette to check the FAQ list before asking a question in the newsgroup. This is especially true if you are a new user.

There are two other Commodore newsgroups on the Internet. One group, called comp.binaries.cbm, is a place where people post ASCII-encoded programs in messages. You need a program called Udecode, a version of which is available for the 64/128, to obtain the original files. Once you get some experience on the Internet, you can read the comp.binaries.cbm FAQ to find out how to download these files.

The other newsgroup is comp.emulators.cbm. Because of the recent popularity of Commodore 64 emulators for other computers, a newsgroup was created to discuss them. It's a relatively new newsgroup, so it only gets around ten messages per day, but the message traffic increases as emulators become more popular.

Once you get used to comp.sys.cbm, you can forage for more topics. Remember: There are thousands of topics available. Interested in cars? Try rec.autos. Need to read about video production? Try rec.video.production. Happen to be a watcher of Comedy Central on cable television? Try alt.tv.comedy-central. You name the topic; Internet has the newsgroup.

End of part 1

December, Gazette 1994

WAX UP YOUR COMMODORE;
WE'RE GOING INTERNET SURFING!

Part 2

By Nick Rossi

TALK, PHONE, AND IRC
Great. You can leave messages to people, but can you actually chat with anyone?

Most of us Commodore modem users have been up late at night on a local BBS, had a problem, and decided to page the sysop about it. The sysop grumbled but answered your page, and the two of you typed back and forth.

With a program called Talk or Phone, you can do the same thing on the Internet. For instance, if I'm logged into eskimo.com as'Voyager, and my friend Galaxyranger is also logged in, I can type TALK GALAXYRANGER to set up a split-screen chat mode. If he answers my talk request, we start typing back and forth in realtime. My typing appears in the upper window, and his appears in the lower window. Simple enough.

Now, suppose I want to talk to an old college buddy of mine, Prusso, who is logged in on my old school's computer, hmcvax.claremont.edu. I can type TALK PRUSSO@HMCVAX.CLAREMONT.EDU (yep, that's his E-mail address), and if he's logged in, we can type back and forth in realtime. We have a two-way, realtime chat between Seattle and southern California, and there's no long distance phone bill!

OK, so you can do that on any major online network like CompuServe. The beauty of doing this on the Internet is that the two of us aren't logged into the same system as we would be on CompuServe. (Remember, the Internet has no central computer, so our talk session involves the exchange of network data packets between my host computer and his.) This matters because I have another friend in Germany, and I can chat to him the same way by typing TALK TRICHTEX@PRFHFB.FH-FRIEDBERG.DE. Or, if I want to talk to my friend in Australia, I can type TALK MIKE@DEFIANCE.VUT.EDU.AU. Most major online networks aren't worldwide, so I couldn't chat with these friends on one of those systems. It would also cost me more than $2 a minute to call either of these people on the phone. (Where did I meet someone from Germany or Australia? Take a guess: The answer starts with an I.)

The Internet has its own public conferencing, similar to what you'd find on the major networks. The Internet's version is called IRC, which stands for Internet Relay Chat. The IRC system is much like CB radio in that people type to each other on "channels." When using IRC, you must join a channel before you can communicate with anyone.

Commodore users can be found on IRC as well. Everyone gets together in a channel called C-64. In the evenings, there are usually anywhere from 5 to 20 Commodore users from all over the world chatting there.

DOWNLOADZ! MORE WAREZ!
Great. Now where are all the Commodore programs?

Looking for Commodore programs such as disk utilities, word processing, or accounting software? Come on; you can admit it--you want GAMES! You can get all these on the Internet.

One thing you can do on the Internet (provided you have full Internet access, which you can't get on CompuServe, or Delphi) is to actually log into another host machine anywhere on the network. To use my E-mail address example, I can be logged into my account as Voyager on eskimo.com in Seattle, and from there I can use a command called "telnet" to establish a connection to hmcvax.claremont.edu and log in as Nrossi. I can use my account on the far-away system just as if I had dialed it directly! I need a user name and password on the other system, of course. And I don't even lose my connection to eskimo.com; I get control of that again when I log out of hmcvax. The Internet just gets niftier all the time, doesn't it?

There is an alternate method for connecting to another host machine. Instead of getting a full, normal log-in like "telnet," which would let me use all the Internet services from the new host, I can connect with a command called "FTP" that only gives me access to files. As you might guess, FTP stands for file transfer protocol. After I FTP to hmcvax (as it's phrased), I can copy files between my account on hmcvax and my local account on eskimo, but I can't do anything else. I still need my user name and password on the other host system to get at my files.

Logging in with FTP just to get access to files sure sounds like a download area, doesn't it? In fact, there are sites you can FTP that allow you to give the user name "anonymous." These anonymous FTP sites, as they're called, contain files that are available to the public. You can grab as many files as you like. You copy them to your local account and download them to your computer later, after you exit the FTP program.

Some very generous users have set up and maintain anonymous FTP sites for 64 and 128 software. When you FTP to one of these sites, you often have to change to a certain subdirectory where the files are located. (Look in a book on MS-DOS or Unix if you're not sure what a subdirectory is.)

Thus, when FTP sites are listed (about twice a month in comp.sys.cbm), the subdirectory is supplied as well. (Notice that the names of FTP sites are simply host machine names; that's where the files are actually located.) Some common Commodore FTP sites are:

```
ccnga.uwaterloo.ca    directory: /pub/cbm
watson.mbb.sfu.ca     directory: D:/pub/
```

```
c64 nic.funet.fi      directory: /pub/cbm
eskimo.com            directory: /voyager/Novaterm
```

These FTP sites collectively contain a huge selection of public domain and shareware Commodore software. Most of the software on these sites tends to be recent. That is, you won't find a lot of ancient programs kept around since the mid-1980s. The people who maintain these sites keep them up-to-date, erasing old versions of software when new versions come out.

WWW, ARCHIE, AND GOPHER, OH MY!
Great. The Internet sounds like a big BBS. Come on, where's the bang for my buck?

Think back to why you bought a modem in the first place. (OK, maybe your Aunt Gigi gave it to you for Christmas after you complained about the paisley shirt one year.) You may have been lured by the promise of information at your fingertips. You must have also wanted to communicate with people. After all, it's a communications device.

In the 15 or so years that modems have been around, their use by home computer users has been more or less limited to the basic services I've already mentioned: public messages, electronic mail, file exchange, and realtime conferencing. The Internet offers all these services. However, because the Internet is so vast and collectively has access to so many resources, it can offer much more.

One thing the Internet can supply in quantity is information. There are volumes of information on every imaginable subject located on host machines all over the world, and it's all at your fingertips. A host machine that contains large databases available to the public is typically called an information server, since it serves you information. After all, if you can chat with someone in Australia in realtime, you should be able to get information from a host machine in Australia as well.

The Internet provides tools for you to access these information servers. A complete discussion is beyond this article, but two of the most common are called Archie and Gopher.

Archie is a simple program that searches all anonymous FTP sites, which can be considered information servers, in a sense. You provide a search string, and Archie looks through all public filenames (just the names, not the files themselves) on all anonymous FTP sites in the world, finding file and directory names that contain your search string. If you were to provide the search string "C64," you'd get a list of most of the Commodore FTP sites.

Gopher is a menu-based system for accessing documents from a specific host machine. A host machine that has information set up under the Gopher system is called a Gopher server. There are Gopher servers all over the place. You simply provide a host machine name, and you get access to its database. The information can be on many different subjects.

The World Wide Web (WWW) is another very different type of information system. People have set up World Wide Web "pages" (servers) all over. Where gopher was just a menu-based system, WWW serves you documents that contain references to other documents, just as books can reference other books. Some documents contain real information, and some documents are just menus that point to other documents. You can hop from one document to another along the reference links. The links created by these references form a "web" across the Internet.

The Commodore World Wide Web page, by the way, contains FAQ files for the Commodore newsgroups, a list of Commodore FTP sites, product lists from Commodore hardware and software distributors, and information about 64 emulators for other computers.

There's another handy book, appropriately called "The Internet Yellow Pages," that lists information servers and WWW pages by subject.

BACK AT THE FARM: MUD AND MOO
Great. Is there anything else on the Internet I can waste time on?

Of course there is. Because the Internet is so vast, it also opens up opportunities for new types of communication between users.

Remember how the Internet hosts are just mainframe computers you can run programs on? Well, you can write programs, too. And many Internet users have used this capability to create new types of Internet services, coming up with ways to communicate with people that have never been done before. If you think about it, this makes the Internet unique; it's the only online service where the users themselves have the ability to shape the system and define new ways to use it.

Any programmer with the time, inclination, and wherewithal can create an Internet program and convince others to start using it over the network. You certainly couldn't do this on a system like CompuServe or America Online without getting a job with them. Then, you'd have to convince your boss that you had a good idea, which is a tall order in any company!

You certainly don't have to concern yourself with programming on the Internet, but you can take advantage of the unique services that have been created by Internet users themselves. One of these services is called MUD, which stands for "multiple user dungeon." A MUD is an environment identical to a text adventure game (Zork, anyone?). You can log into a MUD (located on some host machine) and give commands to move around rooms and locations. The only difference is that other users (from all over the world, of course) can log into the MUD. If you're in the same room with them, the MUD tells you and lets you talk to them! You can even create new rooms and new environments. A typical MUD would present you with something like the following.

You are in downtown Dallas. The street bends here, flanked by tall buildings on one side. The president's motorcade is turning the corner. The president waves his hand cheerfully. A gentleman standing next to you is eagerly filming the procession with a new movie camera. To the east is a road underpass, and to the north is a grassy knoll.

```
Users here: Maverick (goof@highball.usastate.edu)
Eliminator (jerk@irs.gov)
Icedra (lady@vine.org)

What now? > take movie camera.
The gentleman slugs you in the stomach.
Icedra says, "Leave that guy alone!"
What now? > tell Icedra, "Take a hike! I want that camera."
```

Obviously, MUDs were originally designed for games, and they're primarily used for games today. You can find out about MUD games in a newsgroup called rec.mud.games.announce.

But wait; there's more.

MOO stands for "MUD Object-Oriented." In a MOO environment, not only can you move from room to room and meet with other users, but you can also manipulate and share objects. MOO designers try to use common metaphors to help people use objects effectively.

For instance, that movie camera in the above example could instead be a collection of real documents---located on information servers---relating to the Kennedy assassination like the "Warren Commission Report," eyewitness accounts, and quotes from books on the subject. An object that represents a collection of documents is called a notebook in a MOO environment. One person can carry around a notebook, and he or she can let others look at and work with it.

Individual documents are called notes. You can pick up notes almost anywhere, from a Gopher server, a World Wide Web page, or from something you wrote yourself. The note and notebook metaphors relate directly to items we use in everyday life, making it easier to work with the information online.

People have extended the MUD/MOO concept beyond the realm of games to turn them into powerful communication tools. For instance, programmers located in different parts of the world can set up MOO environments for collaborating on programming projects online. Program code and documentation can be shared easily by defining them as objects in such an environment.

Electronic "conference centers" are being set up, for instance, by a group of biologists from several major universities. The biologists can enter a MOO that contains a "conference hall" for lectures, a "foyer" for socializing, and "small conference rooms" for smaller meetings, and they can all be located in the same "building." All of this is done online in the old text adventure format.

You can easily see how this concept can extend further to electronic schools, government facilities, or any other meeting place for people who are thousands of miles apart. Someday, of course, we'll see graphical representations of all this, but that'll take time!

BUT ALL I HAVE IS A COMMODORE!

Of course, nothing says you have to get into these new, fascinating developments in telecommunications. You can be quite happy, as I am, with following your choice of newsgroups and downloading files, but everything I've described is accessible with your good old Commodore. You don't have to worry about being road worthy on the information superhighway. (I promise that's the LAST highway metaphor in this article.)

This article only covers a broad sweep of the Internet. There's an entire shelf of excellent introductory books on the Internet at your local bookstore. (Don't buy the Internet software starter kits unless you also have a PC or Macintosh, though!) Just remember that the Internet is the future of telecommunications.

I think Bill Clinton is on the Internet just for re-election purposes, though.

Here are a few companies that can help you get started.

Netcom Online Communication Services
4000 Moorepark Ave. #209
San Jose, CA 95117
(408) 554-8649
Internet: ruthann@netcom.com
Covers California

CERFNet
P. O. Box 85608
San Diego, CA 92186
(800) 876-2373
Internet: help@cerf.net
Covers Southern California, Korea, Mexico, Brazil

Eskimo North
(Internet provider, Seattle area):
BBS (206) 367-3837

The Internet White Pages
IDG Books
155 Bovet Rd, Ste. 310
San Mateo, CA 94402
(415) 312-0650.

The Internet Yellow Pages
Osborne McGraw-Hill
2600 10th St.
Berkeley, CA 94710

Creative Micro Designs
P. O. Box 646
East Longmeadow, MA 01028
(800) 638-3263 (orders)
(413) 525-0023 (info)
SwiftLink RS-232 cartridge: $39.95
Dialogue 128: $29.00

Novaterm 9.5 (C64)
Nick Rossi
10002 Aurora Ave. N. #3353
Seattle, WA 98133
Internet: voyager@eskimo.com
Delphi: NICKROSSI
GEnie: N.ROSSI
Registration: $25 U. S.

Desterm v2.1 (C128)
Steve Cuthbert
Box 196
Radway, Alberta
Canada TOA 2VO

CompuServe
P. O. Box 20212
5000 Arlington Centre Blvd.
Columbus, OH 43220
(800) 848-8199

Delphi@
(800) 695-4005 (voice)
(800) 365-4636 (modem)

America Online
(800) 827-6364, x396 (voice

Gazette, December 1994)