Amazing Online Databases: Instant Information

COMPUTE!

$2.95
July
1985
Issue 62
Vol. 7, No. 7
$3.75 Canada
02193
ISSN 0194-357X

The Leading Magazine Of Home, Educational, And Recreational Computing

Softball Statistics: Is Your Team As Good As You Think?

Ready-To-Run Programs Inside For Commodore 64, VIC, Atari, IBM PC, PCjr, Apple, TI

Extended Color Mode For Commodore Computers

Atari LIST Scroller

An Easier Way To Edit BASIC Programs

Viewports In IBM BASIC

How To Open Windows On Your PC & PCjr

Apple ProDOS Converter

For *SpeedScript 3.0*

0 71486 02193

COMPUTE!

July 1985
Vol. 7, No. 7

FEATURES

REVIEWS

COLUMNS AND DEPARTMENTS

THE JOURNAL

NOTE: See page 93 before typing in programs.

AP Apple, **Mac** Macintosh, **AT** Atari, **V** VIC-20, **64** Commodore 64, **+4** Commodore Plus/4, **16** Commodore 16, **P** PET/CBM, **TI** Texas Instruments, **PC** IBM PC, **PCjr** IBM PCjr, **CC** Radio Shack Color Computer.
*General interest.

TOLL FREE Subscription Order Line
800-334-0868 (In NC 919-275-9809)

COMPUTE! Publications,Inc.
One of the ABC Publishing Companies:
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019
Address all inquiries to:
P.O. Box 5406, Greensboro, NC 27403

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403 USA, Phone: (919) 275-9809. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, $24. Send subscription orders or change of address (P.O. form 3579) to **COMPUTE!** Magazine, P.O. Box 914, Farmingdale, NY 11737. Second class postage paid at Greensboro, NC 27403 and additional mailing offices.

ISSN 0194-357X.

Readers Feedback

The Editors and Readers of COMPUTE!

If you have any questions, comments, or suggestions you would like to see addressed in this column, write to "Readers' Feedback," COMPUTE!, *P.O. Box 5406, Greensboro, NC 27403. Due to the volume of mail we receive, we regret that we cannot provide personal answers to technical questions.*

TurboDisk Translations?

I enjoy your magazine tremendously, especially utility programs such as "TurboTape" [COMPUTE!, January 1985] and "TurboDisk" [COMPUTE!, April 1985]. However, I am a proud Atari user. In the past you have published some interesting and useful programs for Atari, but never something as valuable as the Commodore programs you have published in the last few issues. I hope you will consider printing similar programs for Atari and other computers very soon.

Duyen Nguyen

A number of readers have asked us to translate these programs for other computers, or publish other "breakthrough" programs for their machines. Needless to say, we're as anxious to publish programs of that quality as you are to see them.

What you might not realize is that most of the programs we publish are submitted by readers like yourself. Our Submissions Reviewer has a full-time job testing and evaluating the several hundred programs we receive every month. Very few submitted programs are as extraordinary as the two you mention. But we're always on the lookout, and you can be sure that we'll publish anything of similar quality as soon as it comes in the door.

When a program is particularly good with broad appeal, we do provide translations for other popular computers. For example, the Commodore 64 version of the SpeedScript 3.0 *word processor was adapted for the VIC-20 (April 1985), Atari (May 1985), and Apple II series (June 1985). These are among the best programs we've ever published for those machines. Adapting a large, complex machine language program like* SpeedScript *is far from easy and requires several months of work, but it's possible because word processing is something any computer can do: Every computer can store characters in memory, receive input from the keyboard, and so on.*

Programs such as "TurboTape" and "TurboDisk," on the other hand, are highly machine-specific: *They exploit hardware features unique to the Commodore 64, VIC-20, and 1541 disk drive. Making such programs work on other machines may be technically impossible, or at least require entirely different techniques. But our readers have a habit of surprising us. Perhaps there's someone out there working on Atari TurboDisk or TurboTape right now.*

Hints For ON-GOTO

I want to commend you on promoting the ON-GOTO/ON-GOSUB commands ["The Beginner's Page," COMPUTE!, March 1985], but I think you missed the best (most useful) aspect of them—the ability to perform math functions within the line. In your example program, I would have preferred to see ON A−4 GOTO *linenumber* instead of A=A−4:ON A GOTO *linenumber.* My method preserves the value of A and saves a line of code.

I keep a stat file for our church softball team on a menu-driven program I wrote. I've used ON-GOSUB with the function keys on my Commodore 64 to greatly simplify coding [see below]. The function keys have ASCII values from 133 to 140 in the order listed in my program (f1, f3, f5, f7, f2, f4, f6, f8), so ASC(F$)−132 gives me nice neat numbers from 1 to 8.

Matthew Strange

You're absolutely right, of course—it is indeed more efficient to convert a value for ON-GOTO/ON-GOSUB within the line itself, rather than making it a separate line. As you point out, it saves a little memory and preserves the original value, which may be important in some cases.

In general, the programming examples in "The Beginner's Page" are written more for clarity than for maximum efficiency. For that reason, and also to keep the examples compatible with all the computers we cover, each statement is usually placed on a separate line. (For instance, the built-in BASIC on the TI-99/4A allows neither multistatement lines nor any command following an IF-THEN except GOTO linenumber.*) Readers are encouraged to experiment with the examples and find ways to optimize them for their particular computer and version of BASIC.*

Your method of reading the Commodore 64 function keys is efficient and can be adapted to reading keys and joysticks on a variety of computers. As a module, it can be plugged into any program that prompts a user to select a menu option:

```
10 PRINT "PRESS F1 TO VIEW STA
   TS"
20 PRINT "PRESS F3 TO UPDATE S
   TATS"
30 PRINT "PRESS F5 TO SAVE STA
   TS"
40 PRINT "PRESS F7 TO PRINT ST
   ATS"
50 PRINT "PRESS F2 TO ENTER NE
   W PLAYERS"
60 PRINT "PRESS F4 TO CREATE N
   EW TEAM FILE"
70 PRINT "PRESS F6 TO VIEW FIL
   E NAMES"
80 PRINT "PRESS F8 TO CORRECT
   {SPACE}ENTRY"
90 PRINT "PRESS Q TO QUIT"
100 WAIT 198,1:GET F$:IF F$="Q
    " THEN END
110 IF ASC(F$)<133 OR ASC(F$)>
    140 THEN GOTO 100
120 ON ASC(F$)-132 GOSUB 2000,
    3000,1000,3500,2500,9500,5
    300,4000
```

(This program fragment assumes that subroutines will be added at the linc numbers specified in the ON-GOSUB state-

ment in line 120, so if you run the program as listed and press a function key, you'll get an error.)

The program employs a couple of other useful techniques, such as waiting for a keystroke (line 100), trapping invalid input (line 110), and extracting numeric values from string variables (lines 110 and 120). The main point is line 120, which neatly converts the value returned by the function key into a value that can be acted on by ON-GOSUB. At the same time, it preserves the original value returned in F$. It's a clever way to organize a menu while avoiding confusion over the staggered numbering of the Commodore 64 function keys.

Incidentally, if you're the statistician for your church softball team, see "Softball Statistics" elsewhere in this issue. You may want to include some of its features in your own program.

Atari Translator Disk

I would like to know if it's true that some software will not run on the XL computers unless you load a special program first. Do I have to buy this program or is it available for publication? If I have to buy it, does it come on both cassette and disk?

Randall E. Nowak

The program you're looking for is the Atari Translator Disk, *available from Atari and some local user groups for a nominal fee. The part number is DX-5063. The* Translator *loads the old Atari 400/800 operating system into RAM on the 800XL or 600XL (with 64K), temporarily replacing the XL operating system. With the* Translator, *your XL computer can run most programs which were not written to comply with Atari's published guidelines for upgrade compatibility.*

A few independent software companies also sell XL translator disks which advertise more flexibility than the Atari Translator.

Computer Vocabulary

I want to write a BASIC text adventure game for my IBM PC. How can I give my program a "vocabulary" so that it understands dozens of words, without using IF statements to check for every word, in every room of the game?

Bill Grau

You're correct—an adventure program that tests for every condition with individual IF statements would be woefully inefficient. To answer the second part of your question first, you'll want to structure your program in modular, rather than linear fashion. While it seems straightforward to write a separate routine for each room in the adventure, this is highly duplicative and will make your program as difficult to debug as most adventures are to play. Instead, use general subroutines that perform the same task no matter where the player is. You need only one routine to analyze the player's input, simulate movement, handle objects, and so on.

The best way to create a vocabulary in BASIC is with arrays. *Unlike a simple variable which equates to a single numeric value (A=1) or string of characters (A$="HELLO"), an array is a* group *of related data items. The short example program below (written in plain-vanilla Microsoft BASIC) creates a rudimentary vocabulary with string arrays.*

```
100 DIM VB$(4),OB$(4):FOR J=1
     TO 4:READ A$,B$
110 VB$(J)=A$:OB$(J)=B$:NEXT
    J
120 DATA TAKE, GOLD, PUT, SWO
    RD, EAT, FOOD, THROW, ROC
    K
130 SP=0:I$="":VB$="":OB$="":
    PRINT "YOUR COMMAND";
140 INFUT I$:FOR J-1 TO LEN(I
    $):T$=MID$(I$,J,1)
150 IF T$=CHR$(32) THEN SP=1:
    GOTO 180
160 IF SP=1 THEN OB$=OB$+T$
170 IF SP=0 THEN VB$=VB$+T$
180 NEXT J:VB=0:FOR J=1 TO 4
190 IF VB$=VB$(J) THEN VB=J
200 NEXT J
210 IF VB=0 THEN PRINT "DON'T
     UNDERSTAND ";VB$: GOTO 1
    30
220 OB=0:FOR J=1 TO 4:IF OB$=
    OB$(J) THEN OB=J
230 NEXT J
240 IF OB=0 THEN PRINT "DON'T
     UNDERSTAND ";OB$: GOTO 1
    30
250 PRINT "VERB #";VB;", OBJE
    CT #";OB:GOTO 130
```

Lines 100–120 store the vocabulary in two string arrays. The array named VB$() holds four verb strings (TAKE, PUT, EAT, and THROW) and the OB$() array holds four object strings (GOLD, SWORD, FOOD, and ROCK).

Once the arrays are set up, each word can be referenced by the index number *that identifies its position within the array. For instance, in response to the statement PRINT V$(1), the computer prints TAKE. The statement IF A$=OB$(1) is true when A$ equals GOLD, and so on. Since the vocabulary has been reduced to reference numbers, you can cycle through the arrays with simple FOR-NEXT loops, testing whether your input words match anything in the vocabulary. This is far more efficient than using a multitude of IF statements.*

Lines 140–180 bring the input sentence into the computer as one string (I$) and break it into two parts: the verb string V$ and the object string O$. Note that simple (nonarray) variables like V$ and O$ are distinct from array variables like V$() and O$(). (Because of space limitations, the program uses a primitive method to extract verb and object from the input sentence: It looks for the space character that separates the words, assigning every character before the space to V$, and everything after it to O$.)

Once the verb and object have been extracted, lines 180–200 compare the verb string V$ to every verb in the vocabulary array V$(). The variable V signifies the verb number. As soon as V$ matches up with a word in V$(), V records the V$() index number for future reference. If V$ isn't found in the vocabulary, line 210 prints the unknown word and lets the player try again. A similar loop in lines 220–230 compares the object string O$ to each word in the O$() array, and records the object number in the variable O.

Using arrays makes your program far easier to modify. For instance, say that you've written an adventure using dozens (or more likely, hundreds) of separate IF statements, and then decide to change one of the vocabulary words. It could take hours to locate and change every line that uses that word. If your vocabulary is stored in arrays, you can make the same change in seconds, by replacing one word in a DATA statement. To expand the vocabulary, just add more DATA items and increase the values in the DIM statement and FOR-NEXT loops accordingly.

Of course, there's much, much more to writing a playable adventure. But arrays can help there, too. Use a room description *array to store the description strings for each* room, *and a* room connection *array to show the connections between them. The location of each object can be stored in an* object location *array, and so on. You'll want a more sophisticated* parsing *routine as well, to pick apart the input sentence. These techniques and others are explained in* COMPUTE!'s Guide to Adventure Games *by Gary McGath.*

Commodore Chips, Anyone?

We are a group of about 200 Commodore owners in Sao Paulo, Brazil. Because of import restrictions and the great distance involved, we are not able to send Commodore products to the United States for servicing. Some minor repairs, like aligning the 1541 disk drive, we manage to do here. But we have not been able to find anyone who sells microprocessor, interface, or PLA chips. On a recent visit to New York City, I inquired of several dealers, but they could not help us.

Alberto Dayan
Rua Albuquerque-Lins 867 Apt. 1401
Sao Paulo, 01230 S.P.
Brazil

Every once in a while, our mailbag contains both an excellent question and an excellent answer (see the following letter). Unlike most other computer companies, Commodore manufactures its own chips; in 1976 it acquired MOS Technologies (which invented the 6502 microprocessor), and it still manufactures the 6510 micro-

processor, SID chip, and other major components for the Commodore 64.

Commodore has traditionally shown little interest in making its own chips available to individuals. One reason for this is undoubtedly economics: Commodore is in the business of selling computers, not electronic parts, and has little or no economic incentive to market chips at the retail level.

By the time you read this, however, the chip famine may be ending. According to a company representative, Commodore intends to let its new network of service centers provide repair parts directly to consumers. Of course, it's hard to predict exactly when this will become a reality. In any case, once the national service network is established, Commodore plans to abolish its exchange program. You will no longer be able to replace damaged Commodore products by sending them back to the factory with a check.

Commodore Repair Parts

After you published my address in COMPUTE! (December 1984), I got a great deal of mail from readers with repair problems. I now have a supplier for all Commodore parts, including chips. If anyone has difficulty getting a part, I can help. I also welcome questions on repairs.

Steve Fogolini
8232 Richard Street
Fort Worth, TX 76108

Thank you for the information.

Atari String Handling

I recently heard that the Atari 600XL computer's built-in BASIC doesn't use string arrays. Does this mean that it won't accept programs that use string arrays from the other Atari computers?

Scott Powers

There's nothing to worry about. The BASIC built into the 600XL and 800XL (and new 65XE/130XE) is fully compatible with earlier Atari BASICs. The information you heard about string arrays refers to Atari BASIC in general. True string arrays work like numeric arrays, where each element is a complete string consisting of one or more characters. For example, the statement A$(10)="HELLO" would assign the complete string "HELLO" to the tenth element of the string array A$. In Atari BASIC, however, this statement places the string "HELLO" into the tenth position of the string A$.

*Versions of BASIC that support string arrays cannot use the convenient Atari method of substringing, such as A$(5,5) to access the fifth character of A$, but use functions like MID$(A$,5,1) instead. The Atari can simulate string arrays with substrings. A long string can hold many substrings. For example, to convert the statement A$(5)="CAT", assuming you limit the length of each substring to ten characters, you can use the statement A$(5*10−9,5*10)="CAT". Since the length of an Atari string is limited only by the amount of free memory, you can store a great number of substrings within one long string.*

If you prefer a BASIC with true string arrays, alternative BASICs (including Microsoft BASIC) are available on cartridge and disk for Atari computers.

Hex/Decimal Conversion

Is there a simple formula for converting hexadecimal numbers to decimal, and vice versa? The manual for my computer (an Apple clone) doesn't explain this in much detail.

Howard Heapy

Here's a short Microsoft BASIC program that does both conversions within the range of hexadecimal $0–$FFFF (decimal 0–65,535). When converting from hex to decimal, enter a four-digit hex number, using leading zeros when appropriate. For instance, enter 00FF to find the decimal equivalent of hexadecimal $FF.

```
10 HE$="Ø123456789ABCDEF"
20 PRINT "ENTER 1 FOR DEC TO
   HEX"
30 PRINT"ENTER 2 FOR HEX TO D
   EC";:INPUT A
40 IF A=2 THEN 100
50 IF A<>1 THEN 20
60 PRINT "ENTER DEC #";:INPUT
    A:B=1:C=3:D=16^C:PRINT A;
   :PRINT "= $";:A=A+1
70 IF A-D>Ø THEN A=A-D:B=B+1:
   GOTO 70
80 PRINT MID$(HE$,B,1);:B=1:C
   =C-1:D=16^C:IF C>-1 THEN 7
   Ø
90 PRINT:PRINT:GOTO 20
100 PRINT "ENTER HEX #";:INPU
    T H$:D=Ø:Q=3
110 FOR M=1 TO 4:FOR W=Ø TO 1
    5
120 IF MID$(H$,M,1)=MID$(HE$,
    W+1,1) THEN 140
130 NEXT W
140 D1=W*(16^(Q)):D=D+D1:Q=Q-
     1:NEXT M
150 DE=INT(D):PRINT"$";H$;" =
    ";DE
160 PRINT:GOTO 20
```

Atari users should make the following changes to this program:

```
5 DIM HE$(16),H$(4)
80 J=1:H$=HE$(B,B+J-1):?
   H$;:B=1:C=C-1:D=16^C:I
   F C>-1 THEN 7Ø
 12Ø IF HE$(W+1,W+1)=H$(M,
     M) THEN 14Ø
```

TI Serial Communications

I own a TI computer and have recently decided to buy a printer. While shopping around, I noticed that some printers come with built-in or optional RS-232-C serial interfaces. Does this mean that I can connect the printer to my computer with a "run of the mill" printer cable, or would I still need to buy a TI interface card to get the printer working? If I need the interface card, what purpose does the built-in interface serve?

Crandall Chow

You do need the interface card. A serial data link can transmit one bit (binary 1 or 0) of information at a time, but the computer and printer handle each ASCII character as a byte (eight-bit binary number). Since a byte contains eight bits, you can't send the whole chunk at once through a serial link.

Picture a group of eight friends walking side by side. If they come to a narrow turnstile, they can't all enter at once, so they pass through singly. When everyone is through the turnstile, they reassemble the group and march eight abreast once more. First the group is broken up, then it is reassembled.

The analogy explains why you need an interface at both ends of the serial link. At the computer's end, you need an interface to break each ASCII byte into eight bits and send each bit down the line in order. At the other end of the link, the printer's interface converts each series of eight bits back into a byte which the printer can handle as an ASCII character. In addition to the data bits, extra bits are passed between the computer and printer to coordinate the transmission process.

Because each bit has to pass singly, you might expect serial transmission to be slower than parallel data transfer, which passes more than one bit at once. This is not always true, however. Since serial transmission uses a single data wire, it's less susceptible to electrical interference than the parallel arrangement, where a signal traveling down one of the eight parallel data wires tends to create "noise" on neighboring data wires. Hence, you can accurately send bits serially much faster than you can send bytes in parallel. For example, the RS-232 serial TI-to-printer link we use, which operates at 9600 bps (bits per second), is as fast or faster than any of the parallel links we use with other computers and printers.

Multiple Entry Points In ML

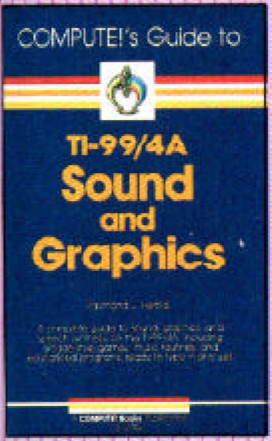
While disassembling ROM routines in my Commodore 64, I noticed that the LOAD routine at $E168 seems to begin with the low byte of an address, rather than an opcode. This is true of other routines as well. Is there something wrong with my disassembler, or have I missed something?

J. C. Vollmer

There's nothing wrong with your disassembler. You've come across a memory-saving machine language trick that can look baffling unless you already know its purpose: to provide more than one entry

A special offer from COMPUTE! Books for owners and users of TI-99/4A computers.

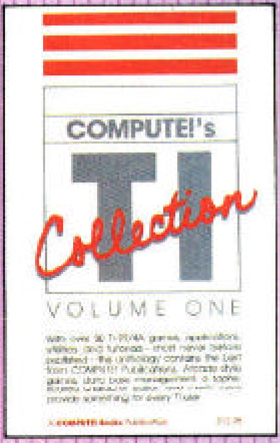
Buy any two of these books and get 15% off.
Buy all four of these titles for only $39.95 (a savings of $13.85—over 25% off).



COMPUTE!'s Guide to TI-99/4A Sound and Graphics
Raymond J. Herold

Using dozens of examples and clear, nontechnical explanations, *COMPUTE!'s Guide to TI Sound and Graphics* introduces you to the tremendous sound and graphics capabilities of your TI home computer. This book includes exciting arcade-style games, challenging educational programs, a versatile sprite editor, and many more useful, ready-to-type-in programs. Whether you're a beginning TI user or an experienced programmer, this is a book you'll refer to again and again.

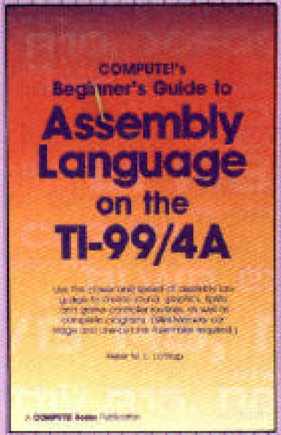
$12.95
ISBN 0-942386-46-9



COMPUTE!'s TI Collection, Volume One

An anthology of *COMPUTE!*'s best games, applications, utilities, and tutorials for the TI-99/4A, this book includes more than 30 programs, most never before published. "SuperFont" is an exceptionally powerful and easy-to-use character editor. Other utilities, such as "Sprite Editor" and "Sound Shaper," make graphics and sound programming simple. Games like "Worm of Bemer" will provide hours of fun, and applications like "Mailing List" let you use your TI to organize your home.

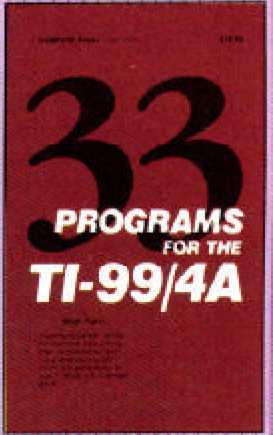
$12.95
ISBN 0-942386-71-X



COMPUTE!'s Beginner's Guide to Assembly Language on the TI-99/4A
Peter M. L. Lottrup

A clearly written, step-by-step tutorial on machine language (ML) programming for the TI-99/4A. Using the Line-by-Line Assembler, the basic concepts of machine language programming are fully explained and illustrated. Many practical, easy-to-follow ML examples are included, from elementary keyboard, joystick, and data handling routines to sophisticated redefined character, sprite manipulation, and even high-resolution bitmap mode techniques.

$14.95
ISBN 0-942386-74-4



33 Programs for the TI-99/4A
Brian Flynn

Contains something for everyone: chapters on games ("Rings and Poles"), money management ("IRA Planner"), business ("Internal Rate of Return"), simple statistics ("Mean, Variance, and Standard Deviation"), and more. A wide variety of applications software, plus games. Thirty-three ready-to-type-in programs at a low cost.

$12.95
199 pages, spiral wire bound
ISBN 0-942386-42-6

Order today!
This offer expires
August 10, 1985

Yes! I want to save money while enjoying COMPUTE! Books.

______ COMPUTE!'s Guide to TI-99/4A Sound and Graphics $12.95
______ COMPUTE!'s TI Collection, Volume One $12.95
______ COMPUTE!'s Beginner's Guide to Assembly Language on the TI-99/4A $14.95
______ 33 Programs for the TI-99/4A $12.95

All orders must be prepaid.
□ Payment enclosed (check or money order).
□ Charge: □ Visa □ MasterCard □ American Express
Acct. No. ______________ Exp. Date ______
Signature ______________
Name ______________
Address ______________
City ______ State ______ Zip ______

To order call toll-free, 1-800-334-0868 (in NC, call 919-275-9809) or mail this coupon with your payment to: COMPUTE! Books, P.O. Box 5058, Greensboro, NC 27403.

Please send me: □ 4 books for $39.95
□ 2 books at 15% off
NC residents add 4.5% tax ______
Add $2.00 per book for shipping and handling ______
Total paid ______

Please allow 4-6 weeks for delivery.

757114G

choose
NON in
prompt.
you can
key, and

bbrevia-
ecessary
You can
COMPU,
ou must
ımation
es. If the
y Butter-
ring BU
ticles by
names
s. Enter-
the two
tterfield

options 5
ore than
Option 5
n to find
ategories:
COMPUTE!
Charles
and enter
the num-
enter the
ames as

OR func-
ber of two

Fast Filer

Richard Mansfield, Senior Editor, and Patrick Parrish, Programming Supervisor

Maintain a master index of magazine articles with this short BASIC program for the Commodore 64, 128, VIC-20, Plus/4, 16, PET, Atari, Apple II series, IBM PC/PCjr, and TI-99/4A. With slight modifications, the Commodore version can work on any computer with Microsoft BASIC.

How many times have you been working on a program when you recall a magazine article that has just the information you need—but finding it is another matter? That is, you know the article's *somewhere* in the house—but where? You could spend hours paging through back issues to find what you're looking for. Now, with "Fast Filer," you'll have a fast and easy way to retrieve such information.

Enter and save Fast Filer from one of the listings below. Program 1 works on any Commodore computer, including the PET/CBM, VIC-20, 64, Plus/4, 16, and the new Commodore 128 (in 64 mode or 128 mode). Program 2 is the Atari version. Enter Program 3 for Apple, Program 4 for the IBM PC/PCjr, or Program 5 for the TI-99/4A.

If you type in one of the non-Commodore versions, *be sure to add lines 1999–2050 from Program 1.* (TI users should also note line 100, which configures your system for printer output. Check your printer manual and change this line as needed to set up your particular printer.) The program should be easy to convert for other computers (such as the TRS-80) that use Microsoft BASIC. The only lines you need to change are those that involve screen formatting and printer output; consult your user's manual for the proper commands to clear the screen and so on.

Searching The Database

Fast Filer is designed for simplicity and convenience. To search the database, all you really need to do is type RUN and follow the prompts. The program first asks whether you want to send output to the screen or the printer. Then the menu displays several options. You can search the database in several different ways: by magazine title, by author, by subject, by publication date, or by two categories at once.

For example, say you want to list all articles from COMPUTE!. Simply choose option 1 and enter COMPUTE! when prompted for the magazine name. To list all articles by Charles Brannon, choose option 2 and enter BRANNON in response to the author prompt. Once the listing begins, you can pause it by pressing any key, and resume by pressing P.

Fast Filer accepts abbreviations, so it's usually not necessary to type in the entire name. You can abbreviate COMPUTE! as COMPU, for example. However, you must give Fast Filer enough information to distinguish similar names. If the database contains articles by Butterfield and Buncombe, entering BU for the author lists all articles by *both* authors, since both names share those two characters. Entering BUT would distinguish the two names and list all Butterfield articles.

For added flexibility, options 5 and 6 let you search by more than one category at a time. Option 5 provides an *AND* function to find articles that *share* two categories: For instance, to find all COMPUTE! articles written by Charles Brannon, select option 5 and enter 1,2 (be sure to separate the numbers with a comma). Then enter the magazine and author names as prompted.

Option 6 provides an *OR* function to find articles in *either* of two categories. For instance, perhaps you're interested in machine language. With option 6 you could find every article that was categorized under the subject MACHINE LANGUAGE, *or* that was written by Jim Butterfield (who often writes on that subject). The ability to search two categories simultaneously is very powerful.

Easy Data Entry

Of course, no database is useful until it contains some data. Line 1999 of Fast Filer is a template that shows the format for entering data. To enter new data, simply add new lines to Fast Filer, using line numbers higher than 1999. (Lines 2000–2040 are examples which you can modify or delete.)

Every new entry must be in the form of a BASIC line consisting of a line number and a DATA statement, followed by six data items separated by commas. Here is the format:

MAGAZINE TITLE, AUTHOR, SUBJECT, DATE, PAGE NUMBER, COMMENTS

Because Fast Filer separates data items with commas, you must not put commas within the data itself. For instance, enter BRANNON C for an author's name, not BRANNON, C.

You cannot omit any of the data items for a particular entry; if you do, the entire list of data is thrown out of sequence. Instead of leaving a particular item blank, substitute something like N/A (for not applicable). For example, you might have an entry for which you don't feel the need to add a comment, like:

2000 COMPUTE!,READERS FEEDBACK,LOWERCASE FOR TI, 4/85,22,N/A

Pay particular attention to line 2050, which tells Fast Filer it has reached the end of the data. This must *always* be the last DATA line in the program. When adding new data, renumber this line accordingly. When you're done adding data, resave Fast Filer on disk or tape. The next time you run it, all the new data is available. Since the data is appended to the program itself, the size of the database is limited only by your computer's memory.

Designing A Database

Fast Filer provides the basic framework for a database, but for maximum flexibility, it leaves the most important design choices up to you. You are free to choose whatever subject categories you like, making them as general or as specific as your needs require.

Creating categories deserves some careful forethought. Clearly, a subject category like COMPUTERS is too broad to be useful. On the other hand, the subject must have enough breadth to encompass more than one article. Consistency is essential, too. If you pick MACHINE LANGUAGE as a subject, then stick with that subject name; categorizing other machine language articles under subject names like ML or MACH LANG will result in incomplete searches.

Before adding your first entry, you may want to decide on standard names for your major categories. These could be saved for future reference in a written list or added to Fast Filer as REM statements.

Use consistent names for magazine titles and authors as well. If you enter a magazine title as COMPUTE (without the exclamation point), it won't be found when you search for articles under the key word COMPUTE! (although the reverse would work). Likewise, GAZETTE is a more convenient title than COMPUTE!'S GAZETTE.

Fast Filer's ability to abbreviate can work to your advantage. For instance, say that you pick GRAPHICS as a major category. If you enter graphics articles under subject names like GRAPHICS VIC, GRAPHICS C64, GRAPHICS PET, and so on, then Fast Filer can find *all* graphics articles (under the subject GRAPHICS) as well as graphics articles for a particular computer.

There are limits to what Fast Filer can do, of course, as there are with any BASIC program this brief. But its simplicity makes the program easier to customize. One of the best ways to learn programming is to begin with an existing program and alter it to fit your own needs. Such changes can range from the purely cosmetic (changing screen or character colors) to more significant improvements (formatting printer output, adding extra categories, etc.). In fact, with only a few modifications, Fast Filer can be used to index practically anything, from books or record albums to investments, rare coins, or stamps.

Program 1: Commodore Fast Filer

Please refer to "COMPUTE!'s Guide to Typing In Programs" before entering this listing.

```
10 DIMA$(6):G$="{7 RIGHT}":REM
    FOR VIC, SET G$=""  :rem 3
20 PRINT"{CLR}{10 DOWN}"G$"PRI
   NT TO {RVS}S{OFF}CREEN OR
   {RVS}P{OFF}RINTER ?":rem 91
30 GETK$:IFK$=""OR(K$<>"P"ANDK
   $<>"S")THEN30       :rem 191
40 DE=-(K$="P")*4-(K$="S")*3:O
   PENDE,DE             :rem 14
50 LABEL$(1)="MAGAZINE TITLE:"
   :LABEL$(2)="AUTHOR'S LAST N
   AME:"               :rem 146
60 LABEL$(3)="THE TARGET SUBJE
   CT:":LABEL$(4)="DATE (IE.,
   {SPACE}1/14/85 OR 1/85):"
                         :rem 5
70 PRINT"{CLR}{5 DOWN}"G$"CHOO
   SE ONE (1-8):":PRINTG$"
   {DOWN}{RIGHT}1. MAGAZINE"
                       :rem 226
80 PRINTG$"{RIGHT}2. AUTHOR":P
   RINTG$"{RIGHT}3. SUBJECT":P
   RINTG$"{RIGHT}4. DATE"
                        :rem 11
90 PRINTG$"{RIGHT}5. AND":PRIN
   TG$"{RIGHT}6. OR":PRINTG$"
   {RIGHT}7. PRINT ALL":PRINTG
   $"{RIGHT}8. QUIT{DOWN}"
                        :rem 59
100 GETK$:IFK$=""OR(VAL(K$)<10
    RVAL(K$)>8)THEN100 :rem 21
110 K=VAL(K$):ONKGOTO120,130,1
    40,150,160,160,300,340
                       :rem 246
120 C=1:GOTO350         :rem 79
130 C=2:GOTO350         :rem 81
140 C=3:GOTO350         :rem 83
150 C=4:GOTO350         :rem 85
160 H$="OR":IFK=5THENH$="AND"
                       :rem 154
170 PRINTG$"# "H$" # (1-4):":P
    RINTG$;:INPUTN1,N2 :rem 73
180 IF(N1<1ORN1>4)OR(N2<1ORN2>
    4)THEN170           :rem 48
190 PRINT"{CLR}TYPE "LABEL$(N1
    ):INPUTI1$:L=LEN(I1$)
                       :rem 141
200 PRINT"{DOWN}TYPE "LABEL$(N
    2):INPUTI2$:L2=LEN(I2$)
                        :rem 56
210 PRINT:Q=0:F=0:RESTORE
                        :rem 99
220 GOSUB450:IFF=1THEN410
                       :rem 238
230 IFK=6THEN260       :rem 169
240 IFLEFT$(A$(N1),L)<>I1$ORLE
    FT$(A$(N2),L2)<>I2$THEN280
                       :rem 141
250 GOTO270            :rem 105
260 IFLEFT$(A$(N1),L)<>I1$ANDL
    EFT$(A$(N2),L2)<>I2$THEN28
    0                  :rem 193
270 Q=1:GOSUB470       :rem 173
280 IFF=0THEN220       :rem 159
290 GOTO410            :rem 105
300 PRINT"{CLR}":F=0:RESTORE
                        :rem 66
310 GOSUB450:IFF=1THEN420
                       :rem 239
```

```
07 12Ø K = VAL (K$): ON K GOTO 1
       3Ø,14Ø,15Ø,16Ø,17Ø,17Ø,31
       Ø,36Ø
86 13Ø C = 1: GOTO 37Ø
A8 14Ø C = 2: GOTO 37Ø
CA 15Ø C = 3: GOTO 37Ø
EC 16Ø C = 4: GOTO 37Ø
9B 17Ø H$ = "OR": IF K = 5 THEN
       H$ = "AND"
AA 18Ø PRINT : PRINT G$"# "H$" #
        (1-4):": PRINT G$;: INPU
       T N1,N2
E8 19Ø IF (N1 < 1 OR N1 > 4) OR
       (N2 < 1 OR N2 > 4) THEN 1
       8Ø
92 2ØØ HOME : PRINT "TYPE "LABEL
       $(N1): INPUT I1$:L = LEN
       (I1$)
37 21Ø PRINT : PRINT "TYPE "LABE
       L$(N2): INPUT I2$:L2 = LE
       N (I2$)
8E 22Ø PRINT :Q = Ø:F = Ø: RESTO
       RE : IF DE = Ø THEN PRINT
        D$;"PR#1": PRINT I$;"8ØN
       "
A3 23Ø GOSUB 48Ø: IF F = 1 THEN
       43Ø
20 24Ø IF K = 6 THEN 27Ø
57 25Ø IF LEFT$ (A$(N1),L) < > I
       1$ OR LEFT$ (A$(N2),L2) <
        > I2$ THEN 29Ø
20 26Ø GOTO 28Ø
E5 27Ø IF LEFT$ (A$(N1),L) < > I
       1$ AND LEFT$ (A$(N2),L2)
       < > I2$ THEN 29Ø
9C 28Ø Q = 1: GOSUB 5ØØ
06 29Ø IF F = Ø THEN 23Ø
11 3ØØ GOTO 43Ø
B6 31Ø HOME :F = Ø: RESTORE : IF
        DE = Ø THEN PRINT D$;"PR
       #1": PRINT I$;"8ØN"
C2 32Ø GOSUB 48Ø: IF F = 1 THEN
       34Ø
34 33Ø GOSUB 5ØØ: IF F = Ø THEN
       32Ø
BD 34Ø IF DE = Ø THEN PRINT D$;"
       PR#Ø"
1D 35Ø GOTO 45Ø
97 36Ø END
07 37Ø HOME : PRINT "TYPE "LABEL
       $(C): INPUT INP$:L = LEN
       (INP$)
9B 38Ø PRINT :Q = Ø:F = Ø: RESTO
       RE : IF DE = Ø THEN PRINT
        D$;"PR#1": PRINT I$;"8ØN
       "
B0 39Ø GOSUB 48Ø: IF F = 1 THEN
       43Ø
32 4ØØ IF LEFT$ (A$(C),L) < > IN
       P$ THEN 42Ø
90 41Ø Q = 1: GOSUB 5ØØ
CA 42Ø IF F = Ø THEN 39Ø
BC 43Ø IF DE = Ø THEN PRINT D$;"
       PR#Ø"
05 44Ø IF Q = Ø THEN PRINT : PRI
       NT G$;: INVERSE : PRINT "
       NO MATCHES FOUND": NORMAL
15 45Ø PRINT : PRINT G$;: INVERS
       E : PRINT "PRESS ANY KEY"
       : NORMAL : POKE - 16368,Ø
3C 46Ø A = PEEK ( - 16384): IF A
        < 128 THEN 46Ø
45 47Ø POKE - 16368,Ø: GOTO 7Ø
7B 48Ø READ A$(1),A$(2),A$(3),A$
       (4),A$(5),A$(6): IF A$(1)
        = "END" THEN F = 1
26 49Ø RETURN
5E 5ØØ PRINT A$(1)"    "A$(2)"
       "A$(3)"    "A$(4)"    P. "A
       $(5)"    "A$(6): PRINT
83 51Ø A = PEEK ( - 16384): IF A
        < 128 THEN RETURN
54 52Ø A = PEEK ( - 16384): IF A
        < 128 THEN 52Ø
54 53Ø A$ = CHR$ (A - 128): POKE
        - 16368,Ø: IF A$ < > "P"
       THEN 52Ø
1D 54Ø RETURN
```

Program 4: IBM PC/PCjr Fast Filer

Version by Patrick Parrish, Programming Supervisor

Please refer to "COMPUTE!'s Guide to Typing In Programs" before entering this listing.

```
ID 1Ø WIDTH 4Ø:KEY OFF:DEF SEG=Ø
      :POKE 1Ø47,PEEK(1Ø47) OR 6
      4:DIM A$(6):G$="
       "
NL 2Ø CLS:LOCATE 1Ø,3,Ø:PRINT "P
      RINT TO SCREEN OR PRINTER
      (S/P)?"
KB 3Ø K$=INKEY$:IF K$="" OR (K$<
      >"P" AND K$<>"S") THEN 3Ø
GI 4Ø DE=-(K$="S"):IF DE=1 THEN
      OPEN "SCRN:" FOR OUTPUT AS
       #1 ELSE OPEN "LPT1:" FOR
      OUTPUT AS #1
CM 5Ø LABEL$(1)="MAGAZINE TITLE:
      ":LABEL$(2)="AUTHOR'S LAST
       NAME:"
DJ 6Ø LABEL$(3)="THE TARGET SUBJ
      ECT:":LABEL$(4)="DATE (IE.
      , 1/14/85 OR 1/85):"
OL 7Ø CLS:PRINT STRING$(6,31)G$"
      CHOOSE ONE (1-8):":PRINT:P
      RINT G$" 1. MAGAZINE"
JP 8Ø PRINT G$" 2. AUTHOR":PRINT
       G$" 3. SUBJECT":PRINT G$"
       4. DATE"
PB 9Ø PRINT G$" 5. AND":PRINT G$
      " 6. OR":PRINT G$" 7. PRIN
      T ALL":PRINT G$" 8. QUIT"
LP 1ØØ K$=INKEY$:IF K$="" OR (VA
       L(K$)<1 OR VAL(K$)>8) THE
       N 1ØØ
PE 11Ø K=VAL(K$):ON K GOTO 12Ø,1
       3Ø,14Ø,15Ø,16Ø,16Ø,29Ø,32
       Ø
NF 12Ø C=1:GOTO 33Ø
OO 13Ø C=2:GOTO 33Ø
OH 14Ø C=3:GOTO 33Ø
PA 15Ø C=4:GOTO 33Ø
FN 16Ø H$="OR":IF K=5 THEN H$="A
       ND"
KB 17Ø PRINT:PRINT G$"# "H$" # (
       1-4):":PRINT G$;:INPUT N1
       ,N2
MB 18Ø IF (N1<1 OR N1>4) OR (N2<
       1 OR N2>4) THEN 17Ø
GI 19Ø CLS:PRINT "TYPE "LABEL$(N
       1):INPUT I1$:L=LEN(I1$)
EE 2ØØ PRINT:PRINT "TYPE "LABEL$
       (N2):INPUT I2$:L2=LEN(I2$
       )
PP 21Ø PRINT:Q=Ø:F=Ø:RESTORE
FD 22Ø GOSUB 42Ø:IF F=1 THEN 39Ø
CO 23Ø IF K=6 THEN 25Ø
PI 24Ø IF LEFT$(A$(N1),L)<>I1$ O
       R LEFT$(A$(N2),L2)<>I2$ T
       HEN 27Ø ELSE 26Ø
PL 25Ø IF LEFT$(A$(N1),L)<>I1$ A
       ND LEFT$(A$(N2),L2)<>I2$
       THEN 27Ø
JA 26Ø Q=1:GOSUB 44Ø
JM 27Ø IF F=Ø THEN 22Ø
IE 28Ø GOTO 39Ø
LP 29Ø CLS:F=Ø:RESTORE
HP 3ØØ GOSUB 42Ø:IF F=1 THEN 4ØØ
JC 31Ø GOSUB 44Ø:IF F=Ø THEN 3ØØ
        ELSE 4ØØ
ID 32Ø CLOSE #1:END
AH 33Ø CLS:PRINT "TYPE "LABEL$(C
       ):INPUT IN$:L=LEN(IN$)
PG 34Ø PRINT:Q=Ø:F=Ø:RESTORE
GK 35Ø GOSUB 42Ø:IF F=1 THEN 39Ø
EI 36Ø IF LEFT$(A$(C),L)<>IN$ TH
       EN 38Ø
JD 37Ø Q=1:GOSUB 44Ø
NG 38Ø IF F=Ø THEN 35Ø
EL 39Ø IF Q=Ø THEN PRINT:PRINT G
       $;:COLOR Ø,7:PRINT"NO MAT
       CHES FOUND":COLOR 7,Ø:PRI
       NT
IH 4ØØ PRINT " "G$;:COLOR Ø,7:PR
       INT "PRESS ANY KEY":COLOR
        7,Ø
CO 41Ø A$=INKEY$:IF A$="" THEN 4
       1Ø ELSE 7Ø
AG 42Ø READ A$(1),A$(2),A$(3),A$
       (4),A$(5),A$(6):IF A$(1)=
       "END" THEN F=1
ME 43Ø RETURN
OM 44Ø PRINT #1,A$(1)"    "A$(2)"
           "A$(3)"    "A$(4)"    P.
        "A$(5)"    "A$(6):PRINT #
       1,
LK 45Ø A$=INKEY$:IF A$="" THEN R
       ETURN
OC 46Ø A$=INKEY$:IF A$="" OR A$<
       >"P" THEN 46Ø
NM 47Ø RETURN
```

Program 5: TI Fast Filer

Version by Patrick Parrish, Programming Supervisor

```
1Ø  DIM A$(6)
2Ø  G$="{5 SPACES}"
3Ø  CALL CLEAR
4Ø  PRINT "PRINT TO SCREEN O
    R PRINTER (S/P)?"
5Ø  CALL KEY(Ø,K,S)
6Ø  IF S=Ø THEN 5Ø
7Ø  IF (K<>8Ø)*(K<>83)THEN 5
    Ø
8Ø  DE=-(K=8Ø)
9Ø  IF DE=Ø THEN 11Ø
1ØØ OPEN #DE:"RS232/2.BA=96
    ØØ.DA=8.PA=N"
11Ø LABEL$(1)="MAGAZINE TIT
    LE:"
12Ø LABEL$(2)="AUTHOR'S LAS
    T NAME:"
13Ø LABEL$(3)="THE TARGET S
    UBJECT:"
14Ø LABEL$(4)="DATE (IE., 1
    /14/85 OR 1/85):"
15Ø CALL CLEAR
16Ø PRINT
17Ø PRINT G$;"CHOOSE ONE (1
    -8):"::
18Ø PRINT G$;" 1. MAGAZINE"
19Ø PRINT G$;" 2. AUTHOR"
2ØØ PRINT G$;" 3. SUBJECT"
21Ø PRINT G$;" 4. DATE"
22Ø PRINT G$;" 5. AND"
23Ø PRINT G$;" 6. OR"
24Ø PRINT G$;" 7. PRINT ALL
    "
25Ø PRINT G$;" 8. QUIT":::
    ::::
26Ø CALL KEY(Ø,K,S)
27Ø IF S=Ø THEN 26Ø
28Ø IF (K<49)+(K>56)THEN 26
    Ø
29Ø K=K-48
3ØØ ON K GOTO 31Ø,33Ø,35Ø,3
    7Ø,39Ø,39Ø,68Ø,76Ø
31Ø C=1
32Ø GOTO 79Ø
33Ø C=2
```

```
34
35
36
37
38
39
4Ø
41
42
43
44
45
46
47
48
49
5Ø
51
52
53
54
55Ø
56Ø
57Ø
58Ø
59Ø
6ØØ
61Ø

62Ø
63Ø

64Ø
65Ø
66Ø
67Ø
68Ø
69Ø
7ØØ
71Ø
72Ø
73Ø
74Ø
75Ø
76Ø
77Ø
78Ø
79Ø
8ØØ
81Ø
82Ø
83Ø
84Ø
85Ø
86Ø
87Ø
88Ø
89Ø
9ØØ
91Ø
92Ø
93Ø
94Ø
95Ø
96Ø
97Ø
98Ø
99Ø
1ØØØ
1Ø1Ø

1Ø2Ø
1Ø3Ø
1Ø4Ø
1Ø5Ø


1Ø6Ø
1Ø7Ø
1Ø8Ø
1Ø9Ø
11ØØ
111Ø
```

```
340 GOTO 790
350 C=3
360 GOTO 790
370 C=4
380 GOTO 790
390 H$="OR"
400 IF K<>5 THEN 420
410 H$="AND"
420 PRINT G$;"# ";H$;" # (1-4):"
430 PRINT G$;
440 INPUT N1,N2
450 IF ((N1<1)+(N1>4))+((N2<1)+(N2>4))THEN
    420
460 CALL CLEAR
470 PRINT "TYPE ";LABEL$(N1)
480 INPUT I1$
490 L=LEN(I1$)
500 PRINT
510 PRINT "TYPE ";LABEL$(N2)
520 INPUT I2$
530 L2=LEN(I2$)
540 Q=0
550 PRINT
560 F=0
570 RESTORE
580 GOSUB 1010
590 IF F=1 THEN 930
600 IF K=6 THEN 630
610 IF (SFG$(A$(N1),1,L)<>I1$)+(SEG$(A$(N2
    ),1,L2)<>I2$)THEN 660
620 GOTO 640
630 IF (SEG$(A$(N1),1,L)<>I1$)*(SEG$(A$(N2
    ),1,L2)<>I2$)THEN 660
640 Q=1
650 GOSUB 1050
660 IF F=0 THEN 580
670 GOTO 930
680 CALL CLEAR
690 F=0
700 RESTORE
710 GOSUB 1010
720 IF F=1 THEN 960
730 GOSUB 1050
740 IF F=0 THEN 710
750 GOTO 960
760 IF DE=0 THEN 780
770 CLOSE #DE
780 END
790 CALL CLEAR
800 PRINT "TYPE ";LABEL$(C)
810 INPUT INP$
820 L=LEN(INP$)
830 Q-0
840 PRINT
850 F=0
860 RESTORE
870 GOSUB 1010
880 IF F=1 THEN 930
890 IF SEG$(A$(C),1,L)<>INP$ THEN 920
900 Q=1
910 GOSUB 1050
920 IF F=0 THEN 870
930 IF Q=1 THEN 960
940 PRINT
950 PRINT G$;"NO MATCHES FOUND"
960 PRINT
970 PRINT G$;" PRESS ANY KEY"
980 CALL KEY(0,D,S)
990 IF S=0 THEN 980
1000 GOTO 150
1010 READ A$(1),A$(2),A$(3),A$(4),A$(5),A$
     (6)
1020 IF A$(1)<>"END" THEN 1040
1030 F=1
1040 RETURN
1050 PRINT #DE:A$(1);"{3 SPACES}";A$(2);"
     {3 SPACES}";A$(3);"{3 SPACES}";A$(4);
     "{3 SPACES}P. ";A$(5);"{3 SPACES}";A$
     (6)
1060 PRINT #DE
1070 CALL KEY(0,D,S)
1080 IF S=0 THEN 1110
1090 CALL KEY(0,D,S)
1100 IF (S=0)+(D<>80)THEN 1090
1110 RETURN
```

The Beginners Page

Tom R. Halfhill

For-Next: Part 3

In the past two columns, we've discussed how to construct FOR-NEXT loops and apply them in practical ways to automate repetitive tasks. But you might be surprised to learn that another very common use of FOR-NEXT is to make a loop which does absolutely nothing.

Sounds crazy, doesn't it? Why would anybody write a routine which does nothing? Okay, so I exaggerated a little. These kind of loops don't do *absolutely* nothing. They just make the computer idle in neutral for a few moments.

Suppose your program needs to pause for a brief period. Perhaps it is displaying a title screen, or printing instructions that are scrolling off the screen too fast for people to read. One answer is a *delay loop*:

```
10 FOR X=1 TO 1000
20 NEXT X
```

You simply insert this loop wherever you want the delay. The computer spins its wheels for 1,000 passes and then carries on. By changing the size of the loop, you can force a delay for a fraction of a second or a minute or more.

Loops Within Loops

As yet another example of the flexibility of FOR-NEXT, you can also put a loop within a loop, or even a loop within a loop within a loop within a loop. These are called *nested loops*. However, there's a rule you have to follow to avoid confusing your computer (not to mention yourself). Each related FOR and NEXT must be completely contained within the loop immediately surrounding it:

```
10 FOR X-1 TO 10
20 PRINT
30 PRINT "OUTER LOOP #";X
40 PRINT
50 FOR Y=1 TO 10
60 PRINT "INNER LOOP #";Y
70 FOR Z=1 TO 200
80 REM DELAY LOOP
90 NEXT Z
100 NEXT Y
110 NEXT X
```

This program looks so unusual that the best way to understand what's going on is simply to run it. The PRINT statements will tell you which loop is executing during each pass; notice how the inner loop executes ten times for each pass of the outer loop.

The key point here is the NEXT statements at lines 90, 100, and 110. Observe how the Z delay loop is nested completely within the Y loop, which in turn is nested completely within the X loop. If you change the order of the NEXT statements, the program won't work.

Nested loops are handy when you need to insert a delay loop within a larger loop that is doing something else too quickly. Another application is embedding a smaller repetitive routine within a larger repetitive routine. For instance, let's take another look at the checkbook routine in last month's column. It adds up all the checks written in a month:

```
10 PRINT "HOW MANY CHECKS
   THIS MONTH";
20 INPUT CH
30 FOR X=1 TO CH
40 PRINT "AMOUNT OF CHECK";
50 INPUT AM
60 SUM=SUM+AM
70 NEXT X
80 PRINT "TOTAL AMOUNT IS
   $";SUM
```

To make this routine sum up all the checks written in a year, you could simply surround it with a FOR-NEXT loop that performs 12 passes. Add or change these lines:

```
5 FOR Z=1 TO 12
8 SUM=0
80 PRINT "AMOUNT FOR MONTH
   ";Z;" IS $";SUM
90 YR=YR+SUM
100 NEXT Z
110 PRINT "TOTAL FOR YEAR IS
   $";YR
```

Line 5 begins the outer loop. Line 8 is necessary to clear out the value of SUM for each monthly calculation. Line 80 prints the total amount for each month. Line 90 creates a new variable, YR, to keep a running total of the yearly amount. Line 100 repeats the outer loop. And line 110 prints the final total of all the checks written during the year.

Next month we'll continue our discussion of FOR-NEXT by showing how to make long-legged loops and even *backward* loops. ©

Programming the TI

C. Regena

Using TI Logo II

Logo is probably the most popular second language on TI computers (after BASIC). That's because *TI Logo* is a versatile, entertaining language, especially for young people. If you liked *TI Logo*, you'll like *TI Logo II* even better.

TI Logo II adds several improvements. It's compatible with any kind of printer, thermal or RS-232 (the old Logo could only use the thermal printer). *TI Logo II* uses sprites as before, but now you can make them big or small. It also has three-channel music capabilities.

You can save both procedures and custom characters on cassette or disk. *TI Logo II* requires the 32K Memory Expansion.

TI Logo II comes with a large loose-leaf binder, a 200-page manual, a *TI Sampler* booklet of procedures and educational activities, a sample disk in a vinyl loose-leaf holder, a sample cassette, and the actual *Logo* cartridge in a plastic holder that fits in the binder.

The manual has been revised since the first version of *TI Logo*. It is well illustrated and has plenty of sample procedures. Chapter 12 is a glossary of Logo Primitive Commands. If you are familiar with other versions of Logo and just need a quick reference guide, this section is a big help. There is also a one-page Keyboard Reference Guide which summarizes the key functions. If you need the details of a command, there is a comprehensive index.

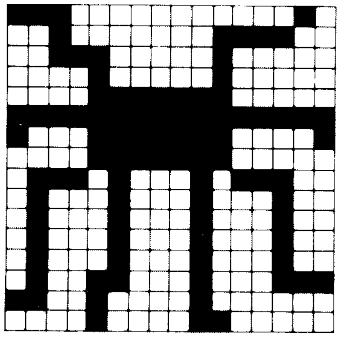
Versatile Sprite Commands

Sprites (smoothly moving screen objects) are a feature of *TI Logo* that children really enjoy. Thirty-two sprites are available. You can use the five predefined shapes, or you can create your own with MAKESHAPE. You may SETCOLOR (16 colors to choose from) and SETHEADING to position the sprite. You can move certain distances with the regular turtle commands such as FORWARD and LEFT. Or you can SETSPEED, which continuously moves an object. And you can FREEZE and THAW sprites. *Logo II* also has the commands BIG and SMALL. SMALL is the original sprite size; BIG makes them twice as large. This means greater fun with sprite animation.

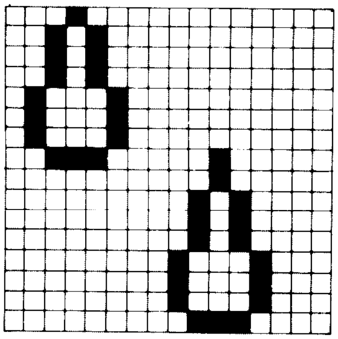
Music is the main reason I bought *TI Logo II*. Like regular *TI BASIC*, there are three voices available plus a noise generator and drum. You can execute other commands while music is playing, which enables you to synchronize the music with animated graphics.

The MUSIC command consists of numbered notes and numbers which represent the rhythm or duration of the notes. You may also SETTEMPO and choose between STACCATO and LEGATO. PM or PLAYMUSIC will play the music you have put in the music buffer with either the MUSIC command or a series of NOTE commands. You may also SET VOLUME, and of course add a DRUM rhythm accompaniment. The manual gives detailed instructions on how to use MUSIC.

The main disappointment I had with *TI Logo II* was that the sample programs that came on the disk and cassette were created with *Logo I* and did not exploit the music features. Following is a program which illustrates music and sprite animation.



MAKESHAPE 6



MAKESHAPE 7

Custom Shapes

First you want to define some custom shapes. Type MAKESHAPE 6 and copy the spider pattern. Next MAKESHAPE 7 for the other pattern. When the grid appears, use the arrow keys to move around. When you want to blacken a square, press the FCTN key as you move.

Now type in the following procedures. Feel free to use the two-letter abbreviations. SS is SETSPEED and designates how fast a sprite will go. SH is SETHEADING and points the sprite in a certain direction. CARRY indicates which shape number the sprite should be. SC is SETCOLOR for the color of the sprite. PU and PD are PENUP and PENDOWN.

To run the program, type SPIDER and press ENTER (in command mode). CS clears the screen and CM clears the music buffer. WAIT is a command to wait a certain time before executing more statements in the procedure. I used WAIT to help choreograph the graphics with the music. If you get the message OUT OF INK while the web is being drawn, simply type SPIDER again and ENTER. You may save your characters and procedures on disk or tape. To reload it later, use the command RECALL. To get a hard copy listing, use PRINTOUT.

A Few Answers

I'm going to take just a little room this month to answer some general questions I'm asked quite often. Yes, Texas Instruments still services their home computers, even though they are no longer selling them. Call 1-800-TI-CARES or write to the Lubbock, Texas address listed in your user's manual.

I still see *lots* of third-party software available. If you registered your computer when you purchased it, you should be on various mailing lists. Of course, it's a good idea to keep in touch with a user group for ongoing information (and there are still many strong user groups). Most of us who have written for the TI still love the computer and will probably continue to write for it forever.

Add-on hardware for the TI is still being developed and produced. CorComp has developed several kinds of peripherals, along with a number of other companies.

And, of course, all the COMPUTE! books for the TI are still available. If you don't see them in the larger bookstores, you may order directly from COMPUTE!.

Thanks to all of you who wrote to ask if the algebra tutorial published for the TI in the October 1984 COMPUTE! is available for the Commodore 64. The answer is now yes. I do write for a variety of computers, but this column describes the TI version only.

Spider For *TI Logo II*

```
TO CM
SETVOICE Ø
SETVOICE 1
END

TO TUNE1
MUSIC [Ø 5 5 5 7 9 9][1 2 1 2 1
  2 2]
MUSIC [5 7 5 7 9 5][1 2 1 2 1 4]
END

TO TUNE2
MUSIC [9 9 1Ø 12 12 1Ø 9 1Ø 12 9
  ][2 2 1 2 2 2 1 2 1 4]
END

TO SUN
TELL SPRITE 9
CARRY :BALL
SS Ø SH Ø FD 8Ø LEFT 9Ø FD 1ØØ
SC 1Ø
END

TO WEB
REPEAT 8 [FORWARD 45 BACK 45 RIG
  HT 45]
HOME FORWARD 2Ø
RIGHT 12Ø FORWARD 15
REPEAT 3 [RT 3Ø FD 15 RT 6Ø FD 15]
RT 3Ø FD 15
PU HOME FORWARD 25 PD
RIGHT 12Ø FORWARD 2Ø
REPEAT 3 [RT 3Ø FD 19 RT 6Ø FD 19]
RT 3Ø FD 2Ø
PU HOME FD 32 PD
RIGHT 12Ø FORWARD 24
REPEAT 3 [RT 3Ø FD 24 RT 6Ø FD 24]
RT 3Ø FD 25
END

TO SPIDER
CS CM
TELL :ALL
SC Ø
FREEZE
HOME
TELL TURTLE
HIDETURTLE
HOME
TUNE1 TUNE2 TUNE2 TUNE1 PM
WEB
TELL SPRITE 6
CARRY 6
SH 18Ø
FD 3Ø
SH Ø
SC 1
THAW
SS 1
TELL SPRITE 7 SS Ø HOME
CARRY 7 WAIT 2ØØ
SH Ø FD 8Ø
SH 18Ø SC 4 SS 1Ø
TELL SPRITE 8
CARRY 7 SS Ø HOME
SH Ø FD 6Ø LEFT 9Ø FD 15
SH 18Ø SC 4 SS 1Ø WAIT 6Ø
TELL SPRITE 6 SH 18Ø SS 1Ø
WAIT 6Ø FREEZE
TELL SPRITE 7 SC Ø
TELL SPRITE 8 SC Ø
WAIT 1ØØ
SUN WAIT 2ØØ
TELL SPRITE 6 SH Ø SS 2 THAW
END
```

©

Advertisers Index

Classified

PHONEMARK DATASETTES: C64, Plus4, Vic20. $25.00 plus shipping. We are direct importers. American Mercantile Co., 2450 1st Ave. South, Seattle, WA 98134 (206)624-6141.

T.I. CHECKWRITER Letter code and dollar amount are all that's necessary to print a complete check. Provides yearly totals for taxes. Disk $24.95. R.I.P. Software 248 Purisima Rd., Woodside CA 94062.

SCIENCE SOFTWARE with graphics for the Commodore 64. Astronomy programs, etc. Catalog and demo disk $3. David Eagle, 7952 W. Quarto Dr., Littleton, CO 80123

SAFEGUARD YOUR COMPUTER KEYBOARD from contaminants with OVERBYTE molded keyboard covers. $24.95 xparent & durable! Call (213)866-2583/send $2 further info. OVERBYTE-P.O. Box 10652, Burbank, CA 91510.

SOFTWARE PROTECTOR 64. Copy and list protect your valuable Basic programs. Only $10 (disk only). Systems Software 44 Rte 25A #209, Smithtown, NY 11787.

PINOCHLE for Commodore 64, PC and PCjr. Real, double-deck, partnership pinochle gives you a partner and two opponents! On disk, $25 postpaid. Jim Bernard, 8018 Sunset Path Ct., Springfield, VA 22153.

LOWEST DISK PRICES-SS/DD with sleeve & label-10/$8.80, bulk 100/$78. Ds/DD with sleeve & label-10/$10.90, bulk 100/$99. Prime quality major mfr's overstock! Money-back satisfaction guarantee! Min order $15. Pay by MC/Visa/AE, UNITECH 20 Hurley St. Cambridge, MA 02141 (800)343-0472, in Mass (617)864-8324.

HELP IS ON THE WAY!
Just call 1-800-334-0868 to get your free copy of the latest COMPUTE! Books Catalog! If you need help in getting information on all of the latest COMPUTE! book titles available plus all COMPUTE! backlist titles, call us today!

$$ Money Maker $$
Increase your income by hundreds and even thousands of dollars. Money back guarantee! (Apple, Macintosh, IBM) Free brochure: 1-800-223-5838 (305-771-6498).

ATARI USERS. 4 games on cassette, $25. Saucers + monsters, poker, sea finder, pinball. Write or call Keith Anderson, 8435 12th Ave., Silver Spring, MD 20903 (301)434-0285.

DISK SERVICE MANUAL. Service floppies without special software or equipment. FREE information. Consumertronics-DSM, PO Drawer 537, Alamogordo, NM 88310.

Max MD1, $1.39-MD2, $1.99. Dys 104/1D, $1.79 -104/2D, $2.39. shipping $3.75. also Verbatim, IBM, 3M, BASF. TAPE WORLD, 220 Spring St., Butler, PA 16001, 1-800-245-6000, Visa, MC.

REPLACEMENT POWER SUPPLIES FOR C-64 or VIC-20 $29 ea. plus $3 shipping. 64 parts list $5–refundable w/1st order. AA Computer, 2726 Park St., JAX, FL 32205.

FREE! SHOP BY MODEM in our revolutionary electronic shopping mall. We even accept credit cards! We have products YOU want! Call 1-800-840-8066 with your modem NOW!

BUY-SELL-TRADE computers, hardware, software. Announce clubs, BBS's etc. Send for free sample of Micro-Swap, the computer classifieds. P.O. Box 24, Esmond, Il 60129.

FIXED ASSET DEPRECIATION on C64 Handles acrs and pre-acrs methods. $39.95. For free info write: MPM Software Prods., P.O. Box 3522, Glendale, CA 91201.

COMPUTER OWNERS!!! Earn dollars and save money with your computer. Write C&D Associates, Box 851, Mt. Prospect, IL 60056 Specify computer type. No Investment!!

TI-99/4A SOFTWARE/HARDWARE bargains. Hard-to-find items. Huge selection. Fast service. Free catalog. D.E.C. Box 690, Hicksville, NY 11801.