COMPUTE! Interviews Gerrard O'Neill

COMPUTE!

$2.95
August
1984
Issue 51
Vol. 6, No. 8
£2.25 UK $3.75 Canada
02193
ISSN0194-347X

The Leading Magazine Of Home, Educational, And Recreational Computing

The Consumer Electronics Show: New Excitement In Home Computing

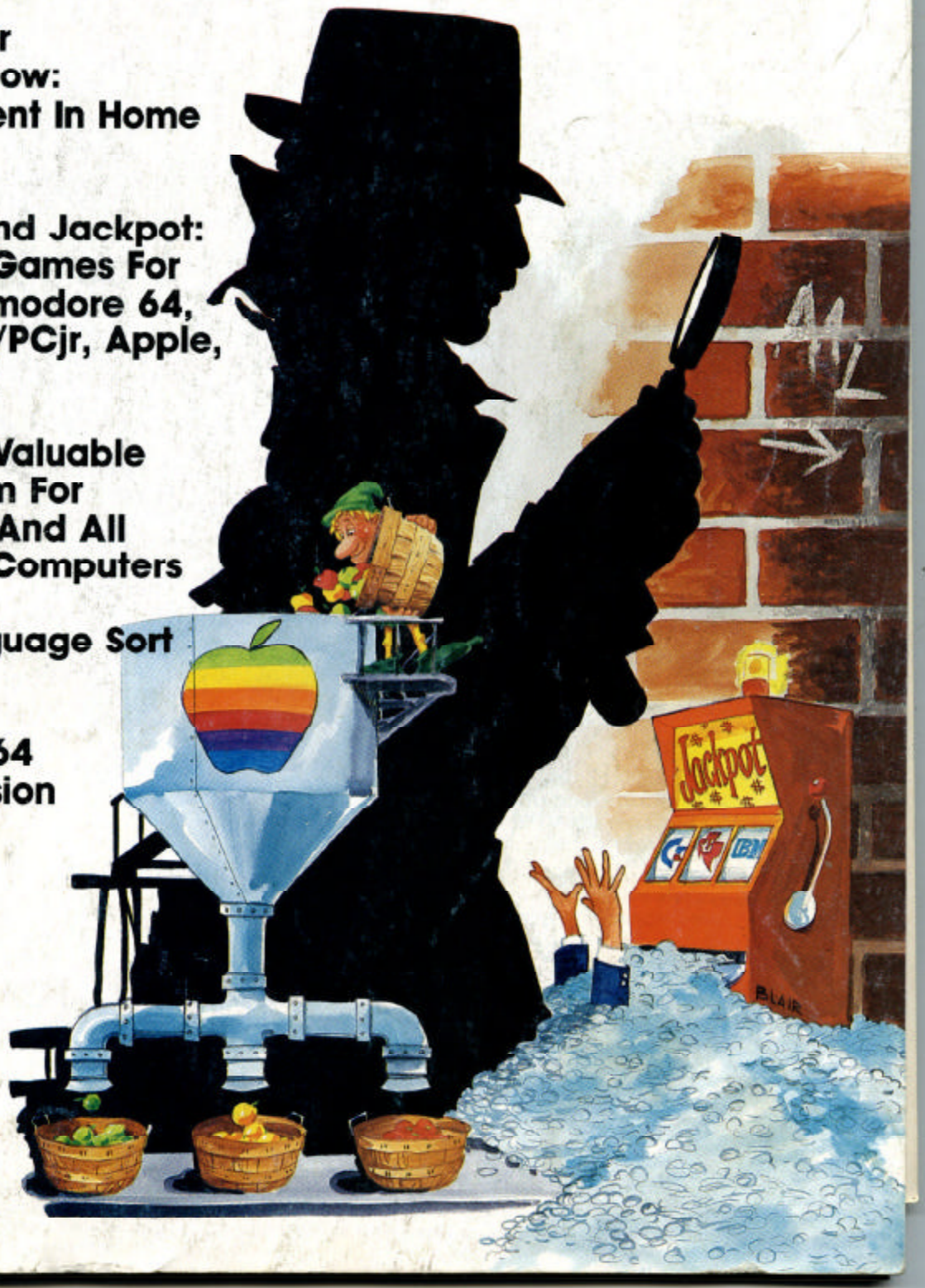
Devastator And Jackpot: Two Exciting Games For VIC-20, Commodore 64, Atari, IBM PC/PCjr, Apple, And TI-99/4A

ML Tracer: A Valuable Utility Program For Atari, Apple, And All Commodore Computers

Machine Language Sort For Apple

Commodore 64 Error Suppression

And More



0 71486 02193 08

August 1984 Vol. 6, No. 8

FEATURES

32 Software Power!: The Summer Consumer Electronics Show ..... Selby Bateman
42 The COMPUTE! Interview: Gerard K. O'Neill ..... Selby Bateman

EDUCATION AND RECREATION

58 Devastator ..... David R. Arnold
83 Jackpot ..... Rick Rothstein

REVIEWS

96 The *Complete Personal Accountant* For The Commodore 64 ..... Richard DeVore
98 *Star League Baseball* ..... Shay Addams

COLUMNS AND DEPARTMENTS

6 The Editor's Notes ..... Robert Lock
10 Readers' Feedback ..... The Editors and Readers of COMPUTE!
24 Computers And Society: Computer Assisted Explorations With Music ..... David D. Thornburg
28 The Beginner's Page: Printing And Asking ..... Robert Alonso
102 INSIGHT: Atari ..... Bill Wilkinson
115 Programming The TI: The Singing Computer ..... C. Regena
120 Machine Language: Decimal Mode, Part 2 ..... Jim Butterfield
122 64 Explorer ..... Larry Isaacs
126 On The Road With Fred D'Ignazio: Are Computers A Home Appliance? ..... Fred D'Ignazio
134 Questions Beginners Ask ..... Tom R. Halfhill

THE JOURNAL

106 ML Tracer ..... Thomas G. Gordon
119 64 Searcher ..... John Krause and Michael Jacobi
128 64 Error Suppression ..... Tom Nuss
130 Hi-Res VIC Drawing ..... Jeff Wise
132 ML Applesort ..... Richard Salley

54 The Automatic Proofreader For VIC, 64, And Atari
56 A Beginner's Guide To Typing In Programs
57 How To Type COMPUTE!'s Programs
135 News & Products
138 MLX: Machine Language Entry Program For Commodore 64
140 CAPUTE! Modifications Or Corrections To Previous Articles
141 Product Mart
144 Advertisers Index

NOTE: See page 57 before typing in programs.

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

GUIDE TO ARTICLES AND PROGRAMS

The Automatic Proofreader For VIC, 64, And Atari	*
A Beginner's Guide To Typing In Programs	*
How To Type COMPUTE!'s Programs	
Devastator	V/64/TI/AP/PC/PCjr
Jackpot	TI/64/V/AT/PC/PCjr
The *Complete Personal Accountant* For The Commodore 64	64
Star League Baseball	AT/64
The Editor's Notes	*
Readers' Feedback	*
Computers And Society: Computer Assisted Explorations With Music	*
The Beginner's Page: Printing And Asking	*
INSIGHT: Atari	AT
Programming The TI: The Singing Computer	TI
Machine Language: Decimal Mode, Part 2	*
64 Explorer	64
On The Road With Fred D'Ignazio: Are Computers A Home Appliance?	*
Questions Beginners Ask	*
ML Tracer	P/V/64/AT/AP
64 Searcher	64
64 Error Suppression	64
Hi-Res VIC Drawing	V
ML Applesort	AP
News & Products	
MLX: Machine Language Entry Program For Commodore 64	
CAPUTE! Modifications Or Corrections To Previous Articles	
Product Mart	
Advertisers Index	

AP Apple **AT** Atari, **P** PET/CBM, **V** VIC-20, **C** Radio Shack Color Computer, **64** Commodore 64, **TS** Timex/Sinclair, **TI** Texas Instruments, **PCjr** IBM PCjr, **PC** IBM PC, **AD** Coleco Adam. *All or several of the above.

COMPUTE! Publications, Inc. abc
One of the ABC Publishing Companies

One of the ABC Publishing Companies:
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019

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.

Publisher	Gary R. Ingersoll
Editor in Chief	Robert C. Lock
Director of Administration	Alice S. Wolfe
Senior Editor	Richard Mansfield
Managing Editor	Kathleen E. Martinek
Production Director	Tony Roberts
Production Editor	Gail Walker
Editor, COMPUTE!'s PC & PCjr Magazine	Tom R. Halfhill
Editor, COMPUTE!'s GAZETTE	Lance Elko
Technical Editor	Ottis R. Cowper
Assistant Technical Editors	John Krause, George Miller
Program Editor	Charles Brannon
Features Editor	Selby Bateman
Assistant Editors	Dan Carmichael, Robert Sims, Todd Heimarck, J. Blake Lambert, Robert Alonso
Editorial Assistant	Kathy Yakal
Research Assistant	Sharon Darling
Programming Supervisor	Patrick Parrish
Assistant Programming Supervisor	Gregg Peele
Editorial Programmers	Jeff Hamdani, Kevin Martin, Chris Poer, Tim Victor, Kevin Mykytyn
Programming Assistants	Mark Tuttle, David Florance
Copy Editors	Juanita Lewis, Joan Rouleau, Ann Davies
Proofreaders	Ethel Silver, Dwight Smith, Karen Uhlendorf, Marty Selby
Administrative Assistants	Vicki Jennings, Julie Fleming, Susan Young, Iris Brooks, Jan Kretlow
Associate Editors	Jim Butterfield, Toronto, Canada Harvey Herman, Greensboro, NC Fred D'Ignazio, 2117 Carter Road, S.W., Roanoke, VA 24015 David Thornburg, P.O. Box 1317, Los Altos, CA 94022
Contributing Editor	Bill Wilkinson

COMPUTE!'s Book Division	
Editor	Stephen Levy
Assistant Editors	Gregg Keizer, Stephen Hudson
Assistant Managing Editor	Randall Fosner
Administrative Assistant	Laura MacFadden
Artists	Janice Fary, Debbie Bray
Director, Books Sales & Marketing	Steve Vayatzis
Assistant	Carol Dickerson

Production Manager	Irma Swain
Art & Design Director	Janice Fary
Assistant Editor, Art & Design	Lee Noel
Mechanical Art Supervisor	De Potter
Artists	Leslie Jessup, Cindy Mitchell
Typesetting	Terry Cash
Illustrator	Harry Blair

Director of Advertising Sales	Ken Woodard
Advertising Coordinator	Patti Williams
Assistant	Joyce Morgo
Advertising Accounts	Bonnie Valentino
Promotion Manager	Mindy K. Kutchai

Subscriber Services Supervisor	Patty Jones
Assistants	Chris Patty, Christine Gordon, Sharon Sebastian, Rosemarie Davis
Dealer Sales Supervisor	Fran Lyons
Assistants	Gail Jones, Sharon Minor, Rhonda Savage
Individual Order Supervisor	Dorothy Bogan
Assistants	Judy Taylor, Lisa Flaharty, Anita Roop, Debi Goforth, Jenna Nash, Elizabeth White, Sybil Agee, Mary Hunt, Gayle Benbow, Betty Atkins, Sandra Jenkins
Shipping & Receiving	Jim Coward, Larry O'Connor, Dal Rees, John B. McConnell, Eric Staley, Sam Parker, Eddie Rice, David Hensley, John Archibald, Mary Sprague (Mail Room Coordinator)

Data Processing Manager	Leon Stokes
Assistant	Chris Cain

Vice President, Finance & Planning	Paul J. Megliola
Director, Finance & Planning	R. Steven Vetter
Accountant	Robert L. Bean
Credit Manager	David F. Carpenter
Purchasing Manager	Gregg L. Smith
Financial Analyst	Karen K. Rogalski
Staff	Linda Miller, Doris Hall, Jill Pope, Anna Harris, Anne Ferguson, Pat Fuller, Tracey Hutchins, Susan Booth, Sybil Agee

Robert C. Lock, Chief Executive Officer
Gary R. Ingersoll, President
Paul J. Megliola, Vice President, Finance and Planning
Debi Nash, Executive Assistant
Cassandra Robinson, Assistant

Member ABC — Audit Bureau of Circulations

COMPUTE! Publications, Inc. publishes:
COMPUTE
COMPUTE!'s GAZETTE
COMPUTE! Books
COMPUTE!'s GAZETTE DISK

Corporate Office:
324 West Wendover Ave., Suite 200
Greensboro, NC 27408 USA

Mailing address: COMPUTE!
Post Office Box 5406
Greensboro, NC 27403 USA
Telephone: 919-275-9809

Subscription Orders

COMPUTE! Circulation Dept.
P.O. Box 914
Farmingdale, NY 11737

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

COMPUTE! Subscription Rates (12 Issue Year):

US	(one yr.) $24	Air	
	(two yrs.) $45	Europe, Australia	$42
	(three yrs.) $65	Middle East, Central America and North	
Canada and Foreign Surface Mail	$30	Africa	$52
		South America, South Africa, Far East	$72

Advertising Sales

1. New England
Al Dalton
David Fay
617-451-0822

2. Mid Atlantic
Sharon Brooks
Joe Porter
Kathy Hicks
215-646-5700
212-567-6717 (NY)

3. Southeast & Foreign
Harry Blair
919-275-9809

4. Midwest
Gordon Benson
312-362-1821

5. Northwest/Mountain/Texas
Phoebe Thompson
408-354-5553
Jerry Thompson
415-348-8222

6. Southwest
Ed Winchell
213-378-8361
JoAnn Sullivan
619-941-2313

Director of Advertising Sales
Ken Woodard

COMPUTE! Home Office 919-275-9809.

Address all advertising materials to:
Patti Williams
Advertising Production Coordinator
COMPUTE! Magazine
324 West Wendover Ave., Suite 200
Greensboro, NC 27408

GOSUB is pushed onto the stack. (Specifically, it's the address minus one byte.)

2. The branch to the subroutine is taken, and the subroutine is executed.

3. When the RETURN is encountered in the subroutine, the return address information is pulled off the stack, program control is returned to that point, and processing continues.

Basically, the same sequence of events is taken when using machine language. When a JSR is encountered, the return information is pushed onto the stack, the branch is taken, and when the RTS (ReTurn from Subroutine) is encountered, the information is pulled from the stack, and control is returned to that place in the program.

When using GOSUBs and JSRs, this stack activity is automatically performed by the computer.

However, you can push and pull stack information yourself. This can be done with the use of the PHA machine language instruction, which pushes the number in the Accumulator onto the stack, and PLA, which pulls a byte off the stack and places it in the Accumulator. Other stack commands available are PHP, which pushes the processor status onto the stack, and PLP, which pulls a byte from the stack and puts it into the status register.

Manipulating the stack can be tricky. However, if, after jumping to a subroutine, you wish to return somewhere else, you can pull the return information off the stack (placed there by the operating system), and replace it with your data using the PHA command.

The stack can also be used as a temporary storage place for data in machine language programming. Instead of storing information in zero page, or some other area, push it onto the stack. When it's needed again, pull it back off. But be careful, because the stack can hold only 256 bytes of information. Also if you RTS before PLAing the byte or bytes off the stack, the return address will be wrong.

TI Programs Vs. Data Files

I read somewhere that if a TI-99/4A program sets up a data file, the data file should be stored on a separate disk or cassette from the program. Why is that? It seems to me that the logical place for the data would be on the same disk or cassette as the program using it.

Florence Fischer

Files are not saved or loaded by name on a cassette, and the TI makes no distinction between data and program files. As a result, if you place a data file on a tape following a program file, you may have difficulty locating the data file (especially if your recorder lacks a moderately accurate counter). Also, if you place the data file prior to the program file on the tape, and later expand your data file, you may end up writing over the program file.

For these reasons, it is wise to keep your program and data files on separate cassettes (or on opposite sides of a single cassette). No such problems exist for disk files since programs are stored by name and are labeled as program, data, etc., on the disk.

Slowing Things Down On VIC, 64, Or PET/CBM

I found something very interesting while experimenting with my 64. While listing a program, I noticed that if you press the CTRL key the listing will slow down. Does this work on all Commodore computers? Is it supposed to do this?

Mike Merriman

Yes, it is. Pressing the CTRL key on the Commodore 64 or the VIC-20 will slow the listings, and some BASIC programs. On the older CBM (Commodore Business Machines) computers like the 8032, the PET, etc., pressing the 1 key will do the same thing. This is to allow you to read the listings more easily as they scroll by.

To see how this affects a BASIC program, type, enter, and RUN the following program. While it is running, press the CTRL key and see what happens:

```
10 PRINT"A":GOTO10
```

Z80 Atari XL?

I have an Atari 800 and I am thinking about moving on to a more sophisticated system like the Atari 1450XLD. I have heard that the 600XL and 800XL are much like the older 400/800 models, but how about the 1400XL and 1450XLD? Is the BASIC language different? I heard it has a Z80 microprocessor. Is all this true?

Alekos Couloumbis

The 600XL and 800XL computers are very much like the 400 and 800. The 600XL and 800XL are almost identical, except that the 600XL has 16K while the 800XL has 64K. There have been some enhancements to the operating system of the XL computers, making it different enough so that some 400/800 programs will not run on the XL computers. However, Atari has a Translator *disk available through its Customer Service that allows you to run 400/800 programs on your XL computer. The BASIC in all XL computers (except the late 1200XL) is built-in, and almost identical to the earlier Atari BASIC, except that the infamous keyboard lockup has been fixed and the exponentiation function has been improved.*

The Atari 600XL and 800XL are now in full

cleared when GRAPHICS 8 is set up. A portion of the previous picture may have been destroyed, though, if you have changed modes (such as from GRAPHICS 8 to GRAPHICS 0).

The reason the screen was sometimes cleared is in line 40. You PLOT and DRAWTO random X,Y coordinates, but also use the Y coordinate for the color number. COLOR also chops off the part of a number that is not used. In GRAPHICS 8, only COLOR 1 and COLOR 0 are valid, so that odd numbers count as COLOR 1, and even numbers work as COLOR 0. PLOT is the same as PRINTing the CHR$ value of the color at the screen X,Y position (try using COLOR and PLOT with GRAPHICS 0, 1, and 2 to see the effect). If, however, the color number is 125, it is interpreted as CHR$(125), which is the same as the code for clear screen (CTRL-CLEAR). So COLOR 125:PLOT x,y will clear the screen. Your program is interesting, but to get the intended effect, you should use a different variable for the color. For example:

```
25 C=INT(2*RND(0)).
40 COLOR C:DRAWTO X,Y
```

TI Synthesizer Update

In the March 1984 issue of COMPUTE!, reader Jim Pate suggested using CALL PEEK(−28672,SP) on the TI-99/4A to check if the Speech Synthesizer is attached. He said that if it were attached, SP would be 96. This was correct to an extent. Because the address −28672 is part of the speech read/write buffer, sometimes (like after a CALL SPGET or CALL SAY) a value of 96 will not be placed into SP. To avoid this problem, instead of:

```
IF SP=96 THEN CALL SAY("UHOH")
```

use this:

```
IF SP THEN CALL SAY("UHOH")
```

This way, the CALL SAY statement will execute as long as SP is not 0.

Mark Chance

Thank you for the clarification on this.

Hidden 64 RAM

I have been dabbling in machine language a bit, and have a question. I would like to know if it is possible to load machine language programs into the RAM that is underneath BASIC ROM. If it is, how do I go about switching out BASIC ROM to use the ML routines, and then switching BASIC back in?

Kenneth Cox

There is 16K (16,384 bytes) of hidden RAM in the 64. 8K can be found underneath BASIC ROM at 40960 to 49151, hex $A000–$BFFF, 8192 bytes, and 8K is under the Kernal at 57344 to 65535, hex $E000–$FFFF.

Switching either BASIC or Kernal ROM in or out to expose the available RAM underneath is done via memory location 1. Normally, there's a 55 in that location. Setting bit 0 here to a zero will switch out BASIC and expose the 8K block of RAM underneath. Setting bit 1 of memory location one to a 0 will switch out both BASIC and Kernal ROM, exposing a total of 16K of RAM.

Use this BASIC line to switch out BASIC ROM:

```
POKE 1,PEEK(1)AND254
```

To switch out both BASIC and the Kernal, use:

```
POKE 1,PEEK(1)AND253
```

When memory location 1 is set at its normal value of 55 (BASIC and Kernal ROM switched in), POKEing and PEEKing to this memory follows special rules. When you PEEK this memory, you will get the values of the BASIC or Kernal ROM, that is, PEEK (40960). However, POKEing this memory (POKE 40960,255) will automatically POKE the RAM underneath.

This makes placing programs into the hidden RAM easy. You can POKE in your machine language routines via a BASIC poker program, or simply load the programs from tape or disk.

File Structure On Atari

I have an Atari 800 and am trying to write a BASIC program to access records in a file. If I open a file with a 9 to append the file, it will use the entire sector to store the data. If I open the file with a 12, I can write to the entire sector, but eventually I will come up with an EOF (End Of File) error. Is there any way to get around this problem? Also, are there any good books (besides the *DOS manual*) on file and record structure for the Atari disk?

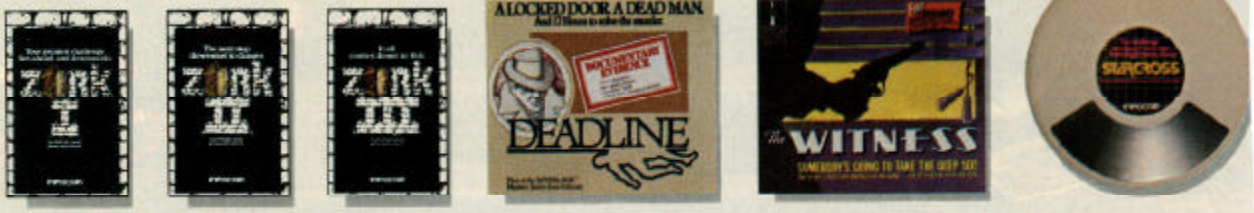
Charles Bentivegna

The OPEN command has four parameters:

```
OPEN IOCB#,access,aux,"filename"
```

IOCB# is a number from 1 to 7. There are eight Input/Output Control Blocks on the Atari. Each IOCB keeps track of an individual file. IOCB #0 is reserved for use by the screen editor (INPUT and PRINT). IOCB #7 is used for LPRINT, CSAVE, SAVE, LOAD, and CLOAD. When you OPEN a file to a particular IOCB, you use the same number when accessing the file with PRINT#IOCB; data or INPUT#IOCB, variable.

The second parameter, access, *is either 4 (OPEN for read), 8 (OPEN for write), 12 (OPEN for read and write), or 9 (OPEN to append). The* aux *byte is usually just 0. Access numbers 4 and 8 are straightforward. OPEN for read lets you GET or INPUT from that file, but not PRINT or PUT to it.*


zork I
zork II
zork III
A LOCKED DOOR, A DEAD MAN.
DEADLINE
the WITNESS
SUSPECT
SUSPICIOUS

THE INCOMPLETE WORKS OF INFOCOM, INC.

Incomplete, yes. But it's not just because we're always bringing out new stories in the Infocom interactive fiction collection. Nor is it simply due to the fact that with all the writing and re-writing, honing and perfecting that we put into every one of our stories, our work is seemingly never done.

The real reason is: an Infocom work of fiction can never be complete until you become a part of it.

You see, as hard as we work at perfecting our stories, we always leave out one essential element—the main character. And that's where you enter in.

Once you've got Infocom's interactive fiction in your computer, you experience something akin to waking up inside a novel. You find yourself at the center of an exciting plot that continually challenges you with surprising twists, unique characters (many of whom possess extraordinarily developed personalities), and original, logical, often hilarious puzzles. Communication is carried on in the same way as it is in a novel—in prose. And interaction is easy—you type in full English sentences.

But there is this key difference between our tales and conventional novels: Infocom's interactive fiction is active, not passive. The course of events is shaped by the actions you choose to take. And you enjoy enormous freedom in your choice of actions—you have hundreds, even thousands of alternatives at every step. In fact, an Infocom interactive story is roughly the length of a short novel in content, but because you're actively engaged in the plot, your adventure can last for weeks and months.

In other words, only you can complete the works of Infocom, Inc. Because they're stories that grow out of your imagination.

Find out what it's like to get inside a story. Get one from Infocom. Because with Infocom's interactive fiction, there's room for you on every disk.

INFOCOM™

Infocom, Inc., 55 Wheeler Street, Cambridge, MA 02138

For your: Apple II, Atari, Commodore 64, CP/M8™, DECmate, DEC Rainbow, DEC RT-11, IBM PC* and PCjr, KAYPRO II, MS-DOS 2.0*, NEC APC, NEC PC-8000, Osborne, Tandy 2000, TI Professional, TI 99/4A, TRS-80 Models I and III.

*Use the IBM PC version for your Compaq, and the MS-DOS 2.0 version for your Wang or Mindset.

DEVASTATOR

Robert Arnold

You and your comrades approach the hostile Devastator—a powerful mothership ready to destroy Earth. Out of nowhere, guardian ships attack. You have 30 seconds to destroy all of them—or else Earth is destroyed. Written for the unexpanded VIC, versions are also included for the 64, Color Computer, TI-99/4A, Apple II, and IBM PC and PCjr. Joystick required for all versions except VIC (optional).

"Devastator" is an action game where you must save Earth from aliens. What makes it different from similar games is that when you fail, Old Terra Firma is destroyed before your eyes.

You and your comrades are in one-man spaceships skimming the surface of a huge alien craft known as *Devastator*. Suddenly, out of nowhere, guardian ships appear, darting and dodging swiftly, causing havoc among your ranks. Blast them by lining up your cross hairs with the center of the spaceships and pressing the fire button. You have a mere 30 seconds to destroy ten ships before *Devastator* annihilates Earth with a death bolt.

The VIC Programs

This program is written in two parts because of the limited memory in an unexpanded VIC-20. Program 1 gives the instructions and customizes the characters. Be sure to save Program 1 before you run it. However, if you wish to view Program 1 before saving it, temporarily add the line 295 END. After you type in Program 2, save it with the name D. (For tape, be sure to save it immediately following Program 1.) Lines 305 and 310 of Program 1 will then cause Program 2 to load and run automatically.

The second program is the actual game. If you hit RUN/STOP and RESTORE anytime during the second program, you must type POKE 36869,255—no line number is needed—to play the game again. This is the location of the customized characters.

Devastator is played with a joystick simply for ease of use. However, if you want to use the keyboard, you can substitute the following lines in Program 2:

```
1000 IFPEEK(197)=17THENR=R-22
1005 IFPEEK(197)=33THENR=R+22
1010 IFPEEK(197)=28THENR=R-1
1015 IFPEEK(197)=36THENR=R+1
1110 POKEL+R,219:IFPEEK(197)<>32THEN1128
```

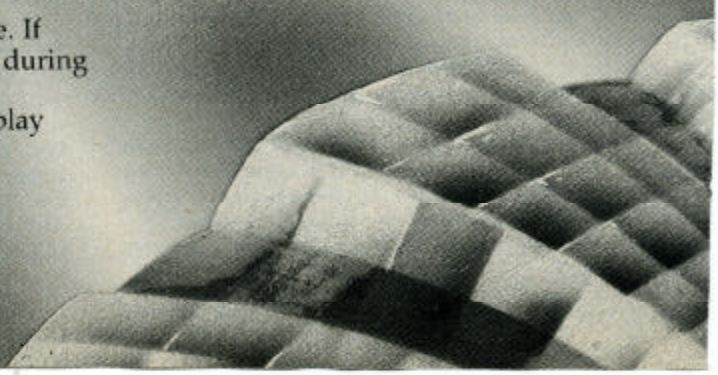
Delete lines 1016–1022.

The difficulty level of this game can be changed by subtracting or adding time in line 140, or by increasing or decreasing the number of points for ships hit (SC) in line 2000. (Each ship is worth ten points.) You can also make the ships harder to hit by changing the 9 in line 500 to a higher number.

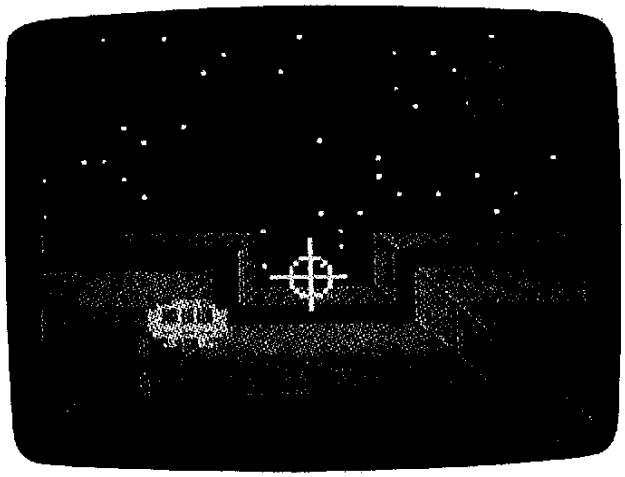
Here is an explanation of Program 2:

Line	
0	Variables.
20	Print Earth and stars.
70	Print first screen of Devastator.
160	Print second screen of Devastator.
250	Print third screen of Devastator.
350	Print fourth screen of Devastator.
500	Subroutine to print ships.
800	Subroutines for sound, joystick, and cross hairs.
1120	PEEK hit of a guardian ship.
1800	Subroutines for printing saucers.
2000	Decide win or loss.
2005	Routine for loss.
2040	"Play again" option.
3000	Routine for win.

Both of these programs use a lot of memor so don't add extra spaces.



```
580 Z=6:FORPS=1TO20:PRINT@PS-1,CHR$
    (128);:PRINT@PS+31,CHR$(128);:G
    OSUB2000:NEXT
585 SOUND 10,2
590 FOR PS=19 TO 1 STEP-1:PRINT@PS+
    6,CHR$(128);:PRINT@PS+38,CHR$(1
    28);:GOSUB2000:NEXT
595 SOUND 50,2:NEXT
597 FORI=0TO16:SOUND255,1:SET(14+I,
    3+I,5):SET(14+I+RND(2)-RND(2),3
    +I+RND(2)-RND(2),RND(8)):NEXT
599 FORI=1TO100:SET(25-RND(10)+RND(
    10),16-RND(10)+RND(10),RND(9)-1
    ):NEXT
600 FORI=1TO50:CLSRND(9)-1:NEXT:FOR
    I=255TO1STEP-17:SOUNDI,1:NEXT
610 PRINT"PLAY AGAIN, HUMANOID? (Y/
    N):";
620 A$=INKEY$:IFA$=""THEN620
630 IF A$="Y" THEN PTS=0:TM=0:GOTO2
    00
640 CLS:END
1000 DATA 1111111111{10 SPACES}1111
     111111
1010 DATA 2222222221{10 SPACES}1222
     2222222
1020 DATA 3333333321{10 SPACES}1233
     3333333
1030 DATA 1111111132111111111111231
     1111111
1040 DATA 2222221322222222222222231
     2222222
1050 DATA 3333321333333333333333331
     2333333
1060 DATA 1111321111111111111111111
     2311111
1070 DATA 2222132222222222222222222
     2312222
2000 ON Z GOSUB 2100,2200,2210,2400
     ,2500,2600
2010 RETURN
2100 PRINT@PS,CHR$(145);:RETURN
2200 PRINT@PS,CHR$(147)CHR$(146);:R
     ETURN
2210 PRINT@PS,CHR$(151)CHR$(146);:P
     RINT@PS+32,CHR$(148);:RETURN
2400 PRINT@PS,CHR$(150);CHR$(158);C
     HR$(146);:PRINT@PS+32,CHR$(148
     )CHR$(156);:RETURN
```



A game of "Devastator" is just starting. TI version.

```
2500 PRINT@PS,CHR$(151)CHR$(157)CHR
     $(157)CHR$(157)CHR$(146);:PRIN
     T@PS+32,CHR$(148)CHR$(156)CHR$
     (156)CHR$(156);:RETURN
2600 PRINT@PS+1,CHR$(147)CHR$(159)C
     HR$(159)CHR$(147);:PRINT@PS+32
     ,CHR$(148)CHR$(155)CHR$(155)CH
     R$(155)CHR$(159)CHR$(152);:RET
     URN
2999 REM *RANDOM -1,0,1*
3000 R=RND(0):R=(R<.3)-(R>.6):RETUR
     N
```

Program 4:

Devastator – TI-99/4A (Extended BASIC) Version

by Patrick Parrish, Programming Supervisor

```
99 REM DEVASTATOR
100 GOTO 150
110 FOR F=12 TO 14 :: CALL COLOR(F,
    2,1):: NEXT F :: RETURN
120 FOR F=10 TO 16 :: CALL SCREEN(F
    ):: NEXT F :: CALL SCREEN(2)::
    RETURN
130 FOR V=1 TO 30 :: CALL SOUND(D1,
    F1,V,F2,V):: NEXT V :: RETURN
140 FOR ROW=2 TO 7 :: CALL HCHAR(RO
    W,22,32,7):: NEXT ROW :: RETURN
150 RANDOMIZE
160 DIM E$(13)
170 CALL CLEAR :: CALL SCREEN(2)
180 GOSUB 530
190 GOSUB 1030 :: CALL CLEAR :: CAL
    L SCREEN(2)
200 FOR H=2 TO 14 :: CALL COLOR(H,2
    ,2):: NEXT H
210 FOR J=1 TO 4 :: FOR I=1 TO 11 :
    : CALL HCHAR(I,INT(RND*28)+3,46
    ):: NEXT I :: NEXT J
220 DISPLAY AT(13,1):"''''''''''a
    {6 SPACES}b''''''''''"
230 DISPLAY AT(14,1):"hhhhhhhhhi'.
    {6 SPACES}'jhhhhhhhhh"
240 DISPLAY AT(15,1):"pppppppqh''
    {6 SPACES}'hrpppppppp"
250 DISPLAY AT(16,1):"pppppppqpha''
    ''''bhprpppppppp"
260 DISPLAY AT(17,1):"''''''appihhh
    hhhhhjppb''''''"
270 DISPLAY AT(18,1):"''''''a'pqpppp
    pppppppprp'b''''''"
280 DISPLAY AT(19,1):"hhhhi''qpppppp
    pppppppr''jhhhh"
290 DISPLAY AT(20,1):"hhhih'a''''''
    ''''''''b'hjhhh"
300 DISPLAY AT(21,1):"hhihha''''''''
    ''''''''bhhjhh"
310 DISPLAY AT(22,1):"pqhhihhhhhhhh
    hhhhhhhhhhihhrp"
320 DISPLAY AT(23,1):"qphihhhhhhhhh
    hhhhhhhhhhhjhpr" :: DISPLAY AT
    24,1):"ppihhhhhhhhhhhhhhhhhhhhh
    hjpp"
330 FOR J=1 TO 2 :: FOR I=12 TO 14
    :: CALL HCHAR(I,INT(RND*6)+14,
    6):: NEXT I :: NEXT J
340 DISPLAY AT(2,24):CHR$(120)&CHR
    (121):: DISPLAY AT(3,23):CHR$(
    22)&CHR$(136)&CHR$(137)&CHR$(1
    3)
```

TI-99/4A Version Notes

The TI-99/4A version of "Devastator" (Program 4) is written in Extended BASIC and requires a joystick. As the game begins, you are cruising above the ominous *Devastator*. A guardian ship from *Devastator* appears. You must eliminate this alien ship and at least nine others that follow in a given period. If you fail, *Devastator* blasts Earth with a lethal laser.

Two levels of difficulty are offered in this version. On either level, you can eliminate the guardian ship by simply positioning the cross hairs over them using the joystick. The main difference between skill levels is the size of these guardian ships (which are actually sprites). The CALL MAGNIFY statement in line 420 produces ships of two sizes. Consequently, on level one, guardian ships are large and can be easily destroyed, but level two features smaller ships which require greater dexterity to eliminate.

The primary game loop for the program is from line 450 to 510. The counter W in line 500 is increased each time through the loop. When W reaches 200, the game is over and Earth is either blasted or not, depending on whether you've destroyed the required number of guardian ships. If the game as written is just too easy or too difficult for you on the skill levels offered, vary the time limit (200) to achieve a comfortable level of play.

The programming techniques used here might aid you in writing your own programs on the TI. You may notice that program execution appears to pause between the title page and the appearance of the playfield (background). Actually, the playfield is being set up, but since the foreground and background colors of all characters are defined as black, nothing appears at this point because the screen color is also black. When all characters on the playfield have been printed, color codes are assigned simultaneously using the CALL COLOR statement so that the entire game field appears at once.

Another trick, also achieved with color coding of characters, gives the game a 3-D effect. The *Devastator* is first printed in lines 220 to 320, using redefined characters from three character sets. By constantly shifting the foreground and background colors of these character sets in line 450, an illusion of movement is produced. Thus, as you watch the screen, you feel that you are actually circling this colossal ship.

```
350 DISPLAY AT(4,22):CHR$(124)&CHR$
    (125)&CHR$(138)&CHR$(139)&CHR$(
    125)&CHR$(126)
360 DISPLAY AT(5,22):CHR$(127)&CHR$
    (125)&CHR$(140)&CHR$(141)&CHR$(
    125)&CHR$(128)
370 DISPLAY AT(6,23):CHR$(129)&CHR$
    (142)&CHR$(143)&CHR$(130)
380 DISPLAY AT(7,24):CHR$(131)&CHR$
    (132)
390 CALL COLOR(12,6,1):: CALL COLOR
    (13,6,1):: CALL COLOR(14,3,6)
400 FOR F=2 TO 8 :: CALL COLOR(F,16
    ,1):: NEXT F
410 CALL SPRITE(#2,108,11,80,80)
420 CALL MAGNIFY(LEVEL):: SPEED=8 :
    : TOL=30 :: IF LEVEL=3 THEN TOL
    =15
430 CALL SPRITE(#1,100,16,100,110)
440 A=9 :: B=10 :: C=11
450 T=A :: A=B :: B=C :: C=T
460 CALL COLOR(A,2,5):: CALL COLOR(
    B,2,14):: CALL COLOR(C,2,7)
470 CALL MOTION(#2,INT(RND*40-20),I
    NT(RND*40-20))
480 CALL JOYST(1,X1,Y1):: CALL MOTI
    ON(#1,-Y1*SPEED,X1*SPEED)
490 CALL COINC(#1,#2,TOL,G):: IF G
    THEN GOSUB 700
500 W=W+1 :: IF W>200 THEN 770
510 GOTO 450
520 REM DEFINE CHARS
530 A$="" :: B$="0102040810204080"
    :: C$="8040201008040201"
540 CALL CHAR(95,B$)
550 FOR I=96 TO 112 STEP 8 :: CALL
    CHAR(I,A$):: CALL CHAR(I+1,B$)
560 CALL CHAR(I+2,C$):: NEXT I
570 FOR I=0 TO 13 :: READ E$(I):: C
    ALL CHAR(120+I,E$(I)):: NEXT I
580 FOR I=0 TO 7 :: READ E$(I):: CA
    LL CHAR(I+136,E$(I)):: NEXT I
590 DATA 00000000000000F7F,000000000
    000F0FE,01030F1F3F7FFFFF
600 DATA 80C0F0F8FCFEFFFF,0001010103
    030303,FFFFFFFFFFFFFFFF
610 DATA 0080808080C0C0C0C0,0303030301
    0101000,C0C0C0C080808000
620 DATA FF7F3F3F1F0F0703,FFFEFCFCF8
    F0E0C0,7F0F000000000000
630 DATA FEF0000000000000,0800667C18
    666810
640 DATA E0F07F7F7FFFFFFF,0818F8F8F0
    F8F0F0,7F7F7F3D1C0E0201
650 DATA F0F0908800180000,03070F0F0F
    070703,F0FFFFFEFCFCF8F0
660 DATA 0303010101010101,E0C0C0C080
    808000
670 CALL CHAR(108,"00073FE2E2E2FFFF
    667F0C1C000000000E0FC474747FFF
    F66FE3038000000000")
```

JACKPOT

Rick Rothstein

Now you can experience the thrill of slot machines without the danger of losing your money. These programs will show you how the bandits work and also how difficult it really is to hit a jackpot! Versions for TI-99/4A with Extended BASIC, Commodore 64, VIC-20, Atari, and IBM PC/PCjr (Color/Graphics Monitor Adapter required on PC).

Have you ever been to a casino in Las Vegas or Atlantic City? If so, your first visit probably left you dumbstruck over the sheer number of slot machines waiting to take your money. These nefarious one-arm bandits dazzle you with bright lights and promises of instant wealth.

A recent trip to Atlantic City—and an unprofitable encounter with some of these machines—prompted me to write "Jackpot." The program features three very different playing levels. Level one offers true casino odds; level two offers very generous odds which gives the player roughly the same odds that a casino normally enjoys; and level three will, in the long run, make you the owner of the casino.

Frustrating Experiences

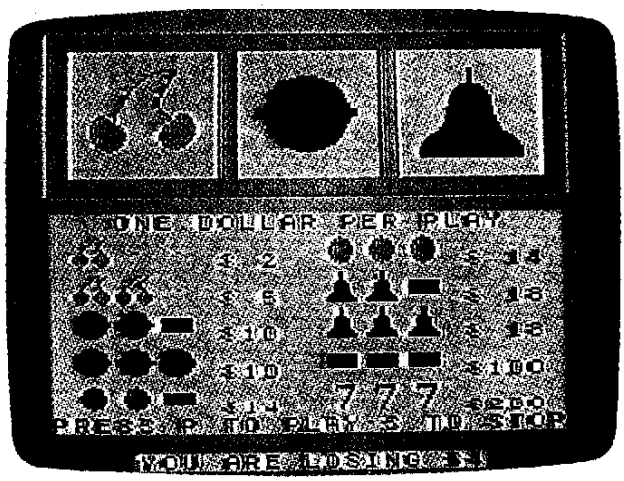
After you experience the frustrations of playing against the legitimate odds of level one, level two should give you a small measure of satisfaction if you play it long enough. Level three was included for you to play *after* you discover that level two, although tilted in your favor, is not overly generous.

Colorful graphics are used to display a payout chart, your current monetary status, and three large windows through which cherries, limes, plums, bells, bars, or lucky sevens will show. The shape displayed in each window is picked at random from 20-position wheels containing the above six shapes scattered randomly around them. The number of times each shape appears on each wheel was selected to produce the desired odds for each level of play.

Payout?

Before play begins in the TI version, the number 1 is displayed in each window, and the prompt ENTER LEVEL appears under the payout chart. If you press the space bar, the displayed level number will change. Press ENTER to begin the game at the displayed level. The payout chart continually prompts you to press the letter P to play and S to stop (the game). In addition, pressing AID (FCTN-7) will allow you to enter a new level of play, and pressing REDO (FCTN-8) will reset your money status to even while retaining the same level of play.

This program is written in Extended BASIC,



A winning combination on the TI version of "Jackpot."

and because it uses both upper- and lowercase letters, it can only be typed into a 99/4A console. However, once the program is recorded on tape or disk it will load and run properly on the older 99/4 console.

In order to facilitate use of the automatic NUMBERing command built into the 99/4A, the line numbers for the program logic begin at line 100 and increase in increments of ten. (Except for the introductory REMarks, all other REMark statements have line numbers ending in five and may be omitted.)

A Character Of Its Own

One of the strongest features of the 99/4A is its ability to display high-resolution graphics and up to 16 colors simultaneously. This program makes excellent use of these features by using seven *different* colors and redefining all 112 Extended BASIC characters which make up the highly detailed displays.

Although the program logic and mathematical theory of slot machines will not be explained, here are some of the programming techniques used in the TI version:

Line 120 sets the foreground and background colors of character set 0, which contains the cursor symbol and the edge character, to the same color and then fills the screen with the cursor symbol. Although the characters in this set cannot be redefined, turning the foreground and background to the same color has the same effect as redefining them to solid blocks of color. Filling the screen with one of these characters produces a solid background color which is independent of any other character—something the blank character cannot do.

After all of the characters have been redefined, they are combined into strings and placed on the screen with the DISPLAY AT command of Extended BASIC. This is a much faster way to place graphics on the screen than using the CALL HCHAR or CALL VCHAR subprograms.

The first statement in line 170 uses a random number, from the sequence that RANDOMIZE generates, for each loop in which either no key or an unrecognized key is pressed. This technique insures that the sequence of plays will not be repeated, since the time period between recognized keypresses will vary from play to play and from person to person.

Sluggish Sprites

Most programmers who work in Extended BASIC think sprites are useful only when they move. Actually, they can be very handy if placed on the screen and left stationary. In this program, one sprite, doubled in size by the CALL MAGNIFY (2) subprogram, is placed in front of each window. They serve as level-of-play indicators and are left transparent during game play. When needed, a simple CALL COLOR makes them visible. The advantage of using sprites in this particular application is that characters (numbers in this case) defined in an area measuring two characters by two characters are displayed with no additional character redefinitions. (Remember, all 112 Extended BASIC characters were redefined and used for the display graphics.) Without sprites, 12 additional character redefinitions would have been necessary to create the three large-sized numbers needed for the level-of-play indicator.

If you wish to save the time and effort of typing this program in, I will be glad to make a copy for you (TI version *only*). Just send $3, a blank cassette or disk, and a self-addressed, stamped mailer to:

Rick Rothstein
P.O. Box 4169
Trenton, NJ 08610

Program 1: TI-99/4A Jackpot

```
99 REM EXTENDED BASIC REQUIRED
100 CALL CLEAR :: CALL SCREEN(12)::
     CALL COLOR(0,12,12):: CALL HCH
    AR(1,1,30,768)
110 CALL COLOR(1,5,16,2,7,16,3,2,16
    ,4,2,16,5,2,16,6,7,16,7,2,16,8,
    7,16)
120 CALL COLOR(9,13,16,10,14,16,11,
    14,16,12,5,16,13,13,16,14,13,16
    )
130 RANDOMIZE :: LEVEL=49 :: TOTAL=
    0 :: OPTION BASE 1 :: DIM SHAPE
    $(6,5),WHEEL$( 3,3),PICK(3):: G
    OTO 310
135 REM ** P,S OR AID PRESSED **
140 RANDOM=RND :: CALL KEY(0,KEY,ST
    ATUS):: IF STATUS=0 THEN 140
```

```
150 IF KEY=83 OR KEY=115 THEN CALL
    CLEAR :: CALL COLOR(1,1,1):: EN
    D ELSE IF KEY= 80 OR KEY=112 TH
    EN TOTAL=TOTAL-1 :: GOTO 200
160 IF KEY=6 THEN TOTAL=0 :: GOSUB
    810 :: GOTO 140 ELSE IF KEY<>1
    THEN 140
170 GOSUB 770 :: CALL COLOR(#1,2,#2
    ,2,#3,2):: DISPLAY AT(24,1)BEEP
    :RPT$(CHR$(30),8)&"ECTERwFEVEF"
    &RPT$(CHR$(30),9)
175 REM ** CHANGE LEVEL **
180 CALL KEY(0,KEY,STATUS):: IF STA
    TUS<1 THEN 180 ELSE IF KEY=13 T
    HEN GOSUB 780 :: GOTO 140 ELSE
    IF KEY<>32 THEN 180
190 LEVEL=LEVEL+1+3*(LEVEL>50):: DI
    SPLAY AT(1,2)SIZE(1)BEEP:"K" ::
     CALL PATTERN(#1,LEVEL,#2,LEVEL
    ,#3,LEVEL):: GOTO 180
195 REM ** PICK 3 SHAPES **
200 CALL SOUND(50,-2,0):: GOSUB 810
     :: GOSUB 770 :: FOR I=1 TO 3 :
    : PICK(I)=VAL(SEG$(WHEEL$(LEVEL
    -40,I),INT(20*RND+1),1)):: NEXT
     I
205 REM ** DISPLAY SHAPES **
210 FOR I=4 TO 20 STEP 8 :: FOR J=3
     TO 7 :: DISPLAY AT(J,I)SIZE(5)
    :SHAPE$(PICK((I+4)/8),J-2):: NE
    XT J
220 CALL SOUND(50,-6,0):: NEXT I ::
     CALL SOUND(100,44000,30)
225 REM ** CHECK FOR WIN **
```



```
230 IF PICK(1)=1 THEN IF PICK(2)=1
    THEN COINS=5 :: GOTO 270 ELSE C
    OINS=2 :: GOTO 270
240 IF PICK(1)<>PICK(2)THEN 140 ELS
    E IF ((PICK(2)<>PICK(3)OR PICK(
    2)=5)AND PICK( 3)<>5)OR(PICK(2)
    =6 AND PICK(3)=5)THEN 140
250 IF PICK(1)<5 THEN COINS=2+4*PIC
    K(1):: GOTO 270
260 IF PICK(1)=5 THEN COINS=100 ::
    GOTO 280 ELSE COINS=200 :: GOTO
     290
265 REM ** UPDATE MONEY STATUS **
270 FOR I=1 TO COINS :: TOTAL=TOTAL
    +1 :: GOSUB 810 :: CALL SOUND(2
    5,1000,0,3250,0,6750,0):: NEXT
    I :: GOTO 140
280 FOR I=5 TO COINS STEP 5 :: TOTA
    L=TOTAL+5 :: GOSUB 810 :: CALL
    SOUND(35,1000, 0,3250,0,6750,0)
    :: NEXT I :: GOTO 140
290 FOR I=40 TO COINS STEP 40 :: FO
    R SIREN=700 TO 900 STEP 10 :: C
    ALL SOUND(-99,SIREN,0):: NEXT S
    IREN :: TOTAL=TOTAL+40 :: GOSUB
     810
300 FOR SIREN=900 TO 700 STEP -20 :
    : CALL SOUND(-200,SIREN,0):: NE
    XT SIREN :: NEXT I :: GOTO 140
305 REM ** DEFINE GRAPHICS **
310 CALL CHAR(33,"1818181818181FF0
    103070F0F0F0F00C0E0E0F0F0F0F"
    ,36,"0F1F1F1F1F1F3F3FF0F8F8F8F8
    F8FCFC000000000030F1F3F7F7FFFF
    FFFFFFF")
```

```
320 CALL CHAR(32,"FFFFFFFFFFFFFFFF"
    ,40,"0000000000001C3E7F0000000078
    FCFEFE7FFFFFFFFFFFFF7FC0F0F8F8F
    CFCFCFC")
330 CALL CHAR(44,"030F1F1F3F3F3F3FFF
    FFFFFFFFFFFFFEFE7F3F3F0F0F030000000
    F8F8F0E08")
340 CALL CHAR(48,"00384444444438000
    0103010101038000003844081020 7C00
    00384418044438")
350 CALL CHAR(52,"0008182848 7C080000
    07840780444380000384078444443800
    007C040810202")
360 CALL CHAR(56,"00384438444438000
    03844443C0478000000000000003F3F3F
    000000000FCFCFC")
370 CALL CHAR(60,"3F3F3F000000000000F
    CFCFC0000000000000385430185438000
    FFFFFFFFFFFFFFFF")
380 CALL CHAR(64,"0038101010103800 00
    3844447C4444000003C223C222 23C00
    00446454544C44")
390 CALL CHAR(68,"0078444444447800 00
    07C407840407C0000404040404 07C00
    003844404C4438")
400 CALL CHAR(72,"1F1F0F0300000000 0F
    CF8F0C0000000000183C7E7E7E3C1800
    FFFFFFFFFFFFFFFF")
410 CALL CHAR(76,"00000/0/040000000 0
    000F8F8183030600000000010101000 00
    60C0C080808")
420 CALL CHAR(80,"0078444478404000 0
    04444281010100000784444784844 00
    00384430084438")
430 CALL CHAR(84,"007C1010101010000 0
    044444444443800004444444428100 00
    00444444545428")
440 CALL CHAR(88,"FFFFFFFFFFFFF8F8F
    FFFFFFFFFFF0000FFFFFFFFFFFFFEFC
    01010307070F1F 1F")
450 CALL CHAR(92,"FCF8F0E0C0C080803
    F3F3F3E7E7E7E7E7C"&RPT$("FC",15
    ),96,"FF7F1F0700000000FFFFFFFFF
    F000000FFFEF8E000000000FFFFFFFF
    FFFFFFFF")
460 CALL CHAR(100,"000000 30F1F1F3F7F
    0000C0F0F8F8FCFE7F3F1F1F0F03000
    0FEFCF8F8F0C")
470 CALL CHAR(104,"00030F1F3F7FFFFF
    00C0F0F8FCFEFFFF01010103030707 0
    7808080C0C0E0E0E")
480 CALL CHAR(108,"070F0F0F0F0F0F07
    E0F0F0F0F0F0F0E00707070303010 10
    1E0E0E0C0C0808080 8")
490 CALL CHAR(112,"FFFF7F3F1F0F0300
    FFFFFEFCF8F0C000FFFFFFFFFFFFFFF
    F000001070F0F1F1F")
500 CALL CHAR(116,"000080E0F0F0F8F8
    1F1F0F0F0703010000F8F8F0F0E0800000
    00")
510 CALL CHAR(120,"FCFEFEFFFFFFFFFF
    0000000000C0F0F8000101010103070
    700808080 80C0E0E")
520 CALL CHAR(124,"070 70F1F3F3F0000
    E0E0F0F8FCFC0000C0000000000000000
    00000003")
530 CALL CHAR(128,"0001060810204080
    3FC1030204040808010202040408080
    810102020404040 4")
540 CALL CHAR(132,"4020201008040201
    0000000001020408000006488102020 1
    0000000000071F7FFF")
550 CALL CHAR(136,"000000FFFFFFFFFF
    00000000E0F8FEFF0103030707 0F0F1
    F80C0C0E0E0F0F0F8")
560 CALL CHAR(140,"1F3FFFFFFFFF3F1F
    F8FCFFFFFFFFC01C0C0C0707 03030
    1F8F0F0E0E0C0C08")
565 REM ** CREATE SHAPES **
570 SHAPE$(1,1)="ww"&CHR$(128)&CHR$
    (129)&"w" :: SHAPE$(1,2)="w"&CH
    R$(130)&"w"&CHR$(131)&"w" :: SH
    APE$(1,3)="("&CHR$(131)&"w"&CHR
    $(132)&")"
580 SHAPE$(1,4)="*+w,-" :: SHAPE$(1
    ,5)="./wHI" :: SHAPE$(2,1)="w"&
    CHR$(135)&CHR$(136)&CHR$(137)&"
    w"
590 SHAPE$(2,2)=CHR$(138)&"ccc"&CHR
    $(139):: SHAPE$(2,3)=CHR$(140)&
    "ccc"&CHR$(141):: SHAPE$(2,4)=C
    HR$(142)&"ccc"&CHR$(143)
600 SHAPE$(2,5)="w`abw" :: SHAPE$(3
    ,1)="whriw" :: SHAPE$(3,2)="jrr
    rk" :: SHAPE$(3,3)="lrrrm" :: S
    HAPE$(3,4)="nrrro" :: SHAPE$(3,
    5)="wprqw"
610 SHAPE$(4,1)="ww!ww" :: SHAPE$(4
    ,2)="w"" #w" :: SHAPE$(4,3)="w$
     %w" :: SHAPE$(4,4)="&' xy" ::
    SHAPE$(4,5)="'(3 SPACES)x"
620 SHAPE$(5,1)="wwwww" :: SHAPE$(5
    ,2)="?????" :: SHAPE$(5,3)="?BA
    R?" :: SHAPE$(5,4)="?????" :: S
    HAPE$(5,5)="wwwww"
630 SHAPE$(6,1)="wXYZw" :: SHAPE$(6
    ,2)="ww[\w" :: SHAPE$(6,3)="ww]
    ww" :: SHAPE$(6,4)="ww^ww" :: S
    HAPE$(6,5)="ww_ww"

635 REM ** DISPLAY GRAPHICS **
640 DISPLAY AT(1,2)SIZE(25):RPT$("K
    ",25):: GOSUB 770 :: DISPLAY AT
    (9,2)SIZE(25):RPT$("K",25)
650 DISPLAY AT(11,2)SIZE(25):"www0C
    EwD0FFARwPERwPFAQwww"
660 DISPLAY AT(12,2)SIZE(25):"w"&CH
    R$(133)&CHR$(134)&"wwwwwwwwwwst
    ststwwwwww" :: DISPLAY AT(13,2)
    SIZE(25):"wJJw{,}w{,}w>w2wwuvuv
    uvw>w14w"
670 DISPLAY AT(14,2)SIZE(25):"w"&CH
    R$(133)&CHR$(134)&CHR$(133)&CHR
    $(134)&"wwwwwwwwz{z{:;wwwwww"
680 DISPLAY AT(15,2)SIZE(25):"wJJJJ
    w{,}w>w5ww!{,}!{,}<=w>w18w" ::
    DISPLAY AT(16,2)SIZE(25):"wdede
    :;wwwwwwz{z{z{wwwwww"
690 DISPLAY AT(17,2)SIZE(25):"wfgfy
    <=w>10ww!{,}!{,}!{,}w>w18w" ::
    DISPLAY AT(18,2)SIZE(25):"wdede
    dewwwwww:;:;:;wwwwww"
700 DISPLAY AT(19,2)SIZE(25):"wfgfy
    fgw>10ww<=<=<=w>100w" :: DISPLA
    Y AT(20,2)SIZE(25):"wstst:;wwww
    wwLMLMLMwwwwwww"
710 DISPLAY AT(21,2)SIZE(25):"wuvuv
    <=w>14wwNONONOw>200w" :: DISPLA
    Y AT(22,2)SIZE(25):"PRESSwPwT0w
    PFAQ"&CHR$(127)&"SwT0wST0P"
720 CALL MAGNIFY(2):: CALL SPRITE(#
    1,LEVEL,1,29,53,#2,LEVEL,1,29,1
    17,#3,LEVEL,1,29,181)
725 REM ** PUT SHAPES ON WHEEL **
```

```
730 FOR I=1 TO 3 :: FOR J=1 TO 3 ::
    READ ORDER$ :: WHEEL$(I,J)=ORD
    ER$ :: NEXT J :: NEXT I :: KEY=
    1 :: GOTO 160
735 REM ** ORDER OF SHAPES **
740 DATA 253123642453142532 34,14216
    3131564252131 32,234243254243642
    34324
750 DATA 126531241351243152 46,62543
    512136423146352,243524635234235
    42364
760 DATA 521346461215315362 41,56231
    5341462131256 45,234562463562543
    52434
765 REM ** CLEAR WINDOWS **
770 FOR I=2 TO 8 :: DISPLAY AT(I,2)
    SIZE(25):"KWWWWWWWKWWWWWWWKWWWW
    WWWK" :: NEXT I :: RETURN
775 REM ** INITIAL WINDOW SHAPES **
780 FOR I=1 TO 3 :: PICK(I)=VAL(SEG
    $(WHEEL$(LEVEL-48,I),INT(20*RND
    +1),1)):: NEXT I
790 CALL COLOR(#1,1,#2,1,#3,1):: TO
    TAL=0 :: FOR I=4 TO 20 STEP 8 :
    : FOR J=3 TO 7
800 DISPLAY AT(J,I)SIZE(5):SHAPE$(P
    ICK((I+4)/8),J-2):: NEXT J :: C
    ALL SOUND(35,-6,0):: NEXT I ::
    CALL SOUND(100,44000,30)
805 REM ** DISPLAY MONEY STATUS **
810 IF TOTAL=0 THEN DISPLAY AT(24,1
    ):RPT$(CHR$(30),5)&"Q0UwAREwC0W
    wEVEC"&RPT$(CHR$(30),7):: RETUR
    N
820 TOTAL$=STR$(ABS(TOTAL)):: LENGT
    H=LEN(TOTAL$):: COLUMN=6+(TOTAL
    >0)-INT(.5+LENGTH/2)
830 IF TOTAL>0 THEN DISPLAY AT(24,C
    OLUMN)SIZE(20+LENGTH):CHR$(30)&
    "Q0UwAREwW@CC@CGw>"&TOTAL$&RPT$
    (CHR$(30),4):: RETURN
840 IF TOTAL<0 THEN DISPLAY AT(24,C
    OLUMN)SIZE(18+LENGTH):CHR$(30)&
    "Q0UwAREwF0S@CGw>"&TOTAL$&RPT$(
    CHR$(30),4):: RETURN
```

Program 2: VIC Jackpot

by Kevin Mykytyn, Editorial Programmer

Refer to the "Automatic Proofreader" article before typing this program in.

```
10 POKE52,28:POKE56,28:POKE51,0:POKE55,0:
   GOSUB300                          :rem 158
20 PRINT"{CLR}";:FORA=1TO4:FORB=1TO22:PRI
   NT"@";:NEXT:NEXT                  :rem 248
30 FORA=1TO2:PRINT"@@@@@@{2 SPACES}@@
   {2 SPACES}@@{2 SPACES}@@@@@@";:NEXT
                                     :rem 141
40 FORA=1TO3:FORB=1TO22:PRINT"@";:NEXT:NE
   XT                                :rem 32
50 PRINTB$V$B$V$B$" 100 "V$B$V$B$"
   {4 SPACES}50":PRINTS$V$S$V$S$"
   {2 SPACES}50 "V$S$V$S$"{4 SPACES}25"
                                     :rem 172
70 PRINTBE$V$BE$V$BE$"{2 SPACES}18 "V$BE$
   V$BE$V$B$"{2 SPACES}10"           :rem 101
80 PRINTC$V$C$V$C$"{2 SPACES}15 "V$C$V$C$
   "{5 SPACES}5 ":PRINTV$P$V$P$V$P$"
   {2 SPACES}14 "V$P$V$P$V$B$"{2 SPACES}1
   0"                                :rem 158
95 PRINTL$V$L$V$L$"{2 SPACES}10 "V$C$"
   {7 SPACES}2"                      :rem 59
100 A=RND(1):A$="":GETA$:IFA$<>"P"ANDA$<>
    "E"THEN100                       :rem 75
110 IFA$="E"THENPRINT"{CLR}":POKE36869,24
    0:END                            :rem 43
115 T$="{DOWN}{2 SPACES}{UP}{2 LEFT}
    {2 SPACES}{DOWN}":GOSUB210:GOSUB220:G
    OSUB220:PRINTY$"{UP}{21 SPACES}"
                                     :rem 127
120 W=0:H=0:N=1:GOSUB200:GOSUB210:GOSUB26
    0:N=2:GOSUB200:GOSUB220:GOSUB260
                                     :rem 10
140 N=3:GOSUB200:GOSUB220:GOSUB260:rem 63
145 FORA=1TO24STEP2:H$=STR$(H):H$=RIGHT$(
    H$,(LEN(H$)-1))                  :rem 249
150 K=LEN(P$(A)):IFP$(A)=LEFT$(H$,K)THENW
    =-VAL(P$(A+1))                   :rem 60
160 NEXT:IFW>0THENPRINTY$"{UP}{2 SPACES}Y
    OU WIN"W-1"DOLLARS":GOSUB280     :rem 187
170 TT=TT-1:IFTT>0THENTT$=STR$(TT)+"
    {2 SPACES}":PRINTY$"{4 SPACES}TOTAL N
    OW "TT$;:POKE198,0:GOTO100       :rem 145
180 PRINTY$"{UP}{4 SPACES}YOU ARE BROKE"
                                     :rem 192
190 PRINT"{3 SPACES}PLAY AGAIN{2 SPACES}Y
    /N ";                            :rem 18
195 GETA$:IFA$<>"Y"ANDA$<>"N"THEN195
                                     :rem 59
197 IFA$="Y"THENTT=50:GOTO20         :rem 189
198 PRINT"{CLR}":END                 :rem 23
200 A=INT(RND(1)*17)+1:B=G%(N,A):T$=F$(B)
    :H=H*10+B:RETURN                 :rem 214
210 PRINT"{HOME}{4 DOWN}{6 RIGHT}"T$;:RET
    URN                              :rem 54
220 PRINT"{UP}{2 RIGHT}"T$;:RETURN
                                     :rem 253
260 POKEV,150:FORA=1TO30:NEXT:POKEV,0:IFN
    <3THENFORA=1TORND(1)*200:NEXT:rem 210
270 RETURN                           :rem 121
280 FORQ=1TOW:TT=TT+1:TT$=STR$(TT)+"
    {2 SPACES}":PRINTY$"{4 SPACES}TOTAL N
    OW "TT$;:POKEV1,220              :rem 220
290 FORA=1TO110-W:NEXT:POKEV1,0:NEXT:RETU
    RN                               :rem 66
300 PRINT"{CLR}{3 DOWN}{2 SPACES}LOADING
    {SPACE}CHARACTERS"               :rem 8
305 DIMG%(3,17):FORA=1TO3:FORB=1TO17:READ
    C:G%(A,B)=C:NEXT:NEXT            :rem 41
310 DATA1,2,3,4,5,6,7,5,7,3,4,5,6,7,3,2,3
                                     :rem 231
320 DATA1,2,3,4,5,6,7,5,7,3,4,5,6,7,3,6,3
                                     :rem 236
330 DATA1,2,3,4,5,6,7,5,7,3,4,5,6,7,3,2,3
                                     :rem 233
340 DIMP$(24):FORA=1TO24:READP$(A):NEXT
                                     :rem 74
350 DATA4,3,44,6,444,16,555,11,661,11,666
    ,15,331,11,333,19,22,26,222,51,11,51,
    111,101                          :rem 103
400 A=7168:B=7679:C=25600:FORI=ATOB:POKEI
    ,PEEK(I+C):NEXT:POKE36869,255:rem 199
410 READB:IFB=-1THEN430              :rem 95
420 FORI=0TO7:READC:POKE7168+B*8+I,C:NEXT
    :GOTO410                         :rem 24
430 B$="{RED}%&{DOWN}{2 LEFT}'(":S$="
    {RED}!-{DOWN}{2 LEFT}#$":L$="{GRN}↑<
    {DOWN}{2 LEFT}>?":C$="{RED}Z[{DOWN}
    {2 LEFT}:;":P$="{PUR}£]{DOWN}
    {2 LEFT}<="                      :rem 110
440 BE$="{YEL})*{DOWN}{2 LEFT}+,":LE$="
    {YEL}↑<{DOWN}{2 LEFT}>?":U$="{UP} ":V
```

PROGRAMMING THE TI

C. Regena

The Singing Computer

If a computer can speak and can play music, can it sing? This month, I'll try to make the TI sing. First, to make the computer talk you need the TI Speech Synthesizer, a small peripheral device that attaches to the right side of the console. To use the speech synthesizer, you also need a command module that is made to provide speech.

To do your *own* programming with speech, you also need a command module. Right now the modules available are Speech Editor, TI Extended BASIC, and Terminal Emulator II. Terminal Emulator II is the easiest to work with because you can type any word in and the computer will pronounce it phonetically. Speech Editor and Extended BASIC use CALL SAY commands and have limited vocabularies.

I've had several letters from people wondering why certain phrases don't work. To make the computer say a phrase, such as Texas Instruments, use the number sign (SHIFT 3) before the phrase. For example, CALL SAY ("#Texas Instruments").

Unlimited Speech

A bit of history here—the original speech synthesizer was designed to use the words in the Speech Editor and Extended BASIC lists. Inserts were going to be made available that had different vocabularies (that's why some of the speech synthesizers have a lift-up lid). Then the Terminal Emulator II command module was invented, which provides unlimited speech, and inserts to the synthesizer were no longer needed.

Extended BASIC has also gone through at least one revision. I assume there are very few of the original version around because most users exchanged the original module for the second version as soon as they could. The first version did not support repeating keys and was notorious for "locking up" the computer. There were also some problems with using IMAGE statements.

The Terminal Emulator II command module has a dual purpose. In fact, it's called Terminal Emulator II because it is used to make your TI act as a terminal for another computer. For telecommunications you can use your TI-99/4A with an RS-232 Interface and a telephone modem, plus the Terminal Emulator II command module.

Pages 33–42 of the Terminal Emulator II instruction manual describe how to use speech. There are two main ways to use speech, "text-to-speech" and allophone speech. I use the text-to-speech method because all you have to do is spell the text phonetically. The allophone speech can be more exact because you can specify certain sounds. The manual contains a list of allophone numbers with their sounds plus a few sample programs of how to use this method.

Singing Requires Experimentation

Working with speech in a program takes a lot of experimentation. First, you need to try different spellings to get the computer to properly say what you want it to say. Then you can try different inflection symbols, ^, _, and >. These are used to change inflections and stress points, but they can also change the tone of the voice. You can also add different pause symbols for different sounds and contours. These symbols are the comma, period, semicolon, colon, exclamation point, question mark, and space. Finally, you can alter the pitch and slope—this is what I do to make the computer sing.

To create speech, you need the following statement:

```
OPEN #1:"SPEECH",OUTPUT
```

You may use any number after the number sign, just as in opening other types of files. Later, when you want the computer to speak, just use a command such as

```
PRINT #1:"MY NAME IS SINNDY."
```

The pitch is how high or low the voice sounds and can be a number from 0 through 63. Zero is a whisper, 1 is the highest pitched voice, and 63 is the lowest pitched voice. The slope is the rate at which the pitch changes in a spoken phrase. The slope may be a number from 0 through 255. For the best results, the manual recommends a slope 3.2 times the pitch. There are certain combinations of pitch and slope that will not be accepted. The default values of pitch and slope are 43 and 128. To change the pitch and slope, use the format *//xx yyy* where *xx* is the pitch period and *yyy* is the slope level. There must be a space between the numbers. An example in a program statement would be:

```
PRINT #1:"//30 96"
```

Changing The Pitch

The following is a sample program that illustrates how the pitch and slope change the sound of the voice. I am trying different pitches from 0 to 63 (and STEPping by 2 so it won't take forever). The slope S is calculated by taking the recommended factor of 3.2 times the pitch. Remember, you may try different slopes if you prefer. B$ combines the double slashes with the pitch, a space, and the slope, so line 170 can set the pitch and slope. Line 180 then speaks the phrase.

```
100 REM PITCH AND SLOPE
110 CALL CLEAR
120 OPEN #5:"SPEECH",OUTPUT
130 FOR P=0 TO 63 STEP 2
140 S=INT(P*3.2+.5)
150 B$="//"&STR$(P)&" "&STR$(S)
160 PRINT B$
170 PRINT #5:B$
180 PRINT #5:"TRY THIS TEST."
190 NEXT P
200 END
```

Since other statements can be executed while a sound is playing, you can play a tone, then say a word. By changing the pitch and slope numbers for the speech, you can make the voice go higher or lower, and program a singing computer.

Remember—I mentioned that working with speech involves a lot of experimentation. Singing takes even more time because there are many parameters that vary with each new tone. After you change the pitch and slope, you can try the inflection symbols and the punctuation marks to vary the voice even more. The TI with Terminal Emulator II can really create synthesized speech that sounds pretty good.

Teaching The ABCs

"Alphabet Song" illustrates simple singing on the computer. However, I did not spend a lot of time fiddling with the program and trying different things to make the speech sound better. You may want to try spelling out the letter as a word, and you may want to add the inflection symbols and punctuation marks. I used different pitches for the singing, but kept the slope numbers just 3.2 times the pitch. You could vary these numbers to get a more human sound and a better singing voice.

My little boy has played a lot with the Early Learning Fun command module. One section teaches the letters of the alphabet, and the child finds the letters on the keyboard. My son is quite proficient at this and knows the names of the letters, but I realized he'd learned them in a random order. Most children learn the alphabet from the ABC song, but I had never sung it to him. I decided I'd let the computer sing it to him.

Lowercase letters are used in the program because my son already knows the capital letters and really needs a little more practice with the lowercase letters. Schoolteachers often recommend learning the lowercase letters right along with the capital letters, and all beginning reading is in lowercase letters.

Lines 120–200 define the lowercase letters. If you have saved the lowercase letters program from my August 1983 column, you can load that program, delete the PRINTing lines, then continue typing this program. If you have any problems running this program, the most likely cause is in typing the data in lines 160–200. Your actual error message will cite line 130 or line 140, but those lines are dependent upon the DATA statements. Do not type a comma at the end of a line.

Extra Option

To hear the singing you will need the TI Speech Synthesizer and the Terminal Emulator II command module. When you turn on the computer with the module plugged in, press any key to start, then press 1 for TI BASIC and program as usual. To run the program without speech, you can select option 2 when the program starts. In this case, you don't need the module or the speech synthesizer.

If you choose no speech, the variable SP will equal 2. All the IF SP=2 THEN ... statements skip over commands that require the Terminal Emulator II module. The CALL SOUND statements play the tune. I used only one note; you may add accompaniment if you'd like. After the tone is played, the letter is sung. The CALL HCHAR or CALL VCHAR statements then place the letter on the screen.

Lines 1880–1910 wait after the song is over until the user presses ENTER, then the song is repeated.

If you prefer to save typing time and effort, you can obtain a copy of this program by sending a $3 copying fee, a blank cassette or diskette, and a stamped, self-addressed mailer to:

C. Regena
P.O. Box 1502
Cedar City, UT 84720

Please specify the name of the program and that you need the TI version.

Alphabet Song

```
100 REM ALPHABET SONG
110 CALL CLEAR
120 FOR C=97 TO 122
130 READ C$
140 CALL CHAR(C,C$)
150 NEXT C
160 DATA 3D4381818181433D,BCC281818
    181C2BC,3C428080808042
    3C,000001
    0101010101,3C4281FF8080423C
170 DATA 0609080808080
    83E,0101010141221C,0000808080808
    089,0000000
    0808080808887,8890A0C0A0908884
180 DATA 0808080808080808,7884020202
    020202,BCC2818181818181,3C4281
    8181814233C,8080808080808,0101010101
    0101
190 DATA BCC2818080808
    08,3C42403C0202423C,0000080808087F08,8181818
    18181433D,4141222214140808,0404
    08885050202
200 DATA 8244281028448282,1010202040
    404,7F02040810204
    07F
210 T=600
220 PRINT TAB(8);"ALPHABET SONG"
230 PRINT : : : :"CHOOSE:"
240 PRINT : :"1  WITH SPEECH"
250 PRINT :"TERMINAL EMULATOR 2 REQ
    UIRED"
260 PRINT : :"2  NO SPEECH": :
270 CALL KEY(0,K,S)
280 IF (K<49)+(K>50)THEN 270
290 SP=K-48
300 CALL CLEAR
310 IF SP=2 THEN 340
320 OPEN #1:"SPEECH",OUTPUT
330 PRINT #1:"//43 128"
340 CALL SOUND(T,262,2)
350 IF SP=2 THEN 370
360 PRINT #1:"A"
370 CALL HCHAR(3,3,97)
380 CALL SOUND(T,262,4)
390 IF SP=2 THEN 410
400 PRINT #1:"B"
410 CALL HCHAR(2,7,104)
420 CALL HCHAR(3,7,98)
430 CALL SOUND(T,392,2)
440 IF SP=2 THEN 470
450 PRINT #1:"//30 96"
460 PRINT #1:"C"
470 CALL HCHAR(3,11,99)
480 CALL SOUND(T,392,4)
490 IF SP=2 THEN 510
500 PRINT #1:"D"
510 CALL HCHAR(2,15,100)
520 CALL HCHAR(3,15,97)
530 CALL SOUND(T,440,2)
540 IF SP=2 THEN 570
550 PRINT #1:"//27 86"
560 PRINT #1:"E"
570 CALL HCHAR(3,19,101)
580 CALL SOUND(T,440,4)
590 IF SP=2 THEN 610
600 PRINT #1:"F"
610 CALL HCHAR(2,23,102)
620 CALL HCHAR(3,23,108)
630 CALL SOUND(T*2,392,2)
640 IF SP=2 THEN 670
650 PRINT #1:"//30 96"
660 PRINT #1:"G"
670 CALL HCHAR(3,27,97)
680 CALL HCHAR(4,27,103)
690 CALL SOUND(T,349,2)
700 IF SP=2 THEN 730
710 PRINT #1:"//34 109"
720 PRINT #1:"H"
730 CALL HCHAR(7,6,104)
740 CALL HCHAR(8,6,110)
750 CALL SOUND(T,349,4)
760 IF SP=2 THEN 780
770 PRINT #1:"I"
780 CALL HCHAR(7,10,105)
790 CALL HCHAR(8,10,108)
800 CALL SOUND(T,330,2)
810 IF SP=2 THEN 840
820 PRINT #1:"//36 115"
830 PRINT #1:"J"
840 CALL HCHAR(7,14,105)
850 CALL HCHAR(8,14,108)
860 CALL HCHAR(9,14,106)
870 CALL SOUND(T,330,4)
880 IF SP=2 THEN 910
890 PRINT #1:"K"
900 PRINT #1:"//39 125"
910 CALL HCHAR(7,18,104)
920 CALL HCHAR(8,18,107)
930 CALL SOUND(T/2,294,1)
940 IF SP=2 THEN 960
950 PRINT #1:"L"
960 CALL VCHAR(12,8,108,2)
970 CALL SOUND(T/2,294,3)
980 IF SP=2 THEN 1000
990 PRINT #1:"M"
1000 CALL HCHAR(13,12,110)
1010 CALL HCHAR(13,13,109)
1020 CALL SOUND(T/2,294,2)
1030 IF SP=2 THEN 1050
1040 PRINT #1:"N"
1050 CALL HCHAR(13,17,110)
1060 CALL SOUND(T/2,294,4)
1070 IF SP=2 THEN 1090
1080 PRINT #1:"O"
1090 CALL HCHAR(13,21,111)
1100 CALL SOUND(T*2,262,2)
1110 IF SP=2 THEN 1140
1120 PRINT #1:"//43 128"
1130 PRINT #1:"P"
1140 CALL HCHAR(13,25,98)
1150 CALL HCHAR(14,25,112)
1160 CALL SOUND(T,392,2)
1170 IF SP=2 THEN 1200
1180 PRINT #1:"//30 96"
1190 PRINT #1:"Q"
1200 CALL HCHAR(18,4,97)
1210 CALL HCHAR(19,4,113)
1220 CALL SOUND(T,392,4)
1230 IF SP=2 THEN 1250
1240 PRINT #1:"R"
1250 CALL HCHAR(18,8,114)
```

```
1260 CALL SOUND(T*2,349,2)
1270 IF SP=2 THEN 1300
1280 PRINT #1:"//34 109"
1290 PRINT #1:"S"
1300 CALL HCHAR(18,12,115)
1310 CALL SOUND(T,330,2)
1320 IF SP=2 THEN 1350
1330 PRINT #1:"//36 115"
1340 PRINT #1:"T"
1350 CALL HCHAR(17,16,116)
1360 CALL HCHAR(18,16,108)
1370 CALL SOUND(T,330,4)
1380 IF SP=2 THEN 1400
1390 PRINT #1:"U"
1400 CALL HCHAR(18,20,117)
1410 CALL SOUND(T*2,294,2)
1420 IF SP=2 THEN 1460
1430 PRINT #1:"//39 125"
1440 PRINT #1:"V"
1450 PRINT #1:"//30 96"
1460 CALL HCHAR(18,24,118)
1470 CALL SOUND(T,392,2)
1480 IF SP=2 THEN 1500
1490 PRINT #1:"DUB"
1500 CALL HCHAR(23,10,118)
1510 CALL HCHAR(23,11,119)
1520 CALL SOUND(T,392,4)
1530 IF SP=2 THEN 1550
1540 PRINT #1:"BL"
1550 CALL SOUND(T*2,349,2)
1560 IF SP=2 THEN 1590
1570 PRINT #1:"//34 109"
1580 PRINT #1:"U"
1590 CALL SOUND(T,330,2)
1600 IF SP=2 THEN 1630
1610 PRINT #1:"//36 115"
1620 PRINT #1:"X"
1630 CALL HCHAR(23,15,120)
1640 CALL SOUND(T,330,4)
1650 IF SP=2 THEN 1670
1660 PRINT #1:"Y"
1670 CALL HCHAR(23,19,118)
1680 CALL HCHAR(24,19,121)
1690 CALL SOUND(T*2,294,2)
1700 IF SP=2 THEN 1730
1710 PRINT #1:"//39 125"
1720 PRINT #1:"Z"
1730 CALL HCHAR(23,23,122)
1740 CALL SOUND(T,262,2)
1750 CALL SOUND(T,262,4)
1760 CALL SOUND(T,392,2)
1770 CALL SOUND(T,392,4)
1780 CALL SOUND(T,440,2)
1790 CALL SOUND(T,440,4)
1800 CALL SOUND(T*2,392,2)
1810 CALL SOUND(T,349,2)
1820 CALL SOUND(T,349,4)
1830 CALL SOUND(T,330,2)
1840 CALL SOUND(T,330,4)
1850 CALL SOUND(T,294,2)
1860 CALL SOUND(T,294,4)
1870 CALL SOUND(T*4,262,2)
1880 CALL KEY(0,K,S)
1890 IF K<>13 THEN 1880
1900 CALL CLEAR
1910 GOTO 330
1920 END
```

Educational And Entertainment Software For The TI-99/4A

American Software has announced four new software packages for the Texas Instruments 99/4A.

In *Fireball*, an arcade game for ages ten and older, you must climb a volcano without being hit by fireballs or falling into holes. The game requires either the Editor/Assembler cartridge or the Mini-Memory cartridge. Disk only; $16.95.

Letter Fun helps preschoolers learn the letters of the alphabet using colorful graphics and music. The child can choose from three different learning levels. Speech Synthesizer and Extended BASIC are required. Cassette $19.95; disk $21.95.

Try your luck at the horse racing track with *American Derby*. This game is set up to simulate the betting that would go on at a track, including variable track conditions, an insider's sheet, and realistic odds. You can bet on up to 36 different horses. Designed for ages ten to adult; up to six may play at a time. Requires Extended BASIC. Cassette $14.95; disk $16.95.

Speed Read was written for adults who want to improve their reading speed. This package of programs includes information on the reading process as well as pacing aids and reading passages to test your speed. It requires Extended BASIC. Cassette $29.95; disk $31.95 (disk version requires memory expansion).

American Software Design & Distribution Co.
P.O. Box 46
Cottage Grove, MN 55016
(612) 459-0557

Home Educational Software For Apple, Atari, And Commodore

Sunburst Communications, which has supplied educational materials to schools for 12 years, has released three new products from their microcomputer division.

The Incredible Laboratory (ages seven to adult) uses the problem-solving strategy of trial and error and note-taking to discover what combinations of mysterious chemicals make up crazy monsters. Apple and Atari versions are available.

Challenge Math (ages 6–11) lets children practice basic math, estimation, and problem-solving skills. Available for Apple and Commodore 64.

Getting Ready To Read And Add (ages three to six) gives preschoolers practice in letter and number recognition. Available for Apple and Atari.

Suggested retail price for each program is $39.95.

Sunburst Communications, Inc.
Pleasantville, NY 10570
(914) 769-5030

New Telecommunications Package For Apple

The Networker modem, recently introduced by ZOOM Telephonics, is a complete telecommunications package for the Apple II, II+, and IIe computers.

For $129, you get a single-slot, direct-connect, 300-baud modem, terminal software, and a free subscription to The Source.

An enhanced version of the terminal software, Netmaster, can be purchased separately for $79. If purchased with the Networker, the price of the entire package is $179.

Apple owners can get a complete telecommunications package, including modem and terminal software, by purchasing the Netmaster system.

ZOOM Telephonics plans to offer a complete line of modems, including modems for the IBM-PC.

ZOOM Telephonics
207 South St.
Boston, MA 02111
(617) 423-1072

Telecommunications Aid

Source Telecomputing Corporation (STC) has announced *Apple Sourcelink*, the second in its series of communications software designed to supplement use of The Source by personal computer owners.

The software is compatible with the new Apple modem, as well as with the Hayes and Transend modem products, and is designed for the Apple II, IIe, and II+ with a minimum 48K of memory.

It combines features such as automatic dial-up and sign-on procedure for Telenet, Uninet, and Sourcenet data communications networks; "one-button" access to major services on The Source; simultaneous capture of data from The Source in the Apple memory or disks, including