

**COMPUTE!'s**

# **Commodore Collection**

Twenty-seven programs  
for the VIC and 64

## **VOLUME TWO**

Games, graphics and sound routines, educational programs, applications, utilities, and programming aids for both the VIC-20 and the Commodore 64.

A **COMPUTE! Books** Publication  
\$12.95

□ □ □ □ □

□ □ □ □ □

COMPUTE!'s

---

# Commodore Collection

VOLUME TWO

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

Greensboro, North Carolina

The following articles were originally published in *COMPUTE!* magazine, copyright 1984, COMPUTE! Publications, Inc.: "The Mozart Machine" (January); "Sound Shaper" (March); "Trident" (March); "1540/1541 Disk Housekeeping" (April); "Jackpot" (August); "ML Tracer" (August); "Canyon Runner" (October); "Chess" (December).

The following articles were originally published in *COMPUTE!'s Gazette*, copyright 1984, COMPUTE! Publications, Inc.: "LIST Freezer" (January); "Homonym Practice" (February); "VIC Piano" (February); "The Indexer" (original title: "VICreations: The Indexer," March); "French Tutor" (April); "Making Calendars" (April); "Memo Writer" (May); "Up or Down?" (original title: "The Beginner's Corner: Teaching Music with Computers," May); "File Copier" (June); "Data Files for the VIC and 64" (original title: "Tape Data Files for VIC and 64," June); "Therapy" (June); "Color Chart" (original title: "Power BASIC: Color Chart," July); "Robot Math" (July); "Cursor GET for the VIC and 64" (September); "Learning to Count" (September); "SpeedScript Customizer" (September).

The following article was originally published in *VIC Games for Kids* and *64 Games for Kids*, copyright 1984, COMPUTE! Publications, Inc.: "Build a Quiz."

Copyright 1984, COMPUTE! Publications, Inc. All rights reserved

Reproduction or translation of any part of this work beyond that permitted by Sections 107 and 108 of the United States Copyright Act without the permission of the copyright owner is unlawful.

Printed in the United States of America

ISBN 0-942386-70-1

10 9 8 7 6 5 4 3 2 1

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is one of the ABC Publishing Companies and is not associated with any manufacturer of personal computers. Commodore 64 and VIC-20 are trademarks of Commodore Electronics Limited.

# Table of Contents

---

<b>Foreword</b> .....	vii
<b>Chapter 1. Games</b> .....	1
Chess	
<i>John Krause</i> .....	3
Jackpot	
<i>Rick Rothstein</i> .....	21
<i>VIC and 64 Versions by Kevin Mykytyn</i>	
- Nirrad's Labyrinth	
<i>Darrin Mossor</i> .....	30
- Trident	
<i>C. O. Dickerson</i> .....	40
<i>64 Version by Kevin Martin</i>	
Canyon Runner	
<i>Vic Neale</i> .....	50
<i>64 Version by Kevin Mykytyn</i>	
<b>Chapter 2. Education</b> .....	65
Learning to Count	
<i>William W. Braun</i> .....	67
- Robot Math	
<i>Bob Stewart</i> .....	73
- Homonym Practice	
<i>Michael A. Tyborski</i> .....	82
French Tutor	
<i>Michael Quigley</i> .....	88
- Up or Down?	
<i>C. Regena</i> .....	100
- Build a Quiz	
<i>Clark and Kathy Kidd</i> .....	105
<b>Chapter 3. Applications</b> .....	115
SpeedScript Customizer	
<i>J. Blake Lambert</i> .....	117
Memo Writer	
<i>Mark R. Brown</i> .....	129
Making Calendars	
<i>Paul C. Liu</i> .....	140

-	Therapy	
	<i>Steven Rubio</i> .....	156
x -	The Indexer	
	<i>Dan Carmichael</i> .....	162
	<b>Chapter 4. Graphics and Sound</b> .....	167
	VIC Hi-Res Sketchpad	
	<i>Anthony T. Beville</i> .....	169
	SDA: A Sprite Design Aid for the Commodore 64	
	<i>Karl Dittman</i> .....	171
	Multichar	
	<i>John S. Graves</i> .....	177
	The Magic Pointer	
	<i>C. D. Lane</i> .....	183
	Sound Shaper	
	<i>Steven Kaye</i> .....	189
	VIC Piano	
	<i>Brad Bascom</i> .....	192
	The Mozart Machine	
	<i>Donald J. Eddington</i> .....	196
	<i>VIC and 64 Translations by Gregg Peele</i>	
	<b>Chapter 5. Utilities and Programming Aids</b> .....	203
	Color Chart	
	<i>Sheldon Leemon</i> .....	205
	Cursor GET for the VIC and 64	
	<i>David Mills</i> .....	208
	File Copier	
	<i>Martin Engert</i> .....	211
	1540/1541 Disk Housekeeping	
	<i>Michael Maione</i> .....	213
	ML Tracer	
	<i>Thomas G. Gordon</i> .....	219
	<i>VIC and 64 Versions by Tim Victor</i>	
-	LIST Freezer	
	<i>Doug Ferguson</i> .....	226
	REFMAP: A Cross-Reference Map Utility for the Expanded VIC	
	<i>Kenneth D. Day</i> .....	228
	Data Files for the VIC and 64	
	<i>Brian Prescott</i> .....	232

<b>Appendices</b> .....	237
A: A Beginner's Guide to Typing In Programs .....	239
B: How to Type In Programs .....	241
C: The Automatic Proofreader .....	243
D: Using the Machine Language Editor: MLX .....	247
E: Screen Location Table (VIC) .....	254
F: Screen Location Table (64) .....	255
G: Screen Color Memory Table (VIC) .....	256
H: Screen Color Memory Table (64) .....	257
I: Screen Color Codes .....	258
J: Screen and Border Colors (VIC Only) .....	259
K: ASCII Codes .....	260
L: Screen Codes .....	264
M: VIC Keycodes .....	266
N: Commodore 64 Keycodes .....	267
Index .....	269

□ □ □ □ □

□ □ □ □ □



# Foreword

---

Since their introduction in the early 1980s, the VIC and the 64 have earned well-deserved reputations as top-notch personal computers. Now, with *COMPUTE!'s Commodore Collection, Volume 2*, you'll be able to enjoy your Commodore computer system even more.

Do you like exciting games? For muscle-flexing action, climb into the cockpit of "Canyon Runner" and guide a high-performance aircraft through a twisting, rocky chasm. Or play the odds and go for the payoff in "Jackpot." Like something that challenges your mind? Then "Chess," a sophisticated program that allows you to play against the computer, is just for you.

Would you like to use your VIC or 64 as an educational tool? "Homonym Practice" and "French Tutor" can sharpen language skills, and "Robot Math" adds unexpected excitement to basic math. "Up or Down?" turns your computer into a music tutor. There's also "Build a Quiz," an impressive tool in itself, that lets you create multiple-choice quizzes on any subject you choose.

Applications? They're here, too. "Memo Writer" turns your VIC or 64 into a mini word processor. "The Indexer" allows you to catalog just about anything, while "The Mozart Machine" leaves no doubt that your machine is a composer as well as a computer. You're sure to enjoy "Making Calendars," a program that creates calendars for any year you specify, and a session of "Therapy" may reveal some surprising things about your personality and outlook on life.

Programmers, too, will find this book worthwhile. "SDA: A Sprite Design Aid for the Commodore 64" greatly simplifies the task of creating complex sprites. "Color Chart" makes it easy to select text and background colors when designing your own programs. You'll also learn how to trace execution in machine language programs, how to add a cursor to GET statements, and how to create tape data files. The list goes on and on.

Every program in this book has been thoroughly tested, and each article contains complete and easy-to-follow program listings. All you have to do is type them in—a job that's greatly simplified by "The Automatic Proofreader" and "MLX," two valuable program-entry aids that are also included.

*COMPUTE!'s Commodore Collection, Volume 2* has been created with every VIC-20 or Commodore 64 user in mind. Whether you're a BASIC beginner or an experienced ML programmer, this book is certain to give you many pleasant hours at the keyboard.

# Chapter 1

---

# Games

□ □ □ □ □

□ □ □ □ □

# Chess

---

John Krause

*Try to outwit your computer with this fast, multilevel chess game for the VIC (with 8K expansion) or for the 64. A joystick is required.*

**D**uring the late eighteenth century, the world was amazed by a particular machine that had the astonishing ability to play a good game of chess. It entertained royalty and played well enough to defeat that master tactician Napoleon. Hundreds of people paid to play it—until it was discovered that a human was hidden inside the machine.

A chess-playing machine remained only a dream until the late 1950s, when the first computer chess game was played. Now, the World Computer Championship—held every three years since 1974—attracts almost as much publicity as the human championship matches.

Why has there been so much interest in machines that play games? One reason is that chess can be used to measure a computer's intelligence. Chess is easy to play but difficult to master—so difficult, in fact, that some experts believe that a computer would have to be almost as intelligent as a human to become world champion.

Of course, another reason is that chess is just plain fun, but not if you can't find an opponent. To be an entertaining opponent, a computer chess game should be fast, easy to use, and capable of playing at several different skill levels. "Chess" has all these features and more. Although it's really no match for the very best commercial chess games, it has managed to defeat even these giants on rare occasions.

## Typing It In

Both the VIC and 64 versions load in two parts. Commodore 64 users should type in Program 1 and save it. Then type in Program 2 and save it with the name CHESS2.

The VIC version needs at least 8K of expansion memory. VIC users should make the following changes in Program 1 before saving and then type in Program 3 and save it with the name CHESS2.

```
5 POKE56,60:POKE55,0:CLR :rem 171
20 IFK<>79727THENPRINT"ERROR IN DATA":STOP:rem 129
55 POKE6656,0:POKE44,26:NEW :rem 85
140 DATA11,173,20,145,205,127,63,144,18,141,127,63
    ,140,128,63 :rem 222
```

If you are using tape instead of disk, change the 8 to a 1 in line 40 of Program 1. Make sure that the second part is saved immediately after the first part on the tape. To run either version, simply load and run the first part. The second part will load and run automatically.

### Joystick Input

After running the program, you will be asked to specify several play options. First, choose among five skill levels. Then you can decide whether to start a new game or set up some previous position. Finally, decide whether to play against the computer (you can have either white or black pieces) or watch it play against itself.

All of these options will be discussed in greater detail later, but for now, type 1 at each prompt. This puts you in command of the white pieces versus the computer on level 1, the easiest level.

The first time the program is run, you need to wait a few seconds while the computer gets itself in order. Then the board will be displayed with your pieces on the bottom of the screen and the computer's pieces on the top. You should see a frame around the square in the lower-left corner of the board (the VIC version uses a blinking square). This is the cursor which takes the place of your hand to move pieces around the board.

Use the joystick (plugged into port 2 on the 64) to move the cursor to the piece you wish to move. Press and release the joystick button. Then move the cursor to the square you want to move to and tap the button again. Your piece moves to the new square and the computer responds almost instantly with its move.

### Changing the Position

Did you make a stupid move? No problem. One of the most exciting features of Chess is the player's ability to change the position by adding or deleting pieces. I decided to add this feature after having played several games in which I was able

to gradually maneuver into a superior position, only to throw it all away by making a bad blunder.

A piece of either color can be deleted by positioning the cursor on the piece and pressing the space bar. To add a piece or change a piece to a different one, move the cursor to the appropriate square and press P, N, B, R, Q, or K for pawn, knight, bishop, rook, queen, or king, respectively. This will put one of *your* pieces on the square. To add one of the computer's pieces, hold down SHIFT plus the appropriate key as described above.

To take back a move, use the editing keys to delete your piece and put it back on its original square. Don't forget to take back the computer's move, too.

The editing feature also enables you to make special moves which cannot be made with the joystick alone (for example, castling and *en passant* captures). Castling can be accomplished by deleting the king and putting it on its new square, and then moving the rook as you normally would with the joystick.

Although *you* can make these special moves, the computer will never castle or capture *en passant*. Due to their complexity, these moves were not included in its thinking routine.

The computer will always make a legal move, but it doesn't check to see that you do the same. You are free to move any of your pieces to any square you wish without so much as a contemptuous buzz from the computer. If you're an experienced player, that shouldn't pose a problem. If you're a beginner, however, you should familiarize yourself with the basic rules of chess lest you end up playing some weird mutation that bears little resemblance to the real game. On the other hand, if you like to cheat, it does make it easier.

When a pawn reaches the other side of the board, it's automatically promoted to a queen. If you would rather have a knight, bishop, or rook, you can easily make the change using the editing keys.

## Checkmate

The computer thinks by analyzing thousands of possible moves and countermoves and choosing what it considers to be the best one, based on the relative value of the pieces. Most positions offer several moves which are equally good, in

which case the computer chooses one at random. That random factor insures that every game will be different, and makes for varied and interesting play.

Play continues until one side is either checkmated or stalemated. The computer will then stop play and indicate which side has won.

There are a few quirks in the way the computer determines whether checkmate has occurred. On levels 3 through 5, it announces checkmate prematurely. When this happens, the computer has determined that it's impossible to avoid checkmate on the *next* move or two, assuming both sides make the best moves.

Also, the computer doesn't know the subtle difference between checkmate and stalemate. Consequently, when stalemate occurs, it will announce checkmate even though the game is a draw. Since the computer tries as hard as it can to checkmate its opponent, it will also try to achieve stalemate, possibly forcing a draw when it could have won. Fortunately, this rarely happens because the conditions for stalemate exist only in unusual circumstances (for instance, when one side has only the king remaining).

When your king is in check (not checkmate), the computer won't give you any hint that this situation exists. So be extra careful that you don't leave your king in check or move into check. Otherwise, your king would be in check during the computer's turn to move—a highly unorthodox (if not illegal) position. The computer's reply to such a position is unpredictable, but it usually announces checkmate, forcing you to restart the game.

In any case, when the computer announces checkmate, press the joystick button to start a new game. If you want to try out some of the other play options without waiting for checkmate, you can start a new game at any time by pressing RUN/STOP-RESTORE and running the program again.

### Play Options

When you choose the black pieces, the board will be inverted so that you still play from the bottom. Since the player with the white pieces always moves first, you must wait for the computer to move before you will be allowed to make your first move.

If you become mentally exhausted after several bouts



against the computer, give your brain a rest and watch the computer play itself. When you select this option, just set the joystick aside and sit back and watch the action. Beginners will find that this offers an excellent way to learn some good strategies to use against the computer.

You don't have to begin a game from the starting position. If you choose the option to set up a position, an empty board will be displayed, and you can use the editing keys to place pieces on the board in any position you choose. When the position is set up, the computer will start thinking after you make your first move.

This feature is especially useful for continuing a previous game or creating a problem for the computer to solve. It also allows you to experiment with hypothetical or downright ridiculous positions. Live out your fantasy by giving yourself ten queens versus the computer's lone king! The position doesn't even have to be a legal one. You could invent your own type of chess by giving each side two kings, for example, although the computer may get confused trying to determine when checkmate has occurred.

One of the advantages of a computer opponent over a human one is that you can tell the computer exactly how hard you want it to try to beat you—and it will obediently play at that level of difficulty. That can be important, because it's no fun if you always lose or if you win almost effortlessly every time.

You have five skill levels to choose from. The difference between one level and another is the number of future moves that the computer evaluates before making a move of its own. On level 1, for example, it looks two moves ahead (its move and your reply). Each succeeding level looks ahead one more move than the previous level.

But smarter play on higher levels doesn't come without a price. The further ahead the computer looks, the more moves it must examine and the more time it needs to think. The thinking time per move varies from one second on level 1 to about two *hours* on level 5.

Here's a rundown of the five levels:

**Level 1 (Beginner).** Thinking time: one second. Looks ahead two moves. Fast but dumb.

**Level 2 (Intermediate).** Thinking time: five seconds.

Looks ahead three moves. Provides a reasonable challenge for impatient players.

**Level 3 (Tournament).** Thinking time: two minutes.

Looks ahead four moves. Since the usual time limit for tournament play is 40 moves in two hours, an average of three minutes per move, this level is best suited for serious players.

**Level 4 (Mate in two moves).** Thinking time: 30 minutes.

Looks ahead five moves. Capable of solving most mate-in-two problems.

**Level 5 (Postal chess).** Thinking time: two hours. Looks ahead six moves. Simulates postal chess, which has no time limit.

The thinking times given here are average times. The actual times may range from half to twice the average time, depending on the position.

Level 4 can be used to solve mate-in-two problems such as those published in many newspapers. Just select the following options: level 4, set up position, computer versus itself. Enter the position using the editing keys, and then make a do-nothing move by positioning the cursor over a white piece and pressing the joystick button twice. After several minutes of deep thought, the computer should respond by moving one of the white pieces (the solution) and announcing checkmate. The only mate-in-two problems that the computer cannot solve are those which involve castling, *en passant* captures, or pawn promotion.

### How It Thinks

You've probably heard the story that if a monkey spent long enough at a typewriter, it would eventually type the complete works of Shakespeare. Theoretically, that is indeed possible—given enough time. But even at a brisk typing speed of 50 words per minute, it would take that poor monkey billions of years just to come up with "To be, or not to be."

But if you substitute a high-speed computer for the monkey, such a technique becomes a practical method of imitating intelligence. In effect, this program does just that by using a popular trial-and-error technique known as the *minimax* algorithm.

The computer looks at the present board position and mentally moves the pieces through all the possible combinations of future moves and countermoves up to a certain point,

say three moves ahead. For each combination, it calculates a score based on which pieces were captured during the combination. Each piece is worth a certain number of points depending on its general importance: 1 point for a pawn, 3 for a knight or bishop, 5 for a rook, 9 for a queen, and 46 for a king. (Of course, since you lose the game if your king cannot escape capture, the value of a king is actually infinite, but 46 is high enough as far as the computer is concerned.)

After the best combination has been found, the computer's best move in the present position is simply the first move in the combination. The problem has been reduced from analyzing a chess position to finding the maximum and minimum of a series of numbers, which is much better suited for a computer.

Like most algorithms based on trial and error, this one requires sifting through an enormous number of combinations to find the best one. Fortunately, a few tricks can be used to bring this number down to a manageable size. This algorithm uses one called *alpha-beta cutoff*. It makes the computer search more intelligently, giving it the seemingly paradoxical ability to find the best move without looking at all the possible combinations. On level 5, for example, instead of having to search through roughly 2 billion combinations, it looks at only 50 million.

Even so, it would take BASIC from now till Christmas to generate that many combinations. That's why I programmed the algorithm in machine language. To generate all the possible combinations of moves, I used an advanced programming technique known as *recursion* in which a subroutine calls itself. Capable of analyzing about 5000 combinations per second, this routine provides a moderate challenge at a reasonable playing speed.

## Program 1. VIC and 64 Loader

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

10 FOR I=15449 TO 16200:READ J:POKE I,J:K=K+J:NEXT
                                                    :rem 52
20 IF K<>79786 THEN PRINT "ERROR IN DATA":STOP:rem 134
30 POKE 631,13:POKE 632,13:POKE 633,13:POKE 198,3
                                                    2 :rem 79
40 PRINT "{CLR}{3 DOWN}LOAD"CHR$(34)"CHESS2.PRG"CHR
   RS$(34)",8
                                                    :rem 78
50 PRINT "{5 DOWN}RUN{HOME}
                                                    :rem 113
40 PRINT "{CLR}{3 DOWN}LOAD"CHR$(34)"CHESS2"CHR$(
   (34)",8
                                                    :rem 255

```

# 1: Games

---

60 DATA1,12,248,237,235,244,8,19,10,11,1,247,246,  
245,255 :rem 34  
70 DATA9,11,247,245,9,10,1,246,255,46,9,5,3,3,1  
:rem 46  
80 DATA0,1,3,3,5,9,46,120,169,192,141,128,63,162,0  
:rem 187  
90 DATA142,127,63,202,142,126,63,76,97,61,189,108,  
63,24,125 :rem 152  
100 DATA116,63,72,168,185,136,63,188,108,63,153,13  
6,63,104,168 :rem 48  
110 DATA189,76,63,153,136,63,24,105,6,168,174,73,6  
3,169,0 :rem 56  
120 DATA157,129,63,174,126,63,185,113,60,56,253,12  
9,63,168,169 :rem 55  
130 DATA192,157,129,63,152,224,0,208,34,221,128,63  
,48,28,208 :rem 190  
140 DATA11,173,4,220,205,127,63,144,18,141,127,63,  
140,128,63 :rem 170  
150 DATA173,108,63,141,124,63,173,116,63,141,125,6  
3,96,221,128 :rem 29  
160 DATA63,48,250,240,248,152,157,128,63,189,75,63  
,24,105,6 :rem 155  
170 DATA168,185,113,60,56,253,128,63,221,127,63,48  
,59,224,1 :rem 150  
180 DATA240,221,221,127,63,240,50,96,189,108,63,24  
,125,116,63 :rem 235  
190 DATA141,75,63,168,185,136,63,172,74,63,208,6,2  
01,1,16 :rem 48  
200 DATA192,48,8,201,0,48,186,201,7,240,182,157,76  
,63,201 :rem 35  
210 DATA6,240,4,201,250,208,12,169,46,157,128,63,1  
04,104,104 :rem 166  
220 DATA104,76,229,61,188,108,63,185,136,63,172,75  
,63,153,136 :rem 2  
230 DATA63,188,108,63,169,0,153,136,63,236,73,63,2  
08,3,76 :rem 55  
240 DATA144,60,232,142,126,63,169,20,157,108,63,16  
9,16,56,237 :rem 246  
250 DATA74,63,141,74,63,254,108,63,188,108,63,185,  
136,63,201 :rem 203  
260 DATA7,240,86,172,74,63,240,4,201,0,16,77,192,0  
,208 :rem 139  
270 DATA4,201,1,48,69,201,0,16,9,188,108,63,169,0,  
56 :rem 47  
280 DATA249,136,63,201,1,208,6,32,5,62,76,222,61,2  
01,2 :rem 131  
290 DATA208,6,32,192,62,76,222,61,201,3,208,6,32,2  
18,62 :rem 190  
300 DATA76,222,61,201,4,208,6,32,230,62,76,222,61,  
201,5 :rem 170

310 DATA208,6,32,242,62,76,222,61,32,47,63,76,222,  
61,189 :rem 250  
320 DATA108,63,201,98,48,150,224,0,240,16,169,16,5  
6,237,74 :rem 93  
330 DATA63,141,74,63,202,142,126,63,76,144,60,173,  
124,63,24 :rem 133  
340 DATA109,125,63,141,125,63,88,96,173,74,63,208,  
89,189,108 :rem 223  
350 DATA63,24,105,10,168,185,136,63,208,36,169,10,  
157,116,63 :rem 194  
360 DATA32,21,61,189,108,63,201,31,48,21,201,39,16  
,17,24 :rem 232  
370 DATA105,20,168,185,136,63,208,8,169,20,157,116  
,63,32,21 :rem 142  
380 DATA61,189,108,63,24,105,9,168,185,136,63,16,8  
,169,9 :rem 21  
390 DATA157,116,63,32,21,61,189,108,63,24,105,11,1  
68,185,136 :rem 196  
400 DATA63,16,8,169,11,157,116,63,32,21,61,96,189,  
108,63 :rem 0  
410 DATA56,233,10,168,185,136,63,208,36,169,246,15  
7,116,63,32 :rem 253  
420 DATA21,61,189,108,63,201,81,48,21,201,89,16,17  
,56,233 :rem 39  
430 DATA20,168,185,136,63,208,8,169,236,157,116,63  
,32,21,61 :rem 149  
440 DATA189,108,63,56,233,9,168,169,0,217,136,63,1  
6,8,169 :rem 69  
450 DATA247,157,116,63,32,21,61,189,108,63,56,233,  
11,168,169 :rem 205  
460 DATA0,217,136,63,16,8,169,245,157,116,63,32,21  
,61,96 :rem 255  
470 DATA169,0,157,84,63,168,185,89,60,157,116,63,3  
2,21,61 :rem 64  
480 DATA254,84,63,188,84,63,192,8,48,237,96,169,4,  
157,100 :rem 81  
490 DATA63,169,0,157,84,63,240,22,169,8,157,100,63  
,169,4 :rem 7  
500 DATA157,84,63,208,10,169,8,157,100,63,169,0,15  
7,84,63 :rem 53  
510 DATA168,185,105,60,157,116,63,157,92,63,32,21,  
61,189,108 :rem 202  
520 DATA63,24,125,116,63,168,185,136,63,208,13,189  
,116,63,24 :rem 200  
530 DATA125,92,63,157,116,63,76,6,63,254,84,63,189  
,84,63 :rem 23  
540 DATA221,100,63,48,206,96,169,0,157,84,63,168,1  
85,97,60 :rem 114  
550 DATA157,116,63,32,21,61,254,84,63,188,84,63,19  
2,8,48 :rem 15  
560 DATA237,96 :rem 236

## Program 2. 64 Chess (Main Program)

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

10 POKE53280,9:POKE53281,9:POKE53272,21:POKE53249,
   0 :rem 143
20 PRINTCHR$(14)"{CLR}{DOWN}{WHT}"TAB(18)"CHESS"
   :rem 94
30 PRINTTAB(15)"{DOWN}{CYN}JOHN KRAUSE" :rem 108
40 FORI=16256TO16263:POKEI,192:NEXT :rem 109
50 FORI=16264TO16383:POKEI,7:NEXT :rem 11
60 FORI=16285TO16362:READJ:POKEI,J:NEXT :rem 191
70 FORI=54272TO54296:POKEI,0:NEXT :rem 12
80 POKE54296,15:POKE54273,34:POKE54277,10 :rem 51
90 POKE53282,8:POKE53283,1 :rem 203
100 POKE2040,14:POKE53287,7:POKE53277,1:POKE53271,
   1 :rem 130
110 D$=" PNBROKPNBRQK" :rem 23
120 PRINT"{2 DOWN}{YEL}ENTER SKILL LEVEL (1-5)"
   :rem 253
130 GETA$:IFA$=""THEN130 :rem 75
140 IFVAL(A$)=0ORVAL(A$)>5THEN130 :rem 154
150 POKE16201,VAL(A$) :rem 132
160 PRINT"{DOWN}{RVS}1{OFF} NEW GAME OR {RVS}2
   {OFF} SET UP POSITION?" :rem 142
170 GETES$:IFE$=""THEN170 :rem 91
180 IFVAL(E$)=0ORVAL(E$)>2THEN170 :rem 167
190 PRINT"{DOWN}COMPUTER VS. {RVS}1{OFF} YOU OR
   {RVS}2{OFF} ITSELF?" :rem 145
200 GETA$:IFA$=""THEN200 :rem 71
210 IFVAL(A$)=0ORVAL(A$)>2THEN200 :rem 147
220 POKE16202,0:B$="2":IFA$="2"THENPOKE16202,16:B$
   ="1":GOTO260 :rem 151
230 PRINT"{DOWN}YOU HAVE THE {RVS}1{OFF} WHITE OR
   {SPACE}{RVS}2{OFF} BLACK PIECES?" :rem 27
240 GETB$:IFB$=""THEN240 :rem 81
250 IFVAL(B$)=0ORVAL(B$)>2THEN240 :rem 157
260 IFPEEK(12288)<>60THENGOSUB380 :rem 204
270 GOSUB490 :rem 182
280 IFA$="1"ANDB$="1"THEN320 :rem 239
290 IFE$="2"THENGOSUB690:POKE53269,0 :rem 98
300 GOTO330 :rem 98
310 IFA$="2"THEN330 :rem 0
320 GOSUB690:POKE53269,0:POKE16202,0 :rem 66
330 SYS15486:IFPEEK(16256)<229ANDPEEK(16256)>150TH
   ENI=0:GOTO1070 :rem 250
340 J=PEEK(16252)+16264:R=INT(J/10-1628.5):C=J-162
   85-10*R:GOSUB930 :rem 153
350 J=PEEK(16253)+16264:R=INT(J/10-1628.5):C=J-162
   85-10*R:GOSUB980 :rem 160

```

```

360 IFPEEK(16256)<99ANDPEEK(16256)>27THENI=1:GOTO1
      070                                     :rem 101
370 GOTO310                                   :rem 103
380 PRINT" {DOWN} {CYN}PLEASE WAIT..."    :rem 21
390 POKE56334,0:POKE1,51                     :rem 88
400 FORI=0TO431:POKEI+12288,PEEK(I+53248):NEXT
                                             :rem 227
410 POKE1,55:POKE56334,1                     :rem 86
420 FORI=12792TO12799:POKEI,85:NEXT         :rem 123
430 FORI=0TO383:READJ:POKE12800+I,J        :rem 99
440 POKE13184+I,JOR85                        :rem 192
450 POKE13568+I,JAND170                     :rem 36
460 POKE13952+I,(JAND170)OR(255-JAND85):NEXT
                                             :rem 49
470 FORI=896TO922:READJ:POKEI,J:NEXT       :rem 48
480 FORI=923TO958:POKEI,0:NEXT:RETURN      :rem 145
490 POKE53272,29:POKE53270,216            :rem 149
500 PRINT"{CLR}{2 DOWN}"TAB(14)" {CYN}LEVEL"PEEK(16
      201)                                     :rem 115
510 PRINT"[1]";:IFB$="1"THEN530            :rem 203
520 POKE53283,0:PRINT"[2]";:POKE16288,6:POKE16289,
      5:POKE16358,250:POKE16359,251        :rem 18
530 IFB$="1"THEN560                          :rem 12
540 FORI=0TO7:FORJ=0TO7:POKE16285+10*I+J,0:NEXT:NE
      XT                                       :rem 243
550 PRINT:GOSUB1170:GOSUB1170:GOTO680      :rem 62
560 PRINT" {DOWN} {RVS}HIJK{OFF}HIJK{RVS}@ABC{OFF}
      {SHIFT-SPACE}{K}{I}{T}{RVS}XYZ[{OFF}PQRS[U]{O}
      @F}XYZ+"                                :rem 57
570 PRINT" {RVS}LMNO{OFF}LMNO{RVS}DEFG{OFF}{@}{G}
      [+]{M}{RVS}.<=>?[-]-↑[*]4567{H}{J}{L}{Y}
      :rem 202
580 PRINT" *ABC[A]{E}{R}{W}*ABC[A]{E}{R}{W}*ABC[A]
      {E}{R}{W}*ABC[A]{E}{R}{W}"          :rem 158
590 PRINT" DEFG{H}{J}{L}{Y}DEFG{H}{J}{L}{Y}DEFG{H}
      {J}{L}{Y}DEFG{H}{J}{L}{Y}"          :rem 31
600 GOSUB1170                                 :rem 223
610 C$=CHR$(34):PRINT" {RVS}PQRS !"C$"#PQRS !"C$"#
      PQRS !"C$"#PQRS !"C$"#              :rem 229
620 PRINT" {RVS}TUVW$%&'TUVW$%&'TUVW$%&'TUVW$%&' "
                                             :rem 43
630 PRINT" {RVS}89:;XYZ+0123[A]{E}{R}{W}HIJK
      {SHIFT-SPACE}{K}{I}{T}()*+&[N]{Q}" :rem 76
640 PRINT" {RVS}<=>?[-]-↑[*]4567{H}{J}{L}{Y}LMNO
      {@}{G}{+}{M},-./[D]{Z}{S}{P}"      :rem 238
650 IFB$="1"THENRETURN                       :rem 81
660 PRINT" {HOME} {4 DOWN}"SPC(13)" {&}[N]{Q}{RVS}
      PQRS"                                  :rem 161
670 PRINTSPC(13)" {13 DOWN} {RVS}U{O}@F}*ABC
      {DOWN}"                                :rem 245

```

# 1: Games

---

```
680 RETURN :rem 126
690 POKE53269,1 :rem 52
700 GETC$:IFC$=""ORFTHEN780 :rem 68
710 N=0 :rem 83
720 IFMID$(D$,N+1,1)=C$THEN750 :rem 129
730 N=N+1:IFN<13THEN720 :rem 78
740 GOTO780 :rem 115
750 J=16285+C+10*R:IFN>6THENN=262-N :rem 249
760 IFNTHENGOSUB990:GOTO780 :rem 221
770 GOSUB940:FORI=0TO1:FORP=0TO3:POKEK+40*I+P,M:NE
XT:NEXT :rem 182
780 I=NOTPEEK(56320) :rem 140
790 R=R-SGN((IAND2)-(IAND1)) :rem 81
800 C=C+SGN((IAND8)-(IAND4)) :rem 50
810 IFR<0THENR=0 :rem 212
820 IFR>7THENR=7 :rem 229
830 IFC<0THENC=0 :rem 184
840 IFC>7THENC=7 :rem 201
850 POKE53248,30+32*C:POKE53249,193-16*R :rem 167
860 IF(PEEK(56320)AND16)THEN700 :rem 244
870 J=16285+C+10*R :rem 162
880 IFFTHEN970 :rem 68
890 IFPEEK(J)=0ORPEEK(J)>6THEN700 :rem 248
900 F=1:GOSUB930 :rem 163
910 IF(PEEK(56320)AND16)THEN700 :rem 240
920 GOTO910 :rem 110
930 POKE54276,0:POKE54276,17 :rem 52
940 K=1745-80*R+4*C:N=PEEK(J):POKEJ,0 :rem 103
950 M=32:IF(R+C)/2-INT((R+C)/2)THENM=63 :rem 197
960 RETURN :rem 127
970 F=0 :rem 83
980 FORI=0TO1:FORP=0TO3:POKEK+40*I+P,M:NEXT:NEXT
:rem 98
990 K=1745-80*R+4*C :rem 216
1000 M=0:IF(R+C)/2-INT((R+C)/2)THENM=48 :rem 182
1010 IFR=0ANDN=255THENN=251 :rem 92
1020 IFR=7ANDN=1THENN=5 :rem 150
1030 IFN<7THENM=M+96 :rem 180
1040 POKEJ,N:IFN>6THENN=256-N :rem 21
1050 FORI=0TO1:FORJ=0TO3:POKEK+40*I+J,56+M+8*N+4*I
+J:NEXT:NEXT :rem 51
1060 RETURN :rem 167
1070 IFPEEK(16202)THENI=I+1 :rem 34
1080 I=I+VAL(B$):PRINT"{DOWN}{CYN}CHECKMATE!
{2 SPACES}"; :rem 249
1090 IFI/2-INT(I/2)THENPRINT"BLACK WINS.":GOTO1110
:rem 24
1100 PRINT"WHITE WINS." :rem 131
1110 POKE54273,40:POKE54276,0:POKE54276,17 :rem 89
1120 FORI=0TO999:NEXT :rem 40
```



```
1130 POKE54273,20:POKE54276,0:POKE54276,17 :rem 89
1140 PRINT"PRESS JOYSTICK BUTTON." :rem 158
1150 IF(PEEK(56320)AND16)THEN1150 :rem 77
1160 RUN :rem 189
1170 FORI=1TO2:FORJ=1TO2 :rem 234
1180 PRINT" ???#{4 SPACES}????{4 SPACES}????
{4 SPACES}????{4 SPACES}" :rem 139
1190 NEXT:FORJ=1TO2 :rem 184
1200 PRINT"{5 SPACES}????{4 SPACES}????{4 SPACES}?
??#{4 SPACES}????" :rem 132
1210 NEXT:NEXT:RETURN :rem 150
1220 DATA4,2,3,5,6,3,2,4,7,7,1,1,1,1,1,1,1,1,7
:rem 193
1230 DATA7,0,0,0,0,0,0,0,0,7,7,0,0,0,0,0,0,0,0,7
:rem 0
1240 DATA7,0,0,0,0,0,0,0,0,7,7,0,0,0,0,0,0,0,0,7
:rem 1
1250 DATA7,255,255,255,255,255,255,255,255,7
:rem 188
1260 DATA7,252,254,253,251,250,253,254,252 :rem 69
1270 DATA0,0,0,0,0,0,0,0 :rem 152
1280 DATA0,0,0,3,15,15,3,15 :rem 65
1290 DATA0,0,0,192,240,240,192,240 :rem 164
1300 DATA0,0,0,0,0,0,0,0 :rem 146
1310 DATA0,0,0,0,0,0,0,0 :rem 147
1320 DATA3,3,15,63,63,0,0,0 :rem 66
1330 DATA192,192,240,252,252,0,0,0 :rem 165
1340 DATA0,0,0,0,0,0,0,0 :rem 150
1350 DATA0,0,0,0,3,3,3,3 :rem 163
1360 DATA0,192,240,255,255,63,255,255 :rem 83
1370 DATA0,0,0,0,240,252,252,255 :rem 61
1380 DATA0,0,0,0,0,0,0,0 :rem 154
1390 DATA15,15,3,0,0,0,0,0 :rem 10
1400 DATA255,243,3,15,63,255,255,0 :rem 178
1410 DATA255,255,255,255,255,255,255,0 :rem 136
1420 DATA0,192,192,192,192,192,192,0 :rem 29
1430 DATA0,0,0,0,0,0,0,0 :rem 150
1440 DATA0,60,60,255,255,255,255,255 :rem 31
1450 DATA0,60,60,63,207,243,243,243 :rem 225
1460 DATA0,0,0,0,0,0,0,0 :rem 153
1470 DATA0,0,0,0,15,63,48,0 :rem 69
1480 DATA63,48,63,48,255,252,0,0 :rem 90
1490 DATA252,12,252,12,255,63,0,0 :rem 121
1500 DATA0,0,0,0,240,252,12,0 :rem 150
1510 DATA0,3,3,3,0,0,0,0 :rem 158
1520 DATA0,207,207,255,192,255,255,255 :rem 132
1530 DATA0,243,243,255,3,255,255,255 :rem 28
1540 DATA0,192,192,192,0,0,0,0 :rem 220
1550 DATA0,0,0,0,3,15,15,0 :rem 8
1560 DATA255,255,255,192,255,255,255,0 :rem 142
```

# 1: Games

---

```
1570 DATA255,255,255,3,255,255,255,0           :rem 38
1580 DATA0,0,0,0,192,240,240,0                 :rem 212
1590 DATA0,0,0,0,48,48,12,12                   :rem 123
1600 DATA0,48,48,48,48,252,252,252             :rem 192
1610 DATA0,48,48,48,48,252,252,252             :rem 193
1620 DATA0,0,0,0,48,48,192,192                 :rem 231
1630 DATA15,3,3,3,3,3,3,0                       :rem 224
1640 DATA255,0,255,252,255,0,255,0            :rem 178
1650 DATA255,3,255,255,255,3,255,0            :rem 188
1660 DATA192,0,0,0,0,0,0,0                     :rem 7
1670 DATA0,0,0,15,63,63,63,15                  :rem 179
1680 DATA0,63,51,60,243,255,240,252           :rem 230
1690 DATA0,240,48,243,63,255,63,255           :rem 243
1700 DATA0,0,0,192,240,240,240,192           :rem 160
1710 DATA15,3,3,3,3,3,3,0                       :rem 223
1720 DATA255,0,255,252,255,0,255,0            :rem 177
1730 DATA255,3,255,255,255,3,255,0            :rem 187
1740 DATA192,0,0,0,0,0,0,0                     :rem 6
1750 DATA255,255,192,192,0,192,192,0,192      :rem 235
1760 DATA192,0,192,192,0,192,192,0,192        :rem 128
1770 DATA192,0,192,192,0,192,255,255,192      :rem 237
```

## Program 3. VIC Chess (Main Program)

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
10 POKE36879,138:POKE36869,194                 :rem 172
20 PRINT"{CLR}{WHT}"TAB(8)"{DOWN}CHESS         :rem 84
30 PRINT"{DOWN}{CYN}{5 SPACES}JOHN KRAUSE      :rem 188
40 FORI=16256TO16263:POKEI,192:NEXT            :rem 109
50 FORI=16264TO16383:POKEI,7:NEXT              :rem 11
60 FORI=16285TO16362:READJ:POKEI,J:NEXT        :rem 191
70 D$=" PNBROKPNBROK"                          :rem 236
80 PRINT"{2 DOWN}{YEL}SKILL LEVEL (1-5)?      :rem 113
90 GETA$:IFA$=""THEN90                          :rem 245
100 IFVAL(A$)=0ORVAL(A$)>5THEN90                :rem 107
110 POKE16201,VAL(A$)                          :rem 128
120 PRINT"{DOWN}{RVS}1{OFF} NEW GAME          :rem 172
130 PRINT"{RVS}2{OFF} SET UP POSITION           :rem 159
140 GETES$:IFE$=""THEN140                      :rem 85
150 IFVAL(E$)=0ORVAL(E$)>2THEN140              :rem 161
160 PRINT"{DOWN}COMPUTER VS.                  :rem 29
170 PRINT"{RVS}1{OFF} YOU                     :rem 25
180 PRINT"{RVS}2{OFF} ITSELF                  :rem 229
190 GETA$:IFA$=""THEN190                      :rem 87
200 IFVAL(A$)=0ORVAL(A$)>2THEN190              :rem 154
210 POKE16202,0:B$="2":IFA$="2"THENPOKE16202,16:B$
    ="1":GOTO270                                :rem 151
220 PRINT"{DOWN}YOU HAVE THE                  :rem 214
230 PRINT"{RVS}1{OFF} WHITE PIECES            :rem 83
240 PRINT"{RVS}2{OFF} BLACK PIECES           :rem 49
```

```

250 GETB$:IFB$=""THEN250 :rem 83
260 IFVAL(B$)=0ORVAL(B$)>2THEN250 :rem 159
270 IFPEEK(5120)<>28THENGOSUB390 :rem 149
280 GOSUB460 :rem 180
290 IFA$="1"ANDB$="1"THEN330 :rem 241
300 IFE$="2"THENGOSUB660 :rem 137
310 GOTO340 :rem 100
320 IFA$="2"THEN340 :rem 2
330 GOSUB660:POKE16202,0 :rem 114
340 SYS15486:IFPEEK(16256)<229ANDPEEK(16256)>150TH
ENI=0:GOTO1120 :rem 247
350 J=PEEK(16252)+16264:R=INT(J/10-1628.5):C=J-162
85-10*R:GOSUB980 :rem 159
360 J=PEEK(16253)+16264:R=INT(J/10-1628.5):C=J-162
85-10*R:GOSUB1030 :rem 196
370 IFPEEK(16256)<99ANDPEEK(16256)>27THENI=1:GOTO1
120 :rem 98
380 GOTO320 :rem 105
390 PRINT"{DOWN}{CYN}PLEASE WAIT..." :rem 244
400 FORI=0TO431:POKE5120+I,PEEK(32768+I):NEXT
:rem 170
410 FORI=0TO223:READJ:POKE6224+I,J :rem 45
420 POKE5776+I,JOR85 :rem 150
430 POKE6000+I,JAND170 :rem 225
440 POKE5552+I,(JAND170)OR(255-JAND85):NEXT
:rem 252
450 RETURN :rem 121
460 POKE36869,205 :rem 156
470 PRINT"{CLR}{DOWN}{CYN}{7 SPACES}LEVEL"PEEK(162
01)"{DOWN}{WHT} :rem 207
480 POKE36878,15:POKE646,9:IFB$="1"THEN500:rem 128
490 POKE36878,31:POKE646,8:POKE16288,6:POKE16289,5
:POKE16358,250:POKE16359,251 :rem 233
500 IFE$="1"THEN530 :rem 6
510 FORK=0TO70STEP10:FORJ=0TO7:POKE16285+K+J,0:NEX
T:NEXT :rem 54
520 GOSUB1210:GOSUB1210:RETURN :rem 115
530 PRINT"{3 SPACES}{RVS}Z£{OFF}Z£-}{RVS}VX{OFF}
[+][£]{RVS} ${OFF}↑[SHIFT-SPACE]{RVS}RT{OFF}
£I][£]" :rem 16
540 PRINT"{3 SPACES}{RVS}[] {OFF}+-{RVS}WY{OFF}£M]
£{RVS}#£{OFF}[*][K]{RVS}SU{OFF}[T][G]":rem 34
550 PRINT"{3 SPACES}VX{RVS}NP{OFF}VX{RVS}NP{OFF}VX
{RVS}NP{OFF}VX{RVS}NP" :rem 153
560 PRINT"{3 SPACES}WY{RVS}OQ{OFF}WY{RVS}OQ{OFF}WY
{RVS}OQ{OFF}WY{RVS}OQ" :rem 170
570 GOSUB1210 :rem 224
580 PRINT"{3 SPACES}[R][H]:<[R][H]:<[R][H]:<[R][H]
:<" :rem 222

```

# 1: Games

```
590 PRINT "{3 SPACES}[W][J];=[W][J];=[W][J];=[W][J]
;=" :rem 239
600 PRINT "{3 SPACES}_FH[L][U]_BD{RVS}BD{OFF}NP@[C]>*
[V]{RVS}@ " :rem 53
610 PRINT "{3 SPACES}_GI[Y][O]_CE{RVS}CE{OFF}OO[F][X]
?A[B]{RVS}A " :rem 70
620 POKE4173,162 :rem 91
630 IFB$="1"THENRETURN :rem 79
640 PRINT "{HOME}{3 DOWN}"SPC(9)"[N][D]{RVS}↑ "
:rem 43
650 PRINT "{13 DOWN}"SPC(9)"{RVS}FH{OFF}_JL{DOWN}":R
ETURN :rem 240
660 GETC$:IFC$=""ORFTHEN740 :rem 69
670 N=0 :rem 88
680 IFMID$(D$,N+1,1)=C$THEN710 :rem 130
690 N=N+1:IFN<13THEN680 :rem 88
700 GOTO740 :rem 107
710 J=16285+C+10*R:IFN>6THENN=262-N :rem 245
720 IFNTHENGOSUB1040:GOTO740 :rem 248
730 GOSUB990:FORI=0TO1:FORP=0TO1:POKEK+22*P+I,M:NE
XT:NEXT :rem 181
740 POKE37154,127:I=PEEK(37152)AND128:J=(I=0)
:rem 2
750 POKE37154,255:I=PEEK(37151) :rem 206
760 R=R+((IAND8)=0)-((IAND4)=0) :rem 152
770 C=C+((IAND16)=0)-J :rem 149
780 IFR<0THENR=0 :rem 218
790 IFR>7THENR=7 :rem 235
800 IFC<0THENC=0 :rem 181
810 IFC>7THENC=7 :rem 198
820 I=4473-44*R+C+C :rem 223
830 J=PEEK(I) :rem 225
840 P=56:IFJ>106THENP=-P :rem 181
850 POKEI,J+P:POKEI+22,J+P+1 :rem 148
860 POKEI+1,J+P+2:POKEI+23,J+P+3 :rem 81
870 FORP=0TO70:NEXT :rem 198
880 POKEI,J:POKEI+22,J+1 :rem 161
890 POKEI+1,J+2:POKEI+23,J+3 :rem 94
900 FORP=0TO30:NEXT :rem 188
910 IF(PEEK(37151)AND32)THEN660 :rem 244
920 J=16285+C+10*R :rem 158
930 IFFTHEN1020 :rem 99
940 IFPEEK(J)=0ORPEEK(J)>6THEN660 :rem 249
950 F=1:GOSUB980 :rem 173
960 IF(PEEK(37151)AND32)THEN660 :rem 249
970 GOTO960 :rem 120
980 POKE36876,225 :rem 163
990 K=4473-44*R+C+C:N=PEEK(J):POKEJ,0 :rem 125
1000 M=54:IF(R+C)/2-INT((R+C)/2)THENM=110 :rem 21
1010 POKE36876,0:RETURN :rem 117
```

```

1020 F=0 :rem 118
1030 FORI=0TO1:FORP=0TO1:POKEK+22*P+I,M:NEXT:NEXT :rem 131
1040 K=4473-44*R+C :rem 12
1050 M=54:IF(R+C)/2-INT((R+C)/2)THENM=110 :rem 26
1060 IFR=0ANDN=255THENN=251 :rem 97
1070 IFR=7ANDN=1THENN=5 :rem 155
1080 IFN>7THENM=M+28 :rem 182
1090 POKEJ,N:IFN>6THENN=256-N :rem 26
1100 FORI=0TO1:FORJ=0TO1:POKEK+22*J+I,M+4*N+I+I+J: :rem 169
NEXT:NEXT
1110 RETURN :rem 163
1120 IFPEEK(16202)THENI=I+1 :rem 30
1130 I=I+VAL(B$):PRINT"{DOWN}{CYN}CHECKMATE! "; :rem 245
:rem 245
1140 IFI/2-INT(I/2)THENPRINT"BLACK WINS.":GOTO1160 :rem 25
:rem 25
1150 PRINT"WHITE WINS." :rem 136
1160 POKE36876,240:FORI=0TO500:NEXT :rem 79
1170 POKE36876,195:FORI=0TO500:NEXT:POKE36876,0 :rem 44
:rem 44
1180 PRINT"{UP}PRESS JOYSTICK BUTTON.": :rem 110
1190 IF(PEEK(37151)AND32)THEN1190 :rem 84
1200 RUN :rem 184
1210 FORK=1TO2:FORJ=1TO2 :rem 231
1220 PRINT"{3 SPACES}{2 S}RR{2 S}RR{2 S}RR{2 S}RR" :rem 150
:rem 150
1230 NEXT:FORJ=1TO2 :rem 179
1240 PRINT"{3 SPACES}RR{2 S}RR{2 S}RR{2 S}RR{2 S}RR" :rem 152
:rem 152
1250 NEXT:NEXT:RETURN :rem 154
1260 DATA4,2,3,5,6,3,2,4,7 :rem 23
1270 DATA7,1,1,1,1,1,1,1,1,7 :rem 102
1280 DATA7,0,0,0,0,0,0,0,0,7 :rem 95
1290 DATA7,0,0,0,0,0,0,0,0,7 :rem 96
1300 DATA7,0,0,0,0,0,3,0,0,7 :rem 88
1310 DATA7,0,0,0,0,0,0,0,0,7 :rem 89
1320 DATA7,255,255,255,255,255,255,255,255,7 :rem 186
:rem 186
1330 DATA7,252,254,253,251,250,253,254,252 :rem 67
1340 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :rem 118
1350 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 :rem 119
1360 DATA0,0,0,0,0,3,3,0,3,0,0,3,3,0,0 :rem 135
1370 DATA0,0,0,0,192,240,240,192,240,192,192,240,2 :rem 39
40,0,0,0
1380 DATA0,48,63,63,63,15,63,63,60,60,60,0,3,15,15 :rem 26
,0
1390 DATA0,0,0,192,240,240,252,252,252,252,252,252 :rem 100
,252,252,252,0

```

# 1: Games

---

1400 DATA 0,3,15,15,15,15,15,15,15,15,0,3,0,3,63,48,0  
:rem 107

1410 DATA 0,48,204,204,204,204,252,252,252,0,240,0,  
240,63,3,0 :rem 139

1420 DATA 0,51,51,63,63,12,15,15,15,15,15,12,63,63,  
63,0 :rem 114

1430 DATA 0,204,204,252,252,48,240,240,240,240,240,  
48,252,252,252,0 :rem 197

1440 DATA 0,3,3,3,51,51,51,63,15,0,15,15,15,0,15,0  
:rem 105

1450 DATA 0,48,48,48,51,51,243,255,252,0,252,60,252  
,0,252,0 :rem 71

1460 DATA 0,0,3,0,12,63,63,63,63,0,15,15,15,0,15,0  
:rem 107

1470 DATA 0,192,240,192,204,63,255,255,255,0,252,60  
,252,0,252,0 :rem 12

# Jackpot

---

Original Program by Rick Rothstein  
VIC and 64 Versions by Kevin Mykytyn

*Now you can experience the thrill of playing slot machines without the danger of losing your money. For the unexpanded VIC or the Commodore 64.*

**H**ave you ever been to a casino in Las Vegas or Atlantic City? If so, you were probably dumbstruck by the number of slot machines waiting to take your money.

Those nefarious one-armed bandits dazzle you with bright lights and promises of instant wealth, but people don't call them bandits for nothing. A recent trip to Atlantic City—and an unprofitable encounter with some of those machines—prompted me to write "Jackpot." It will give you a taste of casino excitement without taking a bite out of your bank account.

## **Pulling the Lever**

Colorful graphics are used to display a payout chart, your current monetary standing, 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.

To play the VIC version, press the P key to pull the lever. Three shapes will appear in the windows. If you win, your total will be increased appropriately. If you lose, your total will decrease by one dollar. Press E to end the game.

On the 64 version, press 1, 2, 3, or 4 to pull the lever. Pressing 1 lets you give it a gentle tug; pressing 4 is the hardest possible pull. Again, your winnings (or losses) will be reflected in your total. As on the VIC, press E to end the game.

## **Changing the Odds**

In the VIC version, the faces of the three wheels are numerically represented in the DATA statements in lines 310–330. A 1 represents a bar. The number 2 is a seven; 3 is a bell; 4 is a cherry; 5 is a lime; 6 is a plum; and 7 is a lemon. To change the odds, simply change the numbers in the DATA statements. For example, if you change all the numbers to 1's, you will always come up with the triple bar.

The Commodore 64 version uses a machine language subroutine and colored sprites to produce a smooth spinning effect. Six different sprites are used (two for each window), and the result is remarkably realistic spinning. The different shapes are displayed by changing the sprites' data pointers. You can alter the odds by changing the numbers in the DATA statements in lines 155-165.

### Program 1. Jackpot, VIC Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

10 POKE52,28:POKE56,28:POKE51,0:POKE55,0:GOSUB300
                                     :rem 158
20 PRINT"{CLR}";:FORA=1TO4:FORB=1TO22:PRINT"@";:NE
   XT:NEXT
                                     :rem 248
30 FORA=1TO2:PRINT"@@@@@{2 SPACES}@@{2 SPACES}@@
   {2 SPACES}@@@@@";:NEXT
                                     :rem 141
40 FORA=1TO3:FORB=1TO22:PRINT"@";:NEXT:NEXT:rem 32
50 PRINTB$V$B$V$B$ "100 "V$B$V$B$"{4 SPACES}50":PR
   INTSSV$SSV$SS$"{2 SPACES}50 "V$SSV$SS$"{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}10"
                                     :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$<>"E"THEN10
   0
                                     :rem 75
110 IFA$="E"THENPRINT"{CLR}":POKE36869,240:END
                                     :rem 43
115 T$="{DOWN}{2 SPACES}{UP}{2 LEFT}{2 SPACES}
   {DOWN}":GOSUB210:GOSUB220:GOSUB220:PRINTY$ "
   {UP}{21 SPACES}"
                                     :rem 127
120 W=0:H=0:N=1:GOSUB200:GOSUB210:GOSUB260:N=2:GOS
   UB200: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$"{UP}{2 SPACES}YOU WIN"W-
   1"DOLLARS":GOSUB280
                                     :rem 187
170 TT=TT-1:IFTT>0THENTT$=STR$(TT)+"{2 SPACES}":PR
   INTY$"{4 SPACES}TOTAL NOW "TT$;:POKE198,0:GOTO
   100
                                     :rem 145
180 PRINTY$"{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$;:RETURN
:rem 54
220 PRINT"{UP}{2 RIGHT}"T$;:RETURN :rem 253
260 POKEV,150:FORA=1TO30:NEXT:POKEV,0:IFN<3THENFOR
A=1TORND(1)*200:NEXT :rem 210
270 RETURN :rem 121
280 FORQ=1TOW:TT=TT+1:TT$=STR$(TT)+"{2 SPACES}":PR
INTY$"{4 SPACES}TOTAL NOW "TT$;:POKEV1,220
:rem 220
290 FORA=1TO110-W:NEXT:POKEV1,0:NEXT:RETURN:rem 66
300 PRINT"{CLR}{3 DOWN}{2 SPACES}LOADING CHARACTER
S" :rem 8
305 DIMG%(3,17):FORA=1TO3:FORB=1TO17:READC: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 DATA4,3,44,6,444,16,555,11,661,11,666,15,331,1
1,333,19,22,26,222,51,11,51,111,101 :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}I-{DOWN}
{2 LEFT}#S":L$="{GRN}↑←{DOWN}{2 LEFT}>?" :C$="
{RED}Z[ {DOWN}{2 LEFT}:" :P$="{PUR}£]{DOWN}
{2 LEFT}<=" :rem 118
440 BE$="{YEL})*{DOWN}{2 LEFT}+,":LE$="{YEL}↑←
{DOWN}{2 LEFT}>?" :U$="{UP} " :V$="{UP}":Y$="
{HOME}{22 DOWN}" :rem 54
450 F$(1)=B$:F$(2)=S$:F$(3)=BE$:F$(4)=C$:F$(5)=L$:
F$(6)=P$:F$(7)=LE$ :rem 177
490 POKE36878,15:V=36877:V1=36876:TT=50:RETURN
:rem 244
500 DATA26,0,0,0,1,2,4,8,28 :rem 64
501 DATA27,0,0,128,128,128,64,56,124 :rem 22
502 DATA28,0,0,0,3,15,31,63,63 :rem 218
503 DATA29,0,96,192,224,240,248,252,252 :rem 182
504 DATA30,0,0,0,0,7,31,63,127 :rem 212
505 DATA31,0,0,0,0,224,248,252,254 :rem 162
506 DATA33,0,0,15,15,0,0,0,0 :rem 103

```

## 1: Games

---

```
507 DATA45,0,0,248,248,24,48,96,192      :rem 248
508 DATA35,1,3,3,3,3,3,0,0              :rem 15
509 DATA36,128,0,0,0,0,0,0,0,0          :rem 108
510 DATA37,0,0,0,255,205,128,177,170     :rem 15
511 DATA38,0,0,0,255,179,1,153,85        :rem 128
512 DATA39,179,170,178,128,205,255,0,0  :rem 133
513 DATA40,217,85,85,1,179,255,0,0      :rem 185
514 DATA41,0,0,3,7,15,15,15,15         :rem 219
515 DATA42,0,0,128,192,224,224,224,224,43,15,15,31,63,63,63,1,0 :rem 49
517 DATA44,224,224,240,248,248,248,128,0,58,62,127,127,127,62,28,0 :rem 181
519 DATA59,254,254,254,254,124,56,0,0,60,63,63,63,63,31,15,3,0 :rem 24
521 DATA61,252,252,252,252,248,240,224,0,62,255,127,63,31,7,0,0,0 :rem 151
523 DATA63,255,254,252,248,224,0,0,0,0,255,255,255,255,255,-1 :rem 57
```

## Program 2. Jackpot, 64 Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
5 PRINT"{CLR}";:POKE51,0:POKE55,0:POKE52,48:POKE56,48:CLR:GOSUB 60 :rem 61
10 TT=50:S=54272:FORL=STOS+24:POKEL,0:NEXT:rem 135
15 POKES+1,112:POKES+5,9:POKES+15,208:POKES+24,15 :rem 119
20 POKES+8,150:POKES+12,8:POKES+13,0 :rem 94
25 PRINT"{CLR}";:FOR A=1TO5:FORB=1TO40:PRINT"{RVS} ";:NEXTB,A :rem 6
30 FOR B=1TO5:PRINT"{RVS}{9 SPACES}{OFF}{6 SPACES}{RVS}{2 SPACES}{OFF}{6 SPACES}{RVS}{2 SPACES}{OFF}{6 SPACES}{RVS}{9 SPACES}";:NEXT :rem 143
35 FORA=1TO5:FORB=1TO40:PRINT"{RVS} ";:NEXTB,A:POKE53269,255 :rem 104
40 PRINT:PRINT"{RED} ({8 SPACES}2{4 SPACES}{RED}({SPACE}({6 SPACES}5{4 SPACES}{GRN}&&{RED}%{3 SPACES}10" :rem 31
45 PRINT:PRINT"{GRN} &{SHIFT-SPACE}&{SHIFT-SPACE}&{3 SPACES}10{4 SPACES}{YEL}&{SHIFT-SPACE}&{SHIFT-SPACE}&{3 SPACES}10{4 SPACES}{PUR}'{SHIFT-SPACE}' {RED}%{3 SPACES}14" :rem 19
50 PRINT:PRINT"{PUR} '{SHIFT-SPACE}'{SHIFT-SPACE}'{SHIFT-SPACE}{2 SPACES}14{4 SPACES}{RED}({SHIFT-SPACE}({SHIFT-SPACE}({3 SPACES}15{4 SPACES}{YEL}#{SHIFT-SPACE}#{SHIFT-SPACE}{RED}%{3 SPACES}18" :rem 91
```

```

55 PRINT:PRINT"{YEL} #{SHIFT-SPACE}#{SHIFT-SPACE}#
   {3 SHIFT-SPACE}18{4 SPACES}{RED}$#{SHIFT-SPACE}$
   {SHIFT-SPACE}$ {3 SPACES}50{4 SPACES}{RED}%
   {SHIFT-SPACE}%{SHIFT-SPACE}%{2 SPACES}100":GOTO
   400 :rem 34
60 POKE 53281,1:POKE53275,255:D$="{HOME}{23 DOWN}"
   :rem 233
65 PRINT"{3 DOWN}{BLK}{RVS}{11 RIGHT}ONE ARMED BAN
   DIT" :rem 77
70 PRINT "{2 DOWN} YOU WILL BEGIN WITH $50 AND TRY
   TO" :rem 46
75 PRINT"{DOWN}{8 SPACES}TURN IT INTO A FORTUNE."
   :rem 0
80 PRINT"{DOWN} IT WILL COST YOU $1 FOR EACH PULL.
   " :rem 13
85 PRINT"{DOWN} TO PULL THE HANDLE USE THE KEYS 1-
   4." :rem 134
90 PRINT"{DOWN} THE HIGHER THE NUMBER, THE HARDER
   {SPACE}THE :rem 15
95 PRINT"{DOWN}{13 SPACES}PULL WILL BE." :rem 122
100 PRINT"{DOWN} TO STOP THE GAME AT ANY TIME PRES
   S (E)" :rem 38
105 PRINT"{2 DOWN} PLEASE WAIT WHILE I LOAD THE SP
   RITES" :rem 101
110 DIM WIN$(24) :rem 53
115 FOR A=1TO24:READWIN$(A):NEXT :rem 169
120 DATA CHERRY,2,CHERRYCHERRY,5,LIMELIMEBAR,10,LIM
   ELIMELIME,10 :rem 9
125 DATA PLUMPLUMBAR,14,PLUMPLUMPLUM,14,BELLBELLBAR
   ,18 :rem 223
130 DATABELLBELLBELL,18,SEVENSEVENSEVEN,50,BARBAB
   AR,100 :rem 72
135 DATA LEMONLEMONLEMON,10,CHERRYCHERRYCHERRY,15
   :rem 214
140 M=2047:NN=12288:OO=53248:S=54272:FORL=STOS+24:
   POKEL,.:NEXT :rem 249
145 POKE S+5,9:POKES+6,0:POKES+24,15:POKES+1,120
   :rem 16
150 FORI=49664TO49714:READB:B=B+239:POKEI,B:NEXT
   :rem 177
155 DATA 4,2,3,7,5,6,7,6,3,1,5,5,7,4,6,7,6:rem 249
160 DATA 1,2,5,4,5,6,7,4,3,7,5,6,7,2,6,7,3:rem 241
165 DATA 1,2,3,4,5,6,7,7,3,4,5,6,7,5,6,7,4:rem 248
170 FORI=830TO833:POKEI,0:NEXT :rem 104
175 AA=15360:BB=15807:CC=12568:DD=12615 :rem 92
180 FOR A=AA TO BB:READB:IF B=>. THEN POKEA,B:GOTO
   190 :rem 13
185 D=ABS(B)-1:FORC=. TO D:POKE A+C,.:NEXT:A=A+D
   :rem 239

```

# 1: Games

---

```
190 NEXT :rem 217
195 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)AND25
1:FOR I=. TO M :rem 133
200 POKE NN+I,PEEK(OO+I):NEXT:POKE 1,PEEK(1)OR4:PO
KE56334,PEEK(56334)OR1 :rem 39
205 PRINT"{CLR}":FOR I=CC TO DD:READB:POKE I,B:NEX
T :rem 132
210 POKE53272,(PEEK(53272)AND240)OR12 :rem 40
215 POKE53249,48:POKE53251,90:POKE53253,48:POKE532
55,90 :rem 105
220 POKE53257,48:POKE53259,90 :rem 103
225 FORQ=2041TO2045STEP2:POKEQ,240:NEXT :rem 166
230 POKE53248,97:POKE53250,97:POKE53252,160:POKE53
254,160:POKE53256,223 :rem 247
235 POKE53258,223:POKE53271,255:POKE53277,255
:rem 255
240 U=0:A=49152:B=49475:FORI=ATOB:READC::U=U+C:POK
EI,C:NEXT:IFU=38200THENRETURN :rem 11
245 PRINT "ERROR IN DATA STATEMENTS 645-845!":END
:rem 78
250 FORA=679TO685:POKEA,INT(RND(1)*16)+1:NEXT
:rem 198
255 PRINTD$"{35 SPACES}":SYS 49152:POKE S+11,128
:rem 35
260 SPIN$="":FORB=2041TO2045STEP2:Q=PEEK(B)-239
:rem 5
265 ONQGOSUB365,370,375,380,385,390,395 :rem 98
270 NEXT:WIN=0 :rem 109
275 IFSPIN$="BARBARBAR"THENGOSUB 850 :rem 215
280 FORA=1TO24:L=LEN(WIN$(A)):IFLEFT$(SPIN$,L)=WIN
$(A)THENWIN=VAL(WIN$(A+1)) :rem 126
285 NEXT:TT=TT-1 :rem 3
290 IF WIN<>0THENPRINTD$"{10 SPACES}YOU WIN "WIN"
{SPACE}DOLLARS"; :rem 147
295 IF WIN>5 THENPRINT"!"; :rem 236
300 IFWIN=0THEN325 :rem 70
305 POKE S,80:POKES+1,112:POKES+15,208 :rem 153
310 FORTT=TT+1TOTT+WIN-1:POKES+4,21:FORTD=1TO150-W
IN:NEXT :rem 196
315 T$=STR$(TT):PRINTD$"{DOWN}{5 SPACES}YOUR TOTAL
IS NOW "T$" DOLLARS{2 SPACES}"; :rem 83
320 POKES+4,20:FORT=1TO150:NEXT:NEXT:POKES+1,0:POK
ES+15,0 :rem 47
325 T$=STR$(TT):PRINTD$"{DOWN}{5 SPACES}YOUR TOTAL
IS NOW "T$" DOLLARS{2 SPACES}"; :rem 84
330 IF TT>0 THEN 360 :rem 3
335 PRINTD$"{8 SPACES}SORRY BUT YOU'RE BROKE"
:rem 140
340 PRINT"{3 SPACES}DO YOU WANT TO PLAY AGAIN?
{2 SPACES}Y/N{4 SPACES}"; :rem 187
```

```

345 GETA$:IFAŞ<>"Y"ANDAŞ<>"N"THEN345           :rem 53
350 IF AŞ="Y"THEN 10                               :rem 246
355 POKE 53269,0:PRINT"{CLR}":END                 :rem 224
360 POKE 198,0:GOTO 400                             :rem 205
365 SPIN$=SPIN$+"BAR":RETURN                       :rem 245
370 SPIN$=SPIN$+"SEVEN":RETURN                    :rem 157
375 SPIN$=SPIN$+"BELL":RETURN                     :rem 64
380 SPIN$=SPIN$+"CHERRY":RETURN                   :rem 234
385 SPIN$=SPIN$+"LIME":RETURN                     :rem 73
390 SPIN$=SPIN$+"PLUM":RETURN                     :rem 92
395 SPIN$=SPIN$+"LEMON":RETURN                    :rem 158
400 GETA$:IFAŞ="E"THEN PRINT"{CLR}":POKE 53269,0:E
    ND                                              :rem 63
405 A=RND(1)                                       :rem 125
410 IF AŞ<"1" OR AŞ>"4" THEN 400                 :rem 185
415 POKE 49238,VAL(AŞ):GOTO 250                    :rem 162
420 DATA-12,255,255,255,204                       :rem 90
425 DATA204,205,128,0,3,158,28,115               :rem 179
430 DATA209,34,73,209,34,73,158,62              :rem 195
435 DATA115,145,34,75,209,34,73,222            :rem 240
440 DATA34,73,128,0,3,153,153,179               :rem 136
445 DATA255,255,255,-22                          :rem 160
450 DATA3,255,240,3,255,240,3                   :rem 184
455 DATA0,112,0,0,224,0,1,192                   :rem 165
460 DATA0,3,128,0,14,0,0,28                     :rem 69
465 DATA0,0,56,0,0,112,0,0                      :rem 12
470 DATA112,0,0,112,0,0,112,0                  :rem 149
475 DATA0,112,-21,60                             :rem 248
480 DATA0,0,255,0,1,255,128,3                   :rem 177
485 DATA255,192,3,255,192,3,255,192            :rem 253
490 DATA3,255,192,3,255,192,3,255              :rem 144
495 DATA192,7,255,224,15,255,240,63            :rem 248
500 DATA255,252,127,255,254,127,255,254       :rem 186
505 DATA127,255,254,0,24,-18,48                 :rem 37
510 DATA0,0,80,0,0,136,0,1                      :rem 7
515 DATA4,0,2,2,0,2,1,240                        :rem 218
520 DATA15,129,248,31,195,252,63,227           :rem 37
525 DATA252,63,227,252,063,225,248,63          :rem 89
530 DATA224,240,31,192,0,15,128,-27            :rem 219
535 DATA255,0,3,255,192,15                       :rem 48
540 DATA255,240,31,255,248,63,255,252          :rem 85
545 DATA127,255,254,255,255,255,127,255        :rem 199
550 DATA254,63,255,252,31,255,248,15           :rem 37
555 DATA255,240,3,255,192,0,255,-16,255       :rem 178
560 DATA0,0,0,0,2,0,0,4                         :rem 111
565 DATA0,0,8,0,0,60,0,0                        :rem 172
570 DATA255,0,3,255,192,7,255,224              :rem 140
575 DATA15,255,240,15,255,240,15,255           :rem 33
580 DATA240,15,255,240,7,255,224,3             :rem 183
585 DATA255,192,0,255,0,0,60,-16,255,-10      :rem 208

```

# 1: Games

---

590 DATA255,0,3,255,192,15 :rem 49  
595 DATA255,240,31,255,248,63,255,252 :rem 95  
600 DATA127,255,254,255,255,255,127,255 :rem 191  
605 DATA254,63,255,252,31,255,248,15 :rem 38  
610 DATA255,240,3,255,192,0,255,-16,255 :rem 170  
615 DATA24,60,60,60,126,255,255,24 :rem 185  
620 DATA127,3,6,12,24,24,24,0 :rem 174  
625 DATA0,255,129,255,255,129,255,0 :rem 243  
630 DATA0,0,60,126,255,126,60,0 :rem 17  
635 DATA4,8,60,126,255,126,60,0 :rem 34  
640 DATA8,20,38,111,255,246,96,0 :rem 87  
645 DATA169,0,141,176,2,141,177,2 :rem 138  
650 DATA141,178,2,173,176,2,208,16 :rem 189  
655 DATA206,167,2,208,11,173,168,2 :rem 190  
660 DATA141,167,2,162,0,32,234,192 :rem 179  
665 DATA173,177,2,208,16,206,169,2 :rem 198  
670 DATA208,11,173,170,2,141,169,2 :rem 180  
675 DATA162,4,32,234,192,173,178,2 :rem 196  
680 DATA208,16,206,171,2,208,11,173 :rem 231  
685 DATA172,2,141,171,2,162,8,32 :rem 84  
690 DATA234,192,238,173,2,208,188,238 :rem 100  
695 DATA61,3,173,61,3,201,2,208 :rem 32  
700 DATA178,169,0,141,61,3,169,128 :rem 192  
705 DATA141,11,212,238,168,2,208,5 :rem 180  
710 DATA169,255,141,168,2,173,168,2 :rem 245  
715 DATA201,112,144,22,173,176,2,208 :rem 20  
720 DATA17,173,1,208,201,48,208,10 :rem 175  
725 DATA169,129,141,11,212,169,1,141 :rem 28  
730 DATA176,2,238,170,2,208,5,169 :rem 142  
735 DATA255,141,170,2,173,170,2,201 :rem 225  
740 DATA112,144,22,173,177,2,208,17 :rem 232  
745 DATA173,5,208,201,48,208,10,169 :rem 242  
750 DATA129,141,11,212,169,1,141,177 :rem 25  
755 DATA2,238,172,2,208,5,169,255 :rem 149  
760 DATA141,172,2,173,172,2,201,112 :rem 219  
765 DATA144,22,173,178,2,208,17,173 :rem 247  
770 DATA9,208,201,48,208,10,169,129 :rem 245  
775 DATA141,11,212,169,1,141,178,2 :rem 183  
780 DATA24,173,176,2,109,177,2,109 :rem 195  
785 DATA178,2,201,3,240,3,76,11 :rem 34  
790 DATA192,96,160,2,254,1,208,189 :rem 202  
795 DATA1,208,201,130,208,62,169,194 :rem 37  
800 DATA133,252,152,72,138,72,74,141 :rem 29  
805 DATA80,3,74,170,189,65,193,133 :rem 202  
810 DATA251,254,62,3,189,62,3,201 :rem 133  
815 DATA17,208,5,169,0,157,62,3 :rem 43  
820 DATA168,177,251,174,80,3,157,248 :rem 48  
825 DATA7,56,233,240,168,185,58,193 :rem 4  
830 DATA157,39,208,104,170,104,168,169 :rem 141  
835 DATA48,157,1,208,232,232,136,208 :rem 36

```
840 DATA179,96,2,2,7,2,5,4           :rem 48
845 DATA7,0,17,34                     :rem 117
850 B=0:POKES+5,9:POKES+6,9           :rem 79
855 FORA=1TO130:POKE53281,A:POKE53280,256-A:B=-(B=
    0):POKE53271,255-255*B             :rem 56
860 POKE 53277,255-255*B:POKES+1,A:POKES+4,33:FORT
    D=1TO20:POKES+4,32                 :rem 220
865 FORTD=1TO20:NEXT:NEXT:POKES+4,32:POKES+1,0:POK
    ES-992,6:POKES-991,1:RETURN        :rem 153
```

# Nirrad's Labyrinth

Darrin Mossor

*Hidden gold, invisible trap doors, and a fearsome creature named Boogens add excitement to this treasure-hunting adventure for the unexpanded VIC or 64. A joystick is required.*

**T**he legends leave no doubt about it: Centuries ago, the evil wizard Nirrad stole all of the gold from your village. He hid it in leather bags in a huge, twisting, tortuous labyrinth, and for each bag he also installed a hidden trap door to trap unwary heroes. As if that was not enough, he created a horrifying demon guard called Boogens whose only purpose in life is to eat gullible adventurers.

For many generations, the triple threat of labyrinth, trap doors, and Boogens has been more than enough to scare away even the bravest of adventurers. But now you have discovered a chilling secret: It is *your* destiny to recover the gold.

All you have to do is feel your way through the darkened maze, with its walls that you can't even see until you bump into them, while looking out for trap doors and dodging that doggoned Boogens. The task seems hopeless.

But what good is a hero and adventurer without a hopeless task now and then?

## Into the Labyrinth

When you run the program, you'll be asked how many bags of gold you want to try for. Next, you will be given the option of challenging the Boogens to double the value of your gold. Then, when you've entered your choices, the screen will turn blue while the computer builds the maze. It takes about 30 seconds for the VIC to create the labyrinth; the 64 takes roughly a minute.

When construction is finished, the computer sounds a tone to alert you. It places S at the start and an F at the finish, and it shows you where the gold is. It also places the trap doors and the walls, but you won't see those until it's too late!

When you have not challenged Boogens, the only pressure is to complete the maze and escape with as much gold as possible. But if you choose to challenge Boogens, beware of one thing. Although his incessant hopping appears to be ran-



dom, he possesses an uncanny ability to land on you at the worst possible times.

The trap doors can be a help or a hindrance. Sometimes one of them will teleport you to another part of the labyrinth at the worst possible moment; at other times, you may have to jump into one on purpose to reach a certain bag of gold or to escape from Boogens.

You can choose to try for 99 bags of gold, but you had better have a lot of free time if you do. It is extremely difficult to get all 99 bags.

## Loading the Game

The VIC version of Nirrad is in two parts. Program 1 loads the special character set. After you type it in, save it *before* you run it. Tape users should change the ,8 to ,1 in line 109. Save Program 2 as LABYR.VIC2. Tape users should be sure to save LABYR.VIC2 *after* Program 1.

The 64 version is a single program, but it should still be saved before being run. To play the 64 version, plug your joystick into port 2.

## Program 1. VIC Loader

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

10 PRINT"{CLR}":POKE36879,8           :rem 163
12 GOTO115                             :rem 51
15 POKE36869,255                       :rem 109
20 A$="{2 RIGHT}{RVS}{CYN}NIRRAD'S LABYRINTH"
                                         :rem 26
30 FORX=20TO1STEP-1                    :rem 175
40 PRINT"{HOME}";:FORY=1TOX:PRINT"{DOWN}";:NEXTY:P
   RINTA$                                :rem 187
45 FORZ=1TO10*X/2:NEXTZ                 :rem 210
50 PRINT"{CLR}":NEXTX                   :rem 154
55 PRINT"{2 RIGHT}{CYN}{RVS}NIRRAD'S LABYRINTH"
                                         :rem 13
60 POKE36878,15:POKE36876,200:FORX=1TO700:NEXTX:PO
   KE36876,0                              :rem 49
65 B=7902:C=30720:M=4:G=1:S=36876     :rem 49
70 FORN=BTOB+19                         :rem 149
78 POKEN-1,32:POKEN-3,32                :rem 131
80 POKEN,M:POKEN+C,5:POKEN-2,G:POKEN-2+C,4 :rem 81
82 POKES,240                             :rem 174
85 FORP=1TO50:NEXTP                     :rem 227
87 POKES,0                               :rem 77
90 NEXTN                                 :rem 246

```

## 1: Games

---

```
92 POKEN-1,32:POKEN-3,32           :rem 127
93 POKE36869,240                   :rem 109
94 PRINT"{CLR}{2 RIGHT}{DOWN}{CYN}NIRRAD'S LABYRIN
   TH"                               :rem 162
108 POKE198,3:POKE631,13:POKE 632,13:POKE 633,13:P
   RINT"{CLR}{BLK}{3 DOWN}LO";CHR$(34);"LABYR.VIC
   2";                               :rem 107
109 PRINTCHR$(34);",8":PRINT "{5 DOWN}FORX=1TO5000
   :NEXT":PRINT"{2 DOWN}RUN"        :rem 211
110 PRINT"{3 DOWN}{CYN}{5 RIGHT}EXECUTING.....
   {BLK}{HOME}":END                :rem 12
115 POKE52,28:POKE56,28:CLR        :rem 72
120 FORX=7168TO7223:READA:POKEX,A:NEXTX :rem 236
130 FORX=7423TO7431:POKEX,0:NEXTX:CLR:GOTO15
                                     :rem 52
200 DATA255,129,129,255,255,129,129,255,195,129,15
   3,255,219,126,60,102,60,24,189  :rem 176
210 DATA231,231,189,24,60,255,195,165,153,153,165,
   195,255,56,56,146,124,16       :rem 130
215 DATA 40,40,68,60,66,64,60,2   :rem 240
220 DATA66,60,0,126,64,64,120,64,64,64,0 :rem 214
```

## Program 2. VIC Main Program

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
40 DATA-23,-22,-21,-1,0,1,21,22,23 :rem 124
44 DIMJS(2,2),A(3):POKE37139,0:DD=37154:PA=37137:P
   B=37152                             :rem 152
46 FORI=0TO2:FORJ=0TO2:READJS(J,I):NEXTJ,I:rem 196
50 SC=7680:CL=30720:CH=4:GB=1:V=36878:S=36876:POKE
   V,15:BP=7703                         :rem 86
55 A(0)=2:A(1)=-44:A(2)=-2:A(3)=44:WL=0:HL=32:A=SC
   +23:POKE36879,27                     :rem 79
60 PRINT"{CLR}{2 RIGHT}{GRN}NIRRAD'S LABYRINTH
   {2 SPACES}{2 DOWN}{BLU}{2 SPACES}WELCOME, YE WH
   O HAVE{DOWN}COME";                 :rem 162
65 PRINT " TO CHOOSE YOUR{2 SPACES}{DOWN} FATE."
                                     :rem 78
70 PRINT"{2 DOWN}{2 SPACES}HOW MUCH GOLD WILL
   {2 SPACES}{DOWN}YOU TRY FOR NOW, BUT{2 SPACES}
   {DOWN}REMEMBER YOU WILL HAVE"      :rem 139
80 PRINT"AN EQUAL NUMBER OF{DOWN}":INPUT"TRAPDOORS
   (1-99)":U$:U=VAL(U$)                :rem 201
82 IFU<LORU>99ORU<>INT(U)THEN60       :rem 247
88 PRINT"{CLR}{2 DOWN}{RED}WOULD YOU LIKE TO
   {5 SPACES}{DOWN}CHALLENGE THE BOUNCING{DOWN}BOO
   GENS TO DOUBLE THE"                :rem 137
90 PRINT"{DOWN}VALUE OF YOUR GOLD?{3 SPACES}{DOWN}
   (Y/N){BLU}"                         :rem 104
92 GETC$:IFC$=""THEN92                :rem 253
```

```

94 IFC$="Y"THENYG=1 :rem 122
120 POKE36869,255:POKE36879,110:PRINT"{CLR}";:FORI
=1TO22:PRINT"@@@@@@@@@@@@@@@@@@@@@@" :rem 179
125 NEXT I :rem 32
130 PRINT"{HOME}{WHT}{RVS}PLEASE BE PATIENT ...
{BLU}" :rem 142
140 J=INT(RND(1)*4):X=J :rem 53
150 B=A+A(J):IFPEEK(B)=WLTHENPOKEB,J+1:POKEA+A(J)/
2,HL:A=B:GOTO140 :rem 4
160 J=-(J+1)*(J<3):IFJ<>XTHEN150 :rem 32
170 J=PEEK(A):POKEA,HL:IFJ<5THENA=A-A(J-1):GOTO140
:rem 33
190 PRINT"{21 DOWN}@@@@@@@@@@@@@@@@@{WHT}F{BLU}@
{HOME}" :rem 77
200 PRINT"{HOME}@{WHT}E{OFF}{BLU}@@@@@@@@@@@@@@@@@
@@@@" :rem 241
220 POKEBP,CH:POKEBP+CL,5 :rem 34
225 FORR=1TO200:POKES,200:NEXTR:POKES,0 :rem 210
230 FORW=1TOU*2 :rem 149
240 X=INT(RND(1)*21):Y=INT(RND(1)*22) :rem 90
250 IFPEEK(7680+22*Y+X)<>32THEN240 :rem 170
260 IFW/2<>INT(W/2)THENPOKESC+22*Y+X,2:POKESC+22*Y
+X+CL,7:NEXTW :rem 181
265 POKESC+22*Y+X,3:NEXTW :rem 38
270 IFYGTHENGOSUB600 :rem 13
275 POKEBP,CH:POKEBP+CL,5:GOSUB1000:PK=PEEK(BP+JSN
) :rem 215
300 IFPK=32THENBP=BP+JSN:FORR=1TO10:POKES,240:NEXT
R:POKES,0:POKEBP-JSN,32:GOTO270 :rem 251
310 IFPK=2THEN 315 :rem 245
312 GOTO 320 :rem 100
315 TR=TR+1:POKEBP,32:POKEBP+JSN+CL,6:BP=BP+JSN:FO
RR=200TO240:POKES,R :rem 157
317 NEXTR:POKES,0:GOTO270 :rem 80
320 IFPK=0THENPOKEBP+JSN+CL,1:FORR=1TO10:POKES,140
:NEXTR:POKES,0 :rem 142
330 IFPK=6THEN2000 :rem 36
340 IFPK=1THEN3000 :rem 33
350 IFPK=3THENPOKEBP+JSN+CL,2:X=INT(RND(1)*21):Y=I
NT(RND(1)*22):POKEBP,32:GOTO400 :rem 225
360 GOTO270 :rem 107
400 IFPEEK(SC+X+22*Y)=2THEN440 :rem 250
405 IFPEEK(SC+X+22*Y)<>32THEN350 :rem 111
410 POKEBP,32:POKEBP+CL,6:FORR=160TO200:POKES,R:NE
XTR:POKE36879,62:FORR=1TO500:NEXTR :rem 113
420 POKE36879,110:FORR=200TO160STEP-1:POKES,R:NEXT
R:POKES,0:BP=SC+X+22*Y:GOTO270 :rem 7
440 FORR=160TO200:POKES,R:NEXTR:POKE36879,62:FORR=
1TO500:NEXTR:POKE36879,110 :rem 9

```

# 1: Games

---

```
450 FORR=200TO160STEP-1:POKES,R:NEXTR:POKES,0           :rem 146
460 POKEBP,32:POKEBP+CL,6:FORR=200TO240:POKES,R:NE
    XTR:POKES,0:BP=SC+X+22*Y:GOTO270                       :rem 219
600 IFRND(1)>.3THENRETURN                                   :rem 57
605 POKEGS,32                                              :rem 245
610 X=INT(RND(1)*21):Y=INT(RND(1)*22):PS=SC+X+22*Y
                                                :rem 160
615 IFPEEK(PS)=CHTHEN3000                                  :rem 254
620 IFPEEK(PS)<>32THEN600                                   :rem 228
630 POKEPS,1:POKEPS+CL,4:FORR=1TO20:POKES,130:NEXT
    R:POKES,0:GS=PS:RETURN                                  :rem 1
1000 POKEDD,127:S3=-((PEEK(PB)AND128)=0):POKEDD,25
    5                                                         :rem 166
1010 P=PEEK(PA):S1=-((PAND8)=0):S2=((PAND16)=0):S0
    =((PAND4)=0)                                           :rem 221
1020 FR=-((PAND32)=0):X1=S2+S3:Y1=S0+S1:JSN=JS(X1+
    1,Y1+1):RETURN                                         :rem 38
2000 POKE36879,62:POKE36869,240:PRINT"{CLR}"
                                                :rem 111
2010 FORR=130TO240:POKES,R:NEXTR:POKES,0                 :rem 35
2020 PRINT"{CLR}{2 DOWN}CONGRATULATIONS!! YOU"
                                                :rem 6
2025 PRINT "{DOWN}HAVE DONE WHAT FEW{4 SPACES}
    {DOWN}OTHER MORTALS HAVE AT-{DOWN}TEMPTED. YO
    U ALSO HAVE"                                           :rem 72
2030 IFYGTENPRINT"DEFEATED THE BOOGENS, {DOWN}AND
    THEREFORE RECEIVE"                                     :rem 203
2035 IFYGTENPRINT"{DOWN}";TR*200;"GOLD PIECES.":G
    OTO2050                                                 :rem 109
2040 PRINTTR*100;"GOLD PIECES"                            :rem 18
2050 PRINT"{DOWN}WOULD YOU LIKE TO TRY{DOWN} AGAIN
    ?? (Y/N)"                                              :rem 14
2080 GETD$:IFD$=""THEN2080                                 :rem 189
2090 IFD$="Y"THENPOKE36879,27:PRINT"{CLR}":RUN
                                                :rem 109
2100 END                                                  :rem 154
3000 POKEPS,1:POKEPS+CL,4:POKES,170:FORR=1TO400:NE
    XTR:POKES,130:FORR=1TO400                               :rem 15
3010 NEXTR:POKES,0                                         :rem 109
3015 POKE36869,240:PRINT"{CLR}":POKE36879,56
                                                :rem 121
3020 PRINT"{CLR}{4 DOWN}THE BOOGENS GOT YOU!!!
    {DOWN}TOO BAD... TRY AGAIN?"                          :rem 151
3040 GETB$:IFB$=""THEN3040                                 :rem 179
3050 IFB$="N"THENEND                                       :rem 146
3060 POKE36879,27:PRINT"{CLR}":RUN                       :rem 107
```

### Program 3. Nirrad's Labyrinth, 64 Version

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

8 POKE53272,21 :rem 249
10 POKE52,48:POKE56,48:CLR:FORX=12288TO12288+55:RE
ADA:POKEX,A:NEXTX :rem 164
15 FORX=12544TO12551:POKEX,0:NEXTX :rem 113
20 DATA 255,129,129,255,255,129,129,255,195,129,15
3,255,219,126,60,102,60 :rem 32
25 DATA 24,189 :rem 181
30 DATA 231,231,189,24,60,255,195,165,153,153,165,
195,255,56,56,146,124,16 :rem 82
35 DATA 40,40 :rem 118
40 DATA 68,60,66,64,60,2,66,60,0,126,64,64,120,64,
64,64,0 :rem 240
50 SC=1024:CL=54272:CH=4:GB=1:BP=SC+81 :rem 163
52 S1=54272 :rem 44
55 A(0)=2:A(1)=-80:A(2)=-2:A(3)=80:WL=0:HL=32:A=SC
+81:POKE53280,6:POKE53281,1 :rem 218
60 PRINT"{CLR}{GRN}{4 DOWN}{11 RIGHT}NIRRAD'S LABY
RINTH" :rem 82
62 PRINT"{3 DOWN}{BLU}{2 SPACES}WELCOME, YE WHO HA
VE COME TO CHOOSE" :rem 251
64 PRINT"{DOWN}{14 RIGHT}YOUR FATE." :rem 127
70 PRINT"{5 DOWN}HOW MANY BAGS OF GOLD WILL YOU TR
Y FOR?" :rem 226
72 PRINT"{2 RIGHT}{DOWN}REMEMBER, YOU GET THE SAME
NUMBER OF" :rem 66
74 INPUT"{DOWN}{8 RIGHT}OF TRAPDOORS (1-99)";A$
:rem 76
76 U=VAL(A$):IFU<1ORU>99THEN60 :rem 94
78 PRINT"{CLR}":PRINT"{3 DOWN}{2 RIGHT}YOU ARE LOO
KING FOR"U"BAGS OF GOLD." :rem 185
79 FOR ZZ=1TO2000:NEXT :rem 87
80 PRINT"{3 DOWN}{RED}WOULD YOU LIKE TO CHALLENGE
{SPACE}THE BOUNCING" :rem 145
85 PRINT"{2 RIGHT}BOGENS TO DOUBLE THE VALUE OF Y
OUR" :rem 37
90 PRINTTAB(17);"{DOWN}GOLD?" :rem 123
92 GETC$:IFC$=""THEN92 :rem 253
95 PRINT"{CLR}":PRINT"{3 DOWN}I'M GOING TO CREATE
{SPACE}A SPECIAL LABYRINTH":FOR ZZ=1TO3000
:rem 48
97 NEXT :rem 175
99 PRINT"{CLR}":PRINT"{10 DOWN}{16 RIGHT}GOOD LUCK
!":FOR ZZ=1TO3000:NEXT :rem 246
110 IFC$="Y"THENYG=1 :rem 159
120 PRINT"{CLR}{BLU}" :POKE53272,(PEEK(53272)AND240
)+12:POKE53280,6:POKE53281,6 :rem 10
125 PRINT;:POKE1024,2:POKE1024+CL,6 :rem 58

```

# 1: Games

---

```
130 FORI=1TO23:PRINT"@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@@@@@@@@@@@@@":NEXTI :rem 199
135 POKEA,5 :rem 106
140 J=INT(RND(1)*4):X=J:GOSUB200000 :rem 225
152 B=A+A(J):IFPEEK(B)=WLTHENPOKEB,J+1:POKEA+A(J)/
2,HL:A=B:GOTO140 :rem 6
160 J=-(J+1)*(J<3):IFJ<>XTHEN152 :rem 34
170 J=PEEK(A):POKEA,HL:IFJ<5THENA=A-A(J-1):GOTO140
:rem 33
195 POKESC+41,5:POKESC+41+CL,1:POKE1981,6:POKE1981
+CL,1:POKE1024+CL,6 :rem 56
200 HN=16:LN=195:DR=300:WF=17:GOSUB100000 :rem 67
230 FORW=1TOU*2 :rem 149
240 X=INT(RND(1)*38):Y=INT(RND(1)*22+1) :rem 190
250 IFPEEK(1024+40*Y+X)<>32THEN240 :rem 156
260 IFW/2<>INT(W/2)THENPOKESC+40*Y+X,2:POKESC+40*Y
+X+CL,7:NEXTW :rem 181
265 POKESC+40*Y+X,3:POKESC+40*Y+X+CL,6:NEXTW
:rem 214
270 IFYGTENGOSUB600 :rem 13
275 POKEBP,CH:POKEBP+CL,5:GOSUB10000:PK=PEEK(BP+JN)
:rem 132
300 IFPK=32THENBP=BP+JN:POKEBP-JN,32 :rem 3
305 IFPK=32THENHN=33:LN=135:WF=17:DR=25:GOSUB100000
:GOTO270 :rem 29
310 IFPK=2THEN 312 :rem 242
311 GOTO 320 :rem 99
312 TR=TR+1:POKEBP,32:POKEBP+JN+CL,6:BP=BP+JN:FORR
=200TO240:GOSUB120000 :rem 103
320 IFPK=0THENPOKEBP+JN+CL,1:HN=67:LN=15:WF=17:DR=
15:GOSUB100000 :rem 129
330 IFPK=6THEN20000 :rem 36
340 IFPK=1THEN30000 :rem 33
345 IFPK=3THEN350 :rem 253
346 GOTO270 :rem 111
350 IFPK=3THENPOKEBP+JN+CL,0:X=INT(RND(1)*38):Y=IN
T(RND(1)*22+1):POKEBP,32 :rem 233
360 IFPEEK(SC+X+40*Y)=2THENGD=1:GOTO410 :rem 104
400 IFPEEK(SC+X+40*Y)<>32THEN350 :rem 106
410 POKEBP,32:POKEBP+CL,0 :rem 248
415 GOSUB170000 :rem 18
420 POKE53281,15:FORR=1TO200:NEXTR:BP=SC+X+Y*40:GO
SUB170300 :rem 36
425 POKE53281,6 :rem 47
430 IFGD=1THENTR=TR+1:GOSUB120000:GD=0 :rem 18
440 GOTO270 :rem 106
600 IFRND(1)>.3THENRETURN :rem 57
605 POKEGS,32 :rem 245
610 X=INT(RND(1)*38):Y=INT(RND(1)*22+1):PS=SC+X+40
*y :rem 4
```

```
615 IFPEEK(PS)=CHTHEN3000 :rem 254
620 IFPEEK(PS)<>32THEN600 :rem 228
630 POKEPS,1:POKEPS+CL,4:GS=PS :rem 162
640 HN=8:LN=97:WF=33:DR=100:GOSUB10000 :rem 233
650 RETURN :rem 123
1000 JI=PEEK(56320)AND15 :rem 64
1035 IFJI=15THENJN=0 :rem 194
1040 IFJI=14THENJN=-40 :rem 30
1045 IFJI=6THENJN=-39 :rem 252
1050 IFJI=7THENJN=1 :rem 145
1055 IFJI=5THENJN=41 :rem 200
1060 IFJI=13THENJN=40 :rem 242
1065 IFJI=9THENJN=39 :rem 212
1070 IFJI=11THENJN=-1 :rem 235
1075 IFJI=10THENJN=-41 :rem 35
1080 RETURN :rem 169
1500 FORZZ=S1TOS1+23:POKEZZ,0:NEXT:RETURN :rem 229
2000 GOSUB18000:PRINT"{CLR}":POKE53281,1:POKE53272
,21 :rem 157
2020 PRINT"{CLR}{2 DOWN}{11 SPACES}CONGRATULATIONS
!!!" :rem 42
2025 PRINT"{DOWN}{4 RIGHT}YOU HAVE DONE WHAT FEW M
ORTALS " :rem 158
2027 PRINT"{DOWN}{11 RIGHT}HAVE ATTEMPTED."
:rem 230
2030 PRINT:PRINT"YOU ALSO HAVE";:IFYGTHENPRINTTR*2
00;"GOLD PIECES":GOTO2050 :rem 7
2040 PRINTTR*100;"GOLD PIECES" :rem 18
2050 PRINT"{DOWN}WOULD YOU LIKE TO TRY AGAIN?? (Y/
N)" :rem 253
2080 GETD$:IFD$=""THEN2080 :rem 189
2090 IFD$="Y"THENPOKE53272,28:PRINT"{CLR}":RUN
:rem 96
2100 END :rem 154
3000 POKEPS,1:POKEPS+CL,4 :rem 24
3015 GOSUB16000:PRINT"{CLR}":POKE53281,1:POKE53272
,21 :rem 162
3020 PRINT"{CLR}{4 DOWN}THE BOOGENS GOT YOU!!!
{7 LEFT}{2 DOWN}TOO BAD... TRY AGAIN?"
:rem 243
3040 GETB$:IFB$=""THEN3040 :rem 179
3050 IFB$="N"THENEND :rem 146
3060 PRINT"{CLR}":RUN :rem 92
10000 POKEs1+24,15:POKEs1+5,0:POKEs1+6,248:POKEs1+
4,WF:POKEs1+1,HN:POKEs1,LN :rem 91
10005 FORR=1TODR:NEXTR :rem 161
10010 FORZZ=S1TOS1+23:POKEZZ,0:NEXTZZ:RETURN
:rem 197
12000 HN=16:LN=195:HJ=21:LJ=31:HK=25:LK=30:WF=17:D
R=100 :rem 182
```

# 1: Games

---

```
12005 POKES1+24,15:POKES1+5,0:POKES1+6,248:POKES1+
12,0:POKES1+13,248 :rem 172
12010 POKES1+19,0:POKES1+20,248:POKES1+4,WF:POKES1
+11,WF:POKES1+18,WF :rem 126
12015 POKES1+1,HN:POKES1,LN:POKES1+8,HJ:POKES1+7,L
J:POKES1+15,HK:POKES1+14,LK :rem 25
12020 FORR=1TODR:NEXTR :rem 160
12025 FORZZ=S1TOS1+23:POKEZZ,0:NEXTZZ:RETURN
:rem 205
13000 POKES1+24,15:POKES1+5,0:POKES1+6,248:POKES1+
4,WF:POKES1+1,HN:POKES1,LN :rem 94
13005 FORR=1TODR:NEXTR:RETURN :rem 190
14000 POKES1+1,HN:POKES1,LN :rem 121
14010 FORR=1TODR:NEXT:POKE53281,6 :rem 29
14020 RETURN :rem 215
15000 FORZZ=S1TOS1+23:POKEZZ,0:NEXT:RETURN :rem 21
16000 HN=8:LN=97:HJ=8:LJ=225:HK=9:LK=104:WF=17:DR=
1000 :rem 154
16005 POKES1+24,15:POKES1+5,0:POKES1+6,248:POKES1+
12,0:POKES1+13,248 :rem 176
16010 POKES1+19,0:POKES1+20,248:POKES1+4,WF:POKES1
+11,33:POKES1+18,129 :rem 74
16015 POKES1+1,HN:POKES1,LN:POKES1+8,HJ:POKES1+7,L
J:POKES1+15,HK:POKES1+14,LK :rem 29
16020 FORR=1TODR:NEXTR :rem 164
16025 FORZZ=S1TOS1+23:POKEZZ,0:NEXTZZ:RETURN
:rem 209
17000 HN=16:LN=195:DR=.0001:WF=17 :rem 90
17010 POKES1+25,15:POKES1+5,0:POKES1+6,248:POKES1+
4,WF :rem 166
17020 FORHN=16TO42:FORLN=195TO196:POKES1+1,HN:POKE
S1,LN:NEXTLN,HN :rem 145
17025 FORZZ=S1TOS1+23:POKEZZ,0:NEXTZZ:RETURN
:rem 210
17030 POKES1+25,15:POKES1+5,0:POKES1+6,248:POKES1+
4,WF :rem 168
17035 FORHN=41TOBSTEP-1:FORLN=195TO194STEP-1:POKES
1+1,HN:POKES1,LN:NEXTLN,HN :rem 153
17060 FORZZ=S1TOS1+23:POKEZZ,0:NEXT:RETURN :rem 29
18000 HN=7:LN=12:WF=17:DR=300:GOSUB19000 :rem 71
18005 HN=9:LN=104:HJ=0:LJ=0:WF=17:DR=300:GOSUB1900
0 :rem 246
18010 HN=0:LN=0:HJ=9:LJ=104:HK=12:LK=143:WF=17:DR=
750:GOSUB19000 :rem 14
18015 HN=15:LN=210:HJ=0:LJ=0:HK=0:LK=0:WF=17:DR=75
:GOSUB19000 :rem 115
18020 HN=15:LN=210:HJ=12:LJ=143:DR=600:GOSUB19000
:rem 64
```



```
18025 FORYY=1TO4 :rem 222
18030 HK=6:LK=71:DR=150:GOSUB19000 :rem 207
18035 HK=4:LK=180:DR=150:GOSUB19000 :rem 3
18040 NEXTYY :rem 238
18045 FORZZ=S1TOS1+23:POKEZZ,0:NEXT:RETURN :rem 33
18050 RETURN :rem 222
19000 POKES1+24,15:POKES1+5,0:POKES1+6,248:POKES1+
12,0:POKES1+13,248 :rem 174
19005 POKES1+19,0:POKES1+20,248:POKES1+4,WF:POKES1
+11,WF:POKES1+18,WF :rem 137
19010 POKES1+1,HN:POKES1,LN:POKES1+8,HJ:POKES1+7,L
J:POKES1+15,HK:POKES1+14,LK :rem 27
19015 FORR=1TODR:NEXTR :rem 171
19020 RETURN :rem 220
20000 IFPEEK(1024+CL)=6THENPOKE1024+CL,7:RETURN
:rem 71
20010 POKE1024+CL,6:RETURN :rem 31
```

# Trident

Original Program by C. O. Dickerson

64 Version by Kevin Martin

*Join the crew of the U.S.S. Trident and test your skills in this exciting naval simulation for the Commodore 64. A joystick is required.*

**Y**ou are missile officer aboard the U.S.S. *Trident*, the world's newest and most powerful nuclear submarine. Suddenly, the Priority One Channel signals a red alert: The enemy has launched an all-out attack.

Since you are hundreds of feet below the surface, you rely solely on your computerized defense screen to show you what is happening on the surface—and the picture isn't good. Enemy missiles are coming in waves, increasing in number and speed with each new attack, and you must use every ounce of skill you have to meet the massive assault. Your defensive missiles can hover in ambush or rocket through the atmosphere at twice the speed of anything the enemy can launch. But even with such weapons at your disposal, you know that lightning reflexes will be required to repel the attack.

## Defensive Postures

You direct each of your defensive missiles to its target with a joystick plugged into port 2. Once you destroy one of the enemy missiles, the computer gets ready to launch another anti-missile. If you destroy all the incoming missiles in the attack wave, you move on to a higher difficulty level in which the speed of the incoming missiles is increased. If you are not successful, you can start over by pressing the fire button.

In each game, you can select a level of difficulty, which determines the speed of the incoming missiles. Each successive level challenges you with higher speeds. You have four choices, which can be selected by pressing the appropriate function key:

- f1: Beginner
- f3: Intermediate
- f5: Advanced
- f7: Expert

## Typing In Trident

"Trident" is written entirely in machine language and must be entered with "MLX," the machine language editor program found in Appendix D. Be sure you read the MLX article and understand how to use that program before you start typing the data for Trident.

MLX requires that you input the starting and ending addresses for your machine language. For Trident, use the following addresses:

Starting address: 49152

Ending address: 51659

After typing in Trident, be sure to use the MLX Save option to store a copy of your work on tape or disk.

After saving, you can load it back into the computer by typing LOAD "TRIDENT",8,1 for disk or LOAD "TRIDENT",1,1 for tape. To run the program, type SYS 49152.

## Trident

*Be sure to use "MLX" (Appendix D) when entering this program.*

```

49152 :032,041,197,169,000,141,068
49158 :032,208,169,011,141,033,088
49164 :208,169,060,141,132,003,213
49170 :169,147,032,210,255,173,236
49176 :030,208,169,000,141,120,180
49182 :003,169,144,032,210,255,075
49188 :160,000,185,071,201,201,086
49194 :008,240,007,153,000,050,244
49200 :200,076,038,192,169,138,093
49206 :133,252,169,197,133,253,167
49212 :160,000,177,252,201,000,082
49218 :240,012,032,210,255,230,021
49224 :252,208,243,230,253,076,054
49230 :062,192,169,063,141,021,214
49236 :208,169,200,141,248,007,033
49242 :141,250,007,141,251,007,119
49248 :141,252,007,141,253,007,129
49254 :169,201,141,249,007,169,014
49260 :000,162,000,157,000,208,123
49266 :232,224,015,208,248,169,186
49272 :015,141,039,208,169,146,070
49278 :141,000,208,169,141,141,158
49284 :001,208,169,140,141,002,025
49290 :208,169,135,141,003,208,234
49296 :169,002,141,040,208,169,105
49302 :004,141,041,208,169,014,215

```

# 1: Games

---

49308 :141,042,208,169,007,141,096  
49314 :043,208,169,013,141,044,012  
49320 :208,169,000,141,016,208,142  
49326 :169,255,141,062,003,032,068  
49332 :011,194,169,255,141,015,197  
49338 :212,169,128,141,018,212,042  
49344 :173,030,208,169,049,032,085  
49350 :136,196,169,254,141,066,136  
49356 :003,169,255,141,067,003,074  
49362 :032,154,196,169,017,141,151  
49368 :005,212,169,243,141,006,224  
49374 :212,169,033,141,004,212,225  
49380 :032,249,192,032,139,193,041  
49386 :032,249,192,032,193,194,102  
49392 :032,064,196,032,015,197,008  
49398 :076,228,192,141,060,003,178  
49404 :142,061,003,173,000,220,083  
49410 :041,008,208,039,174,000,216  
49416 :208,232,224,000,208,014,126  
49422 :173,016,208,009,001,141,050  
49428 :016,208,141,137,197,076,027  
49434 :042,193,173,016,208,041,187  
49440 :001,201,001,208,005,224,160  
49446 :009,208,001,202,142,000,088  
49452 :208,173,000,220,041,004,178  
49458 :208,039,174,000,208,202,113  
49464 :224,000,208,014,173,016,179  
49470 :208,041,254,141,016,208,162  
49476 :141,137,197,076,088,193,132  
49482 :173,016,208,041,001,201,202  
49488 :000,208,005,224,026,208,239  
49494 :001,232,142,000,208,173,074  
49500 :000,220,041,001,208,012,062  
49506 :174,001,208,202,224,054,193  
49512 :208,001,232,142,001,208,128  
49518 :173,000,220,041,002,208,242  
49524 :012,174,001,208,232,224,199  
49530 :228,208,001,202,142,001,136  
49536 :208,173,060,003,174,061,039  
49542 :003,032,060,195,096,173,181  
49548 :062,003,041,003,168,173,078  
49554 :129,197,201,000,208,033,146  
49560 :192,000,240,111,173,130,230  
49566 :197,201,000,208,022,192,210  
49572 :001,240,100,173,131,197,238  
49578 :201,000,208,011,192,002,016  
49584 :240,089,173,132,197,201,184  
49590 :000,240,082,185,129,197,247  
49596 :201,000,240,069,170,192,036  
49602 :001,208,032,173,137,197,174

49608 :041,008,201,000,240,023,201  
49614 :138,056,233,001,201,000,067  
49620 :208,027,173,137,197,041,227  
49626 :195,141,137,197,138,056,058  
49632 :233,002,076,241,193,138,083  
49638 :201,146,144,005,233,001,192  
49644 :076,241,193,105,001,153,237  
49650 :129,197,185,133,197,201,004  
49656 :141,144,005,233,001,076,080  
49662 :002,194,105,001,153,133,074  
49668 :197,136,192,255,208,175,143  
49674 :096,238,062,003,173,062,132  
49680 :003,206,132,003,041,003,148  
49686 :141,063,003,170,160,000,047  
49692 :169,000,141,064,003,238,131  
49698 :063,003,206,064,003,153,014  
49704 :129,197,200,192,004,208,202  
49710 :248,169,028,141,129,197,190  
49716 :169,192,141,137,197,173,037  
49722 :027,212,041,127,105,044,102  
49728 :141,133,197,224,000,240,231  
49734 :069,169,008,141,130,197,016  
49740 :169,008,013,137,197,141,229  
49746 :137,197,173,027,212,041,101  
49752 :127,105,044,141,134,197,068  
49758 :169,203,045,137,197,141,218  
49764 :137,197,224,001,240,034,165  
49770 :169,055,141,135,197,173,208  
49776 :027,212,041,127,105,067,179  
49782 :141,131,197,224,001,240,028  
49788 :015,169,227,141,136,197,241  
49794 :173,027,212,041,127,105,047  
49800 :067,141,132,197,224,000,129  
49806 :208,005,169,007,141,021,181  
49812 :208,224,001,208,005,169,195  
49818 :015,141,021,208,224,002,253  
49824 :208,005,169,031,141,021,223  
49830 :208,224,003,208,005,169,215  
49836 :063,141,021,208,238,063,138  
49842 :003,032,013,196,173,132,215  
49848 :003,201,014,240,003,238,115  
49854 :132,003,096,173,129,197,152  
49860 :201,000,240,012,141,004,026  
49866 :208,173,133,197,141,005,035  
49872 :208,076,222,194,169,000,053  
49878 :141,004,208,169,000,141,109  
49884 :005,208,173,130,197,201,110  
49890 :000,240,012,141,006,208,065  
49896 :173,134,197,141,007,208,068  
49902 :076,251,194,169,000,141,045

# 1: Games

---

49908 :006,208,169,030,141,007,037  
49914 :208,173,131,197,201,000,136  
49920 :240,012,141,008,208,173,014  
49926 :135,197,141,009,208,076,004  
49932 :024,195,169,000,141,008,037  
49938 :208,169,070,141,009,208,055  
49944 :173,132,197,201,000,240,199  
49950 :012,141,010,208,173,136,198  
49956 :197,141,011,208,076,053,210  
49962 :195,169,000,141,010,208,253  
49968 :169,111,141,011,208,173,093  
49974 :137,197,141,016,208,096,081  
49980 :173,030,208,141,065,003,168  
49986 :173,065,003,041,004,201,041  
49992 :004,208,022,169,000,141,104  
49998 :129,197,032,013,196,032,165  
50004 :217,196,173,021,208,041,172  
50010 :251,141,021,208,076,154,173  
50016 :196,173,065,003,041,008,070  
50022 :201,008,208,030,169,000,206  
50028 :141,130,197,173,137,197,059  
50034 :041,247,141,137,197,032,141  
50040 :013,196,032,238,196,173,200  
50046 :021,208,041,247,141,021,037  
50052 :208,076,154,196,173,065,236  
50058 :003,041,016,201,016,208,111  
50064 :022,169,000,141,131,197,036  
50070 :032,013,196,032,249,196,100  
50076 :173,021,208,041,239,141,211  
50082 :021,208,076,154,196,173,222  
50088 :065,003,041,032,201,032,030  
50094 :208,022,169,000,141,132,078  
50100 :197,032,013,196,032,004,142  
50106 :197,173,021,208,041,223,025  
50112 :141,021,208,076,154,196,220  
50118 :096,169,015,141,024,212,087  
50124 :169,010,141,132,003,162,053  
50130 :255,142,001,212,202,142,140  
50136 :068,003,032,015,197,174,193  
50142 :068,003,224,000,208,239,196  
50148 :169,050,141,132,003,032,243  
50154 :015,197,238,032,208,173,073  
50160 :032,208,041,015,201,000,225  
50166 :208,241,169,000,141,024,005  
50172 :212,173,000,220,041,016,146  
50178 :208,249,104,104,104,104,107  
50184 :104,104,076,003,192,169,144  
50190 :146,141,000,208,169,141,051  
50196 :141,001,208,162,007,160,187  
50202 :035,024,032,240,255,206,050

50208 :063,003,238,064,003,173,064  
50214 :063,003,024,105,048,032,057  
50220 :210,255,162,012,160,035,110  
50226 :024,032,240,255,173,064,070  
50232 :003,024,105,048,032,210,222  
50238 :255,096,165,197,201,004,212  
50244 :208,011,169,060,141,132,021  
50250 :003,169,049,032,136,196,147  
50256 :096,201,005,208,011,169,002  
50262 :042,141,132,003,169,050,111  
50268 :032,136,196,096,201,006,247  
50274 :208,011,169,035,141,132,026  
50280 :003,169,051,032,136,196,179  
50286 :096,201,003,208,011,169,030  
50292 :027,141,132,003,169,052,128  
50298 :032,136,196,096,173,141,128  
50304 :002,041,001,201,000,208,069  
50310 :247,096,141,082,003,162,097  
50316 :017,160,035,024,032,240,136  
50322 :255,173,082,003,032,210,133  
50328 :255,096,173,066,003,174,151  
50334 :067,003,024,105,002,144,247  
50340 :001,232,141,066,003,142,237  
50346 :067,003,173,062,003,074,040  
50352 :074,141,072,003,173,066,193  
50358 :003,174,067,003,024,109,050  
50364 :072,003,144,001,232,141,013  
50370 :066,003,142,067,003,162,125  
50376 :004,160,034,024,032,240,182  
50382 :255,173,067,003,174,066,176  
50388 :003,032,205,189,096,173,142  
50394 :004,208,201,138,144,003,148  
50400 :076,199,195,096,173,004,199  
50406 :208,201,138,144,248,076,221  
50412 :199,195,173,006,208,201,194  
50418 :156,176,003,076,199,195,023  
50424 :096,173,009,208,201,133,044  
50430 :144,003,076,199,195,096,199  
50436 :173,011,208,201,151,176,156  
50442 :003,076,199,195,096,142,209  
50448 :060,003,140,061,003,162,189  
50454 :000,160,000,232,208,253,107  
50460 :200,204,132,003,208,247,254  
50466 :174,060,003,172,061,003,251  
50472 :096,169,147,032,210,255,181  
50478 :169,000,141,032,208,169,253  
50484 :011,141,033,208,169,154,000  
50490 :032,210,255,162,012,160,121  
50496 :016,024,032,240,255,162,025  
50502 :000,189,036,201,201,000,185

# 1: Games

---

50508 :240,007,232,032,210,255,028  
50514 :076,071,197,162,021,160,001  
50520 :007,024,032,240,255,162,040  
50526 :000,189,044,201,201,000,217  
50532 :240,007,232,032,210,255,052  
50538 :076,095,197,169,000,162,037  
50544 :000,157,000,208,232,224,165  
50550 :017,208,248,173,000,220,216  
50556 :041,016,208,237,096,000,210  
50562 :000,000,000,000,000,000,130  
50568 :000,252,019,176,195,178,188  
50574 :195,178,195,178,195,178,237  
50580 :195,178,195,178,195,178,243  
50586 :195,178,195,178,195,178,249  
50592 :195,178,195,178,195,178,255  
50598 :195,178,195,174,032,032,204  
50604 :032,032,032,032,032,032,108  
50610 :032,221,032,221,032,221,169  
50616 :032,221,032,221,032,221,175  
50622 :032,221,032,221,032,221,181  
50628 :032,221,032,221,032,221,187  
50634 :032,221,032,221,032,221,193  
50640 :032,221,032,032,032,032,077  
50646 :032,032,032,032,032,171,033  
50652 :195,219,195,219,195,219,182  
50658 :195,219,195,219,195,219,188  
50664 :195,219,195,219,195,219,194  
50670 :195,219,195,219,195,219,200  
50676 :195,219,195,219,195,179,166  
50682 :032,032,032,032,032,032,186  
50688 :032,032,032,221,032,221,058  
50694 :032,221,032,221,032,221,253  
50700 :032,221,032,221,032,221,003  
50706 :032,221,032,221,032,221,009  
50712 :032,221,032,221,032,221,015  
50718 :032,221,032,221,032,032,088  
50724 :083,067,079,082,069,032,192  
50730 :032,171,195,219,195,219,049  
50736 :195,219,195,219,195,219,010  
50742 :195,219,195,219,195,219,016  
50748 :195,219,195,219,195,219,022  
50754 :195,219,195,219,195,219,028  
50760 :195,179,032,032,032,032,062  
50766 :032,032,032,032,032,221,203  
50772 :032,221,032,221,032,221,075  
50778 :032,221,032,221,032,221,081  
50784 :032,221,032,221,032,221,087  
50790 :032,221,032,221,032,221,093  
50796 :032,221,032,221,032,221,099  
50802 :032,032,032,032,032,032,050



50808 :032,032,032,171,195,219,033  
50814 :195,219,195,219,195,219,088  
50820 :195,219,195,219,195,219,094  
50826 :195,219,195,219,195,219,100  
50832 :195,219,195,219,195,219,106  
50838 :195,219,195,179,077,073,064  
50844 :083,083,073,076,069,083,111  
50850 :032,221,032,221,032,221,153  
50856 :032,221,032,221,032,221,159  
50862 :032,221,032,221,032,221,165  
50868 :032,221,032,221,032,221,171  
50874 :032,221,032,221,032,221,177  
50880 :032,221,032,032,032,032,061  
50886 :032,032,032,032,032,171,017  
50892 :195,219,195,219,195,219,166  
50898 :195,219,195,219,195,219,172  
50904 :195,219,195,219,195,219,178  
50910 :195,219,195,219,195,219,184  
50916 :195,219,195,219,195,179,150  
50922 :032,032,076,069,070,084,085  
50928 :032,032,032,221,032,221,042  
50934 :032,221,032,221,032,221,237  
50940 :032,221,032,221,032,221,243  
50946 :032,221,032,221,032,221,249  
50952 :032,221,032,221,032,221,255  
50958 :032,221,032,221,032,032,072  
50964 :032,032,032,032,032,032,212  
50970 :032,171,195,219,195,219,033  
50976 :195,219,195,219,195,219,250  
50982 :195,219,195,219,195,219,000  
50988 :195,219,195,219,195,219,006  
50994 :195,219,195,219,195,219,012  
51000 :195,179,032,032,032,032,046  
51006 :032,032,032,032,032,221,187  
51012 :032,221,032,221,032,221,059  
51018 :032,221,032,221,032,221,065  
51024 :032,221,032,221,032,221,071  
51030 :032,221,032,221,032,221,077  
51036 :032,221,032,221,032,221,083  
51042 :077,073,083,083,073,076,051  
51048 :069,083,032,171,195,219,105  
51054 :195,219,195,219,195,219,072  
51060 :195,219,195,219,195,219,078  
51066 :195,219,195,219,195,219,084  
51072 :195,219,195,219,195,219,090  
51078 :195,219,195,179,032,032,218  
51084 :032,032,032,032,032,032,076  
51090 :032,221,032,221,032,221,137  
51096 :032,221,032,221,032,221,143  
51102 :032,221,032,221,032,221,149

# 1: Games

---

51108 :032,221,032,221,032,221,155  
51114 :032,221,032,221,032,221,161  
51120 :032,221,068,069,083,084,221  
51126 :082,079,089,069,068,171,228  
51132 :195,219,195,219,195,219,150  
51138 :195,219,195,219,195,219,156  
51144 :195,219,195,219,195,219,162  
51150 :195,219,195,219,195,219,168  
51156 :195,219,195,219,195,179,134  
51162 :032,032,032,032,032,032,154  
51168 :032,032,032,221,032,221,026  
51174 :032,221,032,221,032,221,221  
51180 :032,221,032,221,032,221,227  
51186 :032,221,032,221,032,221,233  
51192 :032,221,032,221,032,221,239  
51198 :032,221,032,221,032,032,056  
51204 :032,032,032,032,032,032,196  
51210 :032,171,195,219,195,219,017  
51216 :195,219,195,219,195,219,234  
51222 :195,219,195,219,195,219,240  
51228 :195,219,195,219,195,219,246  
51234 :195,219,195,219,195,219,252  
51240 :195,179,032,032,076,069,111  
51246 :086,069,076,032,032,221,050  
51252 :032,221,032,221,032,221,043  
51258 :032,221,032,221,032,221,049  
51264 :032,221,032,221,032,221,055  
51270 :032,221,032,221,032,221,061  
51276 :032,221,032,221,032,221,067  
51282 :032,032,032,032,032,032,018  
51288 :032,032,032,171,195,219,001  
51294 :195,219,195,219,195,219,056  
51300 :195,219,195,219,195,219,062  
51306 :195,219,195,219,195,219,068  
51312 :195,219,195,219,195,219,074  
51318 :195,219,195,179,032,032,202  
51324 :032,032,032,032,032,032,060  
51330 :032,221,032,221,032,221,121  
51336 :032,221,032,221,032,221,127  
51342 :032,221,032,221,032,221,133  
51348 :032,221,032,221,032,221,139  
51354 :032,221,032,221,032,221,145  
51360 :032,221,032,032,032,032,029  
51366 :032,032,032,032,032,171,241  
51372 :195,219,195,219,195,219,134  
51378 :195,219,195,219,195,219,140  
51384 :195,219,195,219,195,219,146  
51390 :195,219,195,219,195,219,152  
51396 :195,219,195,219,195,179,118  
51402 :032,032,032,032,032,032,138

51408 :032,032,032,221,032,221,010  
51414 :032,221,032,221,032,221,205  
51420 :032,221,032,221,032,221,211  
51426 :032,221,032,221,032,221,217  
51432 :032,221,032,221,032,221,223  
51438 :032,221,032,221,032,032,040  
51444 :032,032,032,032,032,032,180  
51450 :032,173,195,177,195,177,175  
51456 :195,177,195,177,195,177,092  
51462 :195,177,195,177,195,177,098  
51468 :195,177,195,177,195,177,104  
51474 :195,177,195,177,195,177,110  
51480 :195,189,032,032,032,032,024  
51486 :032,032,032,032,032,000,190  
51492 :084,082,073,068,069,078,234  
51498 :084,000,080,082,069,083,184  
51504 :083,032,070,073,082,069,201  
51510 :032,066,085,084,084,079,228  
51516 :078,032,084,079,032,083,192  
51522 :084,065,082,084,000,224,093  
51528 :000,000,224,000,000,224,008  
51534 :000,000,000,000,000,000,078  
51540 :000,000,000,000,000,000,084  
51546 :000,000,000,000,000,000,090  
51552 :000,000,000,000,000,000,096  
51558 :000,000,000,000,000,000,102  
51564 :000,000,000,000,000,000,108  
51570 :000,000,000,000,000,000,114  
51576 :000,000,000,000,000,000,120  
51582 :000,000,000,000,000,000,126  
51588 :000,000,000,127,254,000,001  
51594 :127,254,000,127,254,000,132  
51600 :127,254,000,127,254,000,138  
51606 :127,254,000,127,254,000,144  
51612 :127,254,000,127,254,000,150  
51618 :127,254,000,127,254,000,156  
51624 :127,254,000,127,254,000,162  
51630 :127,254,000,000,000,000,043  
51636 :000,000,000,000,000,000,180  
51642 :000,000,000,000,000,000,186  
51648 :000,000,000,000,000,000,192  
51654 :008,013,013,013,013,013,015

# Canyon Runner

Original Program by Vic Neale  
64 Version by Kevin Mykytyn

*Pilot your craft between jagged canyon walls in this exciting arcade-style test of reflexes and nerve. For the unexpanded VIC or the 64.*

**T**he nameplate stuck on the leather flying jacket reads "Aileron Jones," but your friends all call you Al. When you took the job as a pilot with Wilderness Air Freight, everybody said you were crazy. Why, they asked, would you risk life and limb over the South American jungle for a steady \$117.63 a week? Maybe you do it for the view, which is particularly choice on a clear day like today. There's not a hint of haze, and you can see that unmapped canyon a few miles to the north. Some day you'll have to explore it firsthand.

But not today. It's payday, and your check is waiting back at the field. You scan your instruments; everything looks fine. Ought to be landing in a couple of hours. You look at the fuel gauge. Good shape there too.

Then you glance out the window, and—what's that? A dark speck in the distance? You put the aircraft on autopilot and break out binoculars. Carefully, you focus.

Your expression turns grim. Air pirates again—and this time you're too far out to call for help. You can't outrun them; they're bigger and faster. In fact, you've only got one advantage: maneuverability.

There's only one escape route: into the canyon.

## Twists and Reflections

In the VIC version of "Canyon Runner," you must pilot your craft through an ever-narrowing canyon cut through jagged rock. Scattered dark chunks of loose stone stand in your way—and your accelerator control is jammed!

You'll need all your piloting skill just to handle your wildly careening aircraft, but you've also got to contend with blinding flashes of multicolored sunlight reflected from the river below. It's enough to distract even the most skillful pilots, but don't let the flashes of sunlight distract you. If your mind wanders for even an instant . . . .

The VIC version of Canyon Runner is in two parts. Type in and save Program 1; then type in and save Program 2 using the filename CR. Program 1 will load and run Program 2 automatically.

As printed, the program is ready for disk users to type in and run. If you are using tape, be sure to change the device number from 8 to 1 in line 50 of Program 1. Then save Program 2 immediately following Program 1 on the tape. When you load and run Program 1, leave the PLAY button depressed to load and run Program 2.

Scoring is straightforward. You get 10 points for every second that you stay in the air. There are five distinct zones in the canyon, and you get bonus points every time you make it through one of them. You get 1000 points for the first zone, 2000 points for the second, and so on. A 5000-point bonus is awarded every time you make it through one of the patches of lights. If you make it through all five zones, you'll return to Zone 1, but face an even tougher course.

## **Copter Vs. Copter**

On the Commodore 64, Canyon Runner is a game for two players. One player takes the role of the cargo pilot, while the other pilots the pursuing pirate craft. Each must guide a high-performance copter through a tortuous and ever-changing canyon, aim and fire missiles, and avoid missiles fired by the other craft.

This version is written entirely in machine language and must be entered using the "MLX" program in Appendix D. The starting address is 49152, and the ending address is 51720. To run the program, load it by filename followed by ,1,1 (if you're using tape) or ,8,1 (if you're using disk). Then type SYS 49152, press RETURN, and the game will begin.

The 64 version requires two joysticks and offers many options. For example, you can select the type of missile you want to use by pressing A (for Altitude) or D (for Detonation). Altitude missiles always explode at the altitude at which they are fired, while detonation missiles will change altitude as you change the altitude of your copter.

To get the feel of the controls, you can press S and select the solo flight option. It allows Player 1 to practice flying through the canyon, although it does not fully enable the

missile launcher. Note that even on the solo flight option, both joysticks must be pushed forward to initiate play.

At the bottom left and bottom right of the screen are the numbers 1-9. One number will be highlighted; it indicates the present difficulty level. The lower the number, the more difficult the course. Each player can select a difficulty level by moving the joystick left or right.

Finally, by pressing 1, 2, or 3, you can adjust the width of the canyon. Beginners should start with width 1, which is comfortably wide. Width 2 is more narrow. Width 3 should be reserved for experts, since it requires precision maneuvering.

Guide your copter using your joystick. Move it left or right to go left or right. Pull back to climb, and push forward to dive. A dual altimeter reveals the altitude of each copter.

Fire missiles by pressing the fire button on the joystick. You'll hear the flight of the missile as a whistling sound, and you'll see the cloud of smoke as the missile explodes. Special sprite priorities are used to simulate explosions above and below the target.

Each player starts with five copters and an unlimited supply of missiles. A copter is lost whenever it collides with the canyon wall or runs into an opponent's missile. The first player to run out of copters loses the game.

### Program 1. Canyon Runner, VIC Loader

*For mistake-proof program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

10 POKE51,0:POKE55,0:POKE52,28:POKE56,28:CLR:POKE3
    6869,255                                     :rem 173
15 PRINT"{CLR}{9 DOWN}{4 RIGHT}CANYON RUNNER"
                                                :rem 121
16 PRINT"{3 DOWN}{5 RIGHT}PLEASE WAIT{WHT}"
                                                :rem 240
20 FORI=7168TO7679:POKEI,PEEK(I+25600):NEXT:rem 99
30 FORI=7384TO7399:READA:POKEI,A:NEXT         :rem 84
40 DATA 255,255,255,255,255,255,255,255,195,231,23
    1,231,0,0,165,231                           :rem 229
50 S$="LO"+CHR$(34)+"CR"+CHR$(34)+",8:"+CHR$(131):
    REM CHANGE 8 TO 1 FOR TAPE USERS           :rem 214
60 FORI=1TOLEN(S$):POKE630+I,ASC(MID$(S$,I)):NEXT:
    POKE198,I:END                               :rem 93

```

## Program 2. Canyon Runner, VIC Main Program

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

5 SYS 65017:POKE36869,255 :rem 120
10 PRINT"{CLR}":POKE36879,8:POKE36878,15:S5=36875:
   S1=36877:C=30720:S2=36876 :rem 132
11 DEFFNR(X)=INT(RND(1)*X)+1:DEFFNP(X)=X+(PEEK(1)-
   PEEK(2)):DIMB$(15),T$(5),P(7),L(7) :rem 0
20 FORX=0TO7:READL(X):NEXT :rem 203
40 FORX=1TO6:READC$(X):NEXT :rem 202
100 GOSUB815:FORX=828TO871:READA:POKEX,A:NEXT
   :rem 136
110 FORX=1TO15:READB$(X):NEXT:FORX=1TO5:READT$(X):
   NEXT :rem 214
115 PRINT"{CLR}{DOWN}ENTER LEVEL":PRINT"{2 DOWN}5
   {SPACE}EASY..HARD 1" :rem 66
116 GETSK$:SK=VAL(SK$):IFSK>5ORSK<1THEN116:rem 146
120 B=0:P=7910:SC=0:S=7:X2=1:DL=SK*(15+(SK-5)):R=(
   SK*4)+20:H=SK*5 :rem 171
130 PRINT"{CLR}"SPC(S)"{WHT}"{RVS} READY{2 SPACES}"
   :FORX=1TO21:PRINTSPC(S)B$(1):NEXT:POKEP,28:FOR
   D=1TO1500 :rem 124
135 NEXT:TI$="000000" :rem 116
140 POKES5,128+(100-DL):POKES1,128+(100-DL):FORX=1
   TO5 :rem 26
220 FORY1=1TO40:FORY=1TO5:J=X:J1=X:IFFNR(H)=1THENJ
   1=((X*2)+3+FNR(2)) :rem 130
230 GOSUB780 :rem 180
270 IFPEEK(P)=32THEN630 :rem 88
280 TU=0:GOSUB810:NEXT :rem 124
290 IFFNR(R)=1THENK=X:K1=K:GOTO400 :rem 117
305 TU=230:J=X:J1=X:FORY=1TO25:GOSUB800 :rem 186
310 IFPEEK(P)=32THEN630 :rem 83
315 GOSUB810 :rem 178
320 IFY=13ANDX=5THENJ1=1:TU=0:GOSUB760:S=S-4:IFS<2
   THENS=2 :rem 115
325 IFY=13ANDX=5THENNEXT :rem 46
330 IFY=13THENJ1=J1+1:TU=0:GOSUB760:NEXT :rem 253
370 NEXT:R=R-5:H=H-4:DL=DL-10:IFDL<0THENDL=0:R=10:
   H=3:GOTO140 :rem 248
380 GOTO140 :rem 105
400 FORZ=1TO13:J=X:J1=X:GOSUB800 :rem 255
410 IFPEEK(P)=32THEN630 :rem 84
420 TU=240:GOSUB810:NEXT:POKES5,228:POKES1,228
   :rem 76
430 FORZ1=1TO10:FORZ=KTOLSTEP-1:J=Z:J1=Z:GOSUB780
   :rem 172
470 IFPEEK(P)=32THEN630 :rem 90
480 POKEP,28:FORD=1TODL/2:NEXT:NEXT:K=4 :rem 0
490 FORZ=2TO5:J=Z:J1=Z:GOSUB780 :rem 229

```

# 1: Games

```

530 IFPEEK(P)=32THEN630 :rem 87
540 POKEP,28:FORD=1TODL/2:NEXT:NEXT:NEXT :rem 128
550 FORZ=4TOK1:IFS>INT((Z+11)/2)THENS=S-2 :rem 11
560 IFS<INT((Z+11)/2)THENS=S+1 :rem 252
570 J=Z:J1=Z:GOSUB800 :rem 27
580 IFPEEK(P)=32THEN630 :rem 92
590 TU=0:GOSUB810:NEXT :rem 128
600 FORZ=1TO13:J=X:J1=X:GOSUB800 :rem 1
610 IFPEEK(P)=32THEN630 :rem 86
620 TU=240:GOSUB810:NEXT:SC=SC+5000 :rem 115
625 POKES5,128+(100-DL):POKES1,128+(100-DL):rem 99
627 POKEP+C,T%(X):POKEP,27:PRINTSPC(S)"{WHT}{RVS}B
ONUS":POKEP,28:GOTO290 :rem 184
630 E$=TI$:POKES5,0:POKES1,0:POKES2,0:POKEP+C,2
:rem 101
640 FORX=180TO220STEP2:POKES2,X:FORD=1TO50:NEXT:NE
XT :rem 147
650 POKES2,0:FORX=0TO7:P(X)=P+L(X):POKEP,42:NEXT:P
OKES1,175 :rem 197
660 FORX=15TO7STEP-1:FORY=0TO7:POKEP(Y),46:P(Y)=P(
Y)+L(Y):POKEP(Y)+C,1:POKEP(Y),90:NEXT :rem 215
670 POKE36878,X:NEXT :rem 217
700 POKES1,0:POKE36878,15 :rem 177
710 S9=((VAL(MID$(E$,5,2)))+(VAL(MID$(E$,3,2)))*6
0):SC=SC+(S9*10) :rem 234
720 PRINT"{CLR}{6 RIGHT}{CYN}GAME OVER":PRINT"
{2 DOWN}"S9"SEC. IN TUNNEL" :rem 19
722 PRINT"{2 DOWN}SCORE:"SC :rem 218
725 PRINT"{3 DOWN} FIRE BUTTON TO PLAY" :rem 125
726 PRINT"{DOWN}C TO CHANGE SKILL{5 SPACES}{DOWN}S
TO STOP" :rem 217
730 IF-((PEEK(37151)AND32)=0)=1THEN120 :rem 68
735 GETA$:IFA$="C"THEN115 :rem 156
740 IFA$<>"S"THEN730 :rem 105
750 END :rem 115
760 POKEP+C,T%(J):POKEP,27:PRINTSPC(S)"{WHT}{RVS}B
ONUS":POKEP,28 :rem 154
770 B=B+1000:SC=SC+B:RETURN :rem 117
780 T=FNR(4):IFT<=2THENS=S+1:IFS>J+10THENS=S-2
:rem 114
790 IFT>=3THENS=S-1:IFS<2THENS=S+2 :rem 154
800 POKEP+C,T%(J):SYS828:POKEP,27:P=FNP(P):PRINTSP
C(S)B$(J1):RETURN :rem 215
810 POKEP,28:POKES2,TU:FORD=1TODL:NEXT:POKES2,0:RE
TURN :rem 84
815 FORD=1TO1500:NEXT:RETURN :rem 51
820 DATA-22,-21,1,23,22,21,-1,-23,28,159 :rem 183
830 DATA156,30,31,158,169,128,141,19,145,169,0,133
,1,133,2,169,127,141,34,145,162,119 :rem 141

```



```

840 DATA236,32,145,208,4,169,1,133,1,169,255,141,3
4,145,162,110,236,17,145,208,4,169 :rem 92
850 DATA1,133,2,96,"{BLU}[[[[[[[[[",{GRN}[[[[[[[[["
"{YEL}[[[[[[[[[",{PUR}[[[[[[[[[",{RED}[[[[[[[",{BLU}[
[[{2 SPACES}[[[[[" :rem 194
860 DATA"{BLU}[ [[[[ [[[",{GRN}[[[[{OFF}{2 SPACES}[[
","{GRN}[[{OFF}{2 SPACES}[[[[[",{YEL}[[[[{OFF}
{2 SPACES}[[[",{YEL}[[{OFF}{2 SPACES}[[[" :rem 35
870 DATA"{PUR}[[[[{OFF} [","{PUR}[[{OFF} [[[[[",{RED}
[[[[[",{RED}[[{OFF} [[["6,5,7,4,2 :rem 172

```

### Program 3. Canyon Runner, 64 Version

*Be sure to use "MLX" (Appendix D) when entering this program.*

```

49152 :076,181,195,169,019,141,013
49158 :017,208,169,127,141,013,169
49164 :220,169,032,141,020,003,085
49170 :169,192,141,021,003,169,201
49176 :129,141,013,220,141,026,182
49182 :208,096,169,001,141,025,158
49188 :208,173,018,208,201,255,075
49194 :208,042,169,212,141,018,064
49200 :208,173,242,002,208,008,121
49206 :169,007,141,242,002,032,135
49212 :160,193,173,017,208,041,084
49218 :120,013,242,002,141,017,089
49224 :208,206,242,002,173,013,148
49230 :220,041,001,208,021,076,133
49236 :188,254,169,255,141,018,085
49242 :208,173,017,208,041,120,089
49248 :009,007,141,017,208,076,042
49254 :076,192,206,167,002,240,217
49260 :003,076,050,193,169,006,093
49266 :141,167,002,169,128,141,094
49272 :018,212,173,249,007,201,212
49278 :243,208,021,169,240,141,124
49284 :249,007,174,167,003,208,172
49290 :003,141,250,007,169,129,069
49296 :141,018,212,076,161,192,176
49302 :238,249,007,174,167,003,220
49308 :208,003,238,250,007,173,011
49314 :000,220,172,176,002,174,138
49320 :002,208,074,176,005,192,057
49326 :000,240,001,136,074,176,033
49332 :005,192,255,240,001,200,049
49338 :074,176,001,202,074,176,121
49344 :001,232,074,008,142,002,139
49350 :208,140,176,002,152,074,182
49356 :074,074,024,105,214,141,068

```

# 1: Games

---

49362 :011,208,040,176,005,169,051  
49368 :001,141,192,002,173,001,214  
49374 :220,172,177,002,174,004,203  
49380 :208,074,176,005,192,000,115  
49386 :240,001,136,074,176,005,098  
49392 :192,255,240,001,200,074,178  
49398 :176,014,224,000,208,009,109  
49404 :173,016,208,041,251,141,058  
49410 :016,208,202,202,074,176,112  
49416 :013,224,255,208,008,173,121  
49422 :016,208,009,004,141,016,152  
49428 :208,232,142,004,208,140,186  
49434 :177,002,152,074,074,074,067  
49440 :024,105,214,141,013,208,225  
49446 :173,001,220,041,016,208,185  
49452 :005,169,001,141,193,002,043  
49458 :076,188,254,173,250,003,226  
49464 :240,096,160,039,169,032,024  
49470 :153,208,006,136,016,250,063  
49476 :173,243,002,201,000,208,127  
49482 :021,173,244,002,201,027,230  
49488 :240,008,169,027,141,244,141  
49494 :002,076,129,193,238,243,199  
49500 :002,076,129,193,201,011,192  
49506 :208,021,173,244,002,201,179  
49512 :028,240,008,169,028,141,206  
49518 :244,002,076,129,193,206,192  
49524 :243,002,076,129,193,173,164  
49530 :027,212,016,205,076,100,246  
49536 :193,173,244,002,172,243,131  
49542 :002,153,208,006,153,224,112  
49548 :006,174,245,002,200,202,201  
49554 :208,252,153,208,006,153,102  
49560 :224,006,169,001,141,250,175  
49566 :003,096,169,040,133,251,082  
49572 :169,004,133,252,169,000,123  
49578 :133,253,169,004,133,254,092  
49584 :162,018,160,039,177,251,215  
49590 :145,253,136,016,249,024,237  
49596 :169,040,101,251,133,251,109  
49602 :165,252,105,000,133,252,077  
49608 :024,169,040,101,253,133,152  
49614 :253,165,254,105,000,133,092  
49620 :254,202,208,218,032,053,155  
49626 :193,096,173,064,003,208,187  
49632 :042,173,192,002,208,003,076  
49638 :076,215,194,173,011,208,083  
49644 :141,080,003,169,255,141,001  
49650 :001,212,141,168,002,169,167  
49656 :017,141,004,212,169,017,040

49662 :141,005,212,169,226,141,124  
49668 :006,212,169,001,141,064,085  
49674 :003,173,096,003,240,003,016  
49680 :076,177,194,173,168,002,038  
49686 :141,001,212,206,168,002,240  
49692 :201,040,240,003,076,215,035  
49698 :194,169,060,141,096,003,185  
49704 :169,129,141,004,212,169,096  
49710 :010,141,001,212,169,017,084  
49716 :141,005,212,169,235,141,187  
49722 :006,212,173,013,208,205,107  
49728 :080,003,240,074,144,036,129  
49734 :173,013,208,056,237,128,117  
49740 :003,205,080,003,144,060,059  
49746 :173,004,208,141,000,208,048  
49752 :173,016,208,041,004,240,002  
49758 :008,173,016,208,009,001,253  
49764 :141,016,208,076,177,194,144  
49770 :173,013,208,024,109,128,249  
49776 :003,205,080,003,176,024,091  
49782 :173,004,208,141,006,208,090  
49788 :173,016,208,041,004,240,038  
49794 :008,173,016,208,009,008,040  
49800 :141,016,208,076,177,194,180  
49806 :173,004,208,141,008,208,116  
49812 :169,000,141,005,208,206,109  
49818 :145,003,173,016,208,041,228  
49824 :004,240,008,173,016,208,041  
49830 :009,016,141,016,208,032,076  
49836 :181,198,076,062,196,206,067  
49842 :096,003,208,033,169,000,175  
49848 :141,064,003,141,192,002,215  
49854 :141,000,208,141,006,208,126  
49860 :141,008,208,141,096,003,025  
49866 :173,016,208,041,004,141,017  
49872 :016,208,169,128,141,004,106  
49878 :212,173,065,003,208,042,149  
49884 :173,193,002,208,003,076,107  
49890 :178,195,173,013,208,141,110  
49896 :081,003,169,255,141,008,121  
49902 :212,141,169,002,169,017,180  
49908 :141,011,212,169,017,141,167  
49914 :012,212,169,226,141,013,255  
49920 :212,169,001,141,065,003,079  
49926 :173,097,003,240,003,076,086  
49932 :151,195,173,169,002,141,075  
49938 :008,212,206,169,002,201,048  
49944 :040,240,003,076,178,195,244  
49950 :169,060,141,097,003,169,157  
49956 :129,141,011,212,169,010,196

# 1: Games

---

49962 :141,008,212,169,017,141,218  
49968 :012,212,169,235,141,013,062  
49974 :212,173,011,208,205,081,176  
49980 :003,240,060,144,029,173,197  
49986 :011,208,056,237,129,003,198  
49992 :205,081,003,144,046,173,212  
49998 :002,208,141,000,208,173,042  
50004 :016,208,041,254,141,016,248  
50010 :208,076,151,195,173,011,136  
50016 :208,024,109,129,003,205,006  
50022 :081,003,176,017,173,002,042  
50028 :208,141,006,208,173,016,092  
50034 :208,041,247,141,016,208,207  
50040 :076,151,195,173,002,208,157  
50046 :141,008,208,173,016,208,112  
50052 :041,239,141,016,208,169,178  
50058 :000,141,003,208,206,144,072  
50064 :003,032,181,198,076,062,184  
50070 :196,206,097,003,208,022,114  
50076 :169,000,141,065,003,141,163  
50082 :193,002,141,000,208,141,079  
50088 :006,208,141,008,208,169,140  
50094 :128,141,011,212,076,181,155  
50100 :197,120,165,001,041,251,187  
50106 :133,001,160,000,185,000,153  
50112 :208,153,000,048,185,000,018  
50118 :209,153,000,049,185,000,026  
50124 :210,153,000,050,185,000,034  
50130 :211,153,000,051,185,000,042  
50136 :212,153,000,052,185,000,050  
50142 :213,153,000,053,185,000,058  
50148 :214,153,000,054,185,000,066  
50154 :215,153,000,055,200,208,041  
50160 :205,165,001,009,004,133,245  
50166 :001,088,160,000,185,134,046  
50172 :200,153,000,060,185,134,216  
50178 :201,153,000,061,200,208,057  
50184 :241,160,015,185,118,200,159  
50190 :153,216,048,136,016,247,062  
50196 :169,003,141,128,003,141,093  
50202 :129,003,169,000,141,033,245  
50208 :208,169,147,032,210,255,029  
50214 :024,162,010,160,000,032,170  
50220 :240,255,169,001,141,033,115  
50226 :208,076,041,198,169,005,235  
50232 :141,144,003,141,145,003,121  
50238 :169,125,141,001,208,141,079  
50244 :007,208,141,009,208,141,014  
50250 :015,212,169,000,141,167,010  
50256 :003,141,016,208,169,240,089

50262 :141,249,007,141,250,007,113  
50268 :173,017,208,041,247,141,151  
50274 :017,208,169,015,141,024,160  
50280 :212,169,000,141,192,002,052  
50286 :141,193,002,141,064,003,142  
50292 :141,065,003,141,096,003,053  
50298 :141,097,003,141,016,208,216  
50304 :141,001,212,141,008,212,075  
50310 :141,000,208,141,006,208,070  
50316 :141,008,208,141,004,212,086  
50322 :141,011,212,173,031,208,154  
50328 :173,024,208,041,240,009,079  
50334 :012,141,024,208,169,000,200  
50340 :141,250,003,169,012,141,112  
50346 :005,220,032,003,192,169,023  
50352 :000,141,033,208,169,147,106  
50358 :032,210,255,169,001,141,222  
50364 :033,208,024,162,020,160,027  
50370 :003,032,240,255,160,200,060  
50376 :169,022,032,030,171,024,136  
50382 :162,020,160,028,032,240,080  
50388 :255,160,200,169,033,032,037  
50394 :030,171,024,162,022,160,019  
50400 :003,032,240,255,160,200,090  
50406 :169,043,032,030,171,173,080  
50412 :144,003,024,105,048,032,080  
50418 :210,255,024,162,022,160,051  
50424 :028,032,240,255,160,200,139  
50430 :169,043,032,030,171,173,104  
50436 :145,003,024,105,048,032,105  
50442 :210,255,169,003,141,243,007  
50448 :002,169,027,141,244,002,089  
50454 :169,070,141,002,208,141,241  
50460 :002,208,169,200,141,004,240  
50466 :208,141,004,208,169,125,121  
50472 :141,003,208,141,005,208,234  
50478 :169,006,141,028,208,169,255  
50484 :000,160,000,153,128,061,042  
50490 :200,208,250,169,031,141,033  
50496 :128,061,169,248,141,194,237  
50502 :061,169,224,141,029,208,134  
50508 :169,244,141,255,007,169,037  
50514 :245,141,248,007,141,251,091  
50520 :007,141,252,007,169,246,142  
50526 :141,253,007,169,247,141,028  
50532 :254,007,169,150,141,014,067  
50538 :208,141,010,208,141,012,058  
50544 :208,169,204,141,015,208,033  
50550 :169,001,141,193,061,169,084  
50556 :128,141,129,061,141,023,235

# 1: Games

---

50562 :208,169,000,141,044,208,132  
50568 :141,045,208,169,007,141,079  
50574 :046,208,169,015,141,039,248  
50580 :208,141,042,208,141,043,163  
50586 :208,169,002,141,040,208,154  
50592 :169,005,141,041,208,169,125  
50598 :255,141,167,002,173,031,167  
50604 :208,206,178,003,208,003,210  
50610 :076,220,193,173,031,208,055  
50616 :041,002,240,020,173,002,150  
50622 :208,141,008,208,169,000,156  
50628 :141,003,208,206,144,003,133  
50634 :032,181,198,076,062,196,179  
50640 :173,031,208,041,004,240,137  
50646 :019,169,001,141,167,003,202  
50652 :169,245,141,250,007,206,214  
50658 :145,003,032,181,198,076,093  
50664 :062,196,173,144,003,208,250  
50670 :028,032,065,199,024,162,236  
50676 :005,160,013,032,240,255,181  
50682 :169,033,160,200,032,030,106  
50688 :171,169,053,160,200,032,017  
50694 :030,171,076,041,198,173,183  
50700 :145,003,208,157,032,065,110  
50706 :199,024,162,005,160,013,069  
50712 :032,240,255,169,022,160,134  
50718 :200,032,030,171,169,053,173  
50724 :160,200,032,030,171,032,149  
50730 :065,199,169,060,160,200,127  
50736 :032,030,171,169,049,141,128  
50742 :033,006,169,001,141,113,005  
50748 :006,169,011,141,245,002,122  
50754 :169,255,141,021,208,032,124  
50760 :252,198,173,000,220,013,160  
50766 :001,220,170,041,001,208,207  
50772 :006,032,083,199,076,054,022  
50778 :196,138,041,002,208,005,168  
50784 :169,000,133,198,000,173,001  
50790 :000,220,041,004,208,013,076  
50796 :173,128,003,201,001,240,086  
50802 :003,206,128,003,076,141,159  
50808 :198,173,000,220,041,008,248  
50814 :208,013,173,128,003,201,084  
50820 :009,240,003,238,128,003,241  
50826 :076,141,198,173,001,220,179  
50832 :041,004,208,013,173,129,200  
50838 :003,201,001,240,172,206,205  
50844 :129,003,076,071,198,173,038  
50850 :001,220,041,008,208,159,031  
50856 :173,129,003,201,009,240,155

50862 :152,238,129,003,076,071,075  
50868 :198,169,000,141,001,212,133  
50874 :141,008,212,169,129,141,218  
50880 :004,212,169,017,141,005,228  
50886 :212,169,237,141,006,212,151  
50892 :120,169,020,141,001,212,099  
50898 :169,006,141,166,003,162,089  
50904 :255,160,255,136,208,253,203  
50910 :202,208,248,169,000,141,166  
50916 :004,212,169,128,141,004,118  
50922 :212,169,000,141,008,208,204  
50928 :206,166,003,208,226,169,194  
50934 :000,141,004,212,088,096,019  
50940 :032,065,199,024,162,023,245  
50946 :160,003,032,240,255,160,084  
50952 :200,169,095,032,030,171,193  
50958 :032,095,199,024,162,023,037  
50964 :160,028,032,240,255,160,127  
50970 :200,169,095,032,030,171,211  
50976 :172,128,003,185,154,007,169  
50982 :009,128,153,154,007,172,149  
50988 :129,003,185,179,007,009,044  
50994 :128,153,179,007,160,050,215  
51000 :162,255,202,208,253,136,248  
51006 :208,248,096,120,169,000,135  
51012 :141,026,208,169,049,141,034  
51018 :020,003,169,234,141,021,150  
51024 :003,088,096,120,169,032,076  
51030 :141,020,003,169,192,141,240  
51036 :021,003,096,165,203,201,013  
51042 :056,208,010,169,011,141,181  
51048 :245,002,169,049,141,033,231  
51054 :006,201,059,208,010,169,251  
51060 :009,141,245,002,169,050,220  
51066 :141,033,006,201,008,208,207  
51072 :010,169,007,141,245,002,190  
51078 :169,051,141,033,006,201,223  
51084 :018,208,024,162,011,160,211  
51090 :208,032,240,199,162,013,232  
51096 :160,208,032,003,200,169,156  
51102 :004,141,113,006,169,255,078  
51108 :141,021,208,201,010,208,185  
51114 :024,162,080,160,003,032,119  
51120 :240,199,162,081,160,003,253  
51126 :032,003,200,169,001,141,216  
51132 :113,006,169,255,141,021,125  
51138 :208,201,013,208,010,169,235  
51144 :251,141,021,208,169,019,241  
51150 :141,113,006,024,162,013,153  
51156 :160,016,032,240,255,169,060

# 1: Games

---

51162 :106,160,200,032,030,171,149  
51168 :024,162,015,160,016,032,121  
51174 :240,255,169,112,160,200,086  
51180 :032,030,171,096,142,064,003  
51186 :194,140,065,194,142,078,031  
51192 :194,140,079,194,142,114,087  
51198 :194,140,115,194,096,142,111  
51204 :059,195,140,060,195,142,027  
51210 :073,195,140,074,195,142,061  
51216 :102,195,140,103,195,096,079  
51222 :031,080,076,065,089,069,176  
51228 :082,032,049,032,000,080,047  
51234 :076,065,089,069,082,032,191  
51240 :050,032,000,067,072,079,084  
51246 :080,080,069,082,083,032,216  
51252 :000,087,073,078,083,013,130  
51258 :013,000,032,032,032,032,199  
51264 :032,080,082,069,083,083,237  
51270 :032,085,080,032,084,079,206  
51276 :032,080,076,065,089,044,206  
51282 :032,068,079,087,078,032,202  
51288 :084,079,032,069,078,068,242  
51294 :000,028,049,050,051,052,068  
51300 :053,054,055,056,057,000,119  
51306 :076,069,086,069,076,000,226  
51312 :066,079,077,066,083,000,227  
51318 :128,064,032,064,032,024,206  
51324 :006,001,001,006,008,024,170  
51330 :032,016,096,128,000,190,080  
51336 :000,000,190,000,000,060,130  
51342 :000,000,060,000,000,060,006  
51348 :000,000,060,000,000,060,012  
51354 :000,000,060,000,000,028,242  
51360 :000,000,028,000,000,028,216  
51366 :000,000,223,000,000,223,100  
51372 :000,003,223,192,003,223,048  
51378 :192,003,223,192,003,223,246  
51384 :192,003,223,192,000,223,249  
51390 :000,000,028,000,000,016,234  
51396 :000,255,000,190,000,000,129  
51402 :190,000,000,060,000,000,196  
51408 :060,000,000,060,000,000,072  
51414 :060,000,000,060,000,000,078  
51420 :060,000,000,060,000,000,084  
51426 :060,000,000,060,000,000,090  
51432 :255,000,000,255,000,003,233  
51438 :255,192,003,255,192,085,196  
51444 :085,085,003,255,192,003,099  
51450 :255,192,000,255,000,000,184  
51456 :060,000,000,000,000,255,059



51462 :000,190,000,000,190,000,130  
51468 :000,060,000,000,060,000,132  
51474 :000,060,000,000,060,000,138  
51480 :000,060,000,000,060,000,144  
51486 :000,060,000,000,060,005,155  
51492 :000,060,020,000,255,080,195  
51498 :000,253,064,003,245,192,031  
51504 :003,215,192,003,095,192,236  
51510 :001,127,192,005,255,192,058  
51516 :020,255,000,080,060,000,219  
51522 :000,000,000,255,000,190,255  
51528 :000,000,190,000,000,060,066  
51534 :000,000,060,000,000,060,198  
51540 :000,000,060,000,000,060,204  
51546 :000,000,060,000,000,060,210  
51552 :000,080,060,000,020,060,060  
51558 :000,005,255,000,001,127,234  
51564 :000,003,095,192,003,215,104  
51570 :192,003,245,192,003,253,234  
51576 :064,003,255,080,000,255,009  
51582 :020,000,060,005,000,000,211  
51588 :000,255,255,255,255,224,096  
51594 :222,003,238,223,223,224,247  
51600 :223,223,238,192,223,128,091  
51606 :024,001,192,060,003,128,046  
51612 :024,001,192,060,003,128,052  
51618 :024,001,192,060,003,128,058  
51624 :024,001,192,060,003,128,064  
51630 :024,001,192,060,003,128,070  
51636 :024,001,192,060,003,128,076  
51642 :024,001,192,060,003,128,082  
51648 :024,001,255,255,255,239,197  
51654 :000,000,000,000,000,000,198  
51660 :001,224,000,003,240,000,160  
51666 :003,249,192,003,227,224,084  
51672 :051,255,240,127,255,248,112  
51678 :255,255,248,255,255,128,082  
51684 :255,255,240,247,255,248,192  
51690 :111,249,252,031,251,254,102  
51696 :031,251,238,000,249,254,239  
51702 :001,240,252,001,224,120,060  
51708 :000,000,000,000,000,000,252  
51714 :000,000,000,239,013,013,011

□ □ □ □ □

□ □ □ □ □

# Chapter 2

---

# Education

□ □ □ □ □

□ □ □ □ □

# Learning to Count

---

William W. Braun

*Designed for children in kindergarten through third grade, this colorful program is easily tailored to your child's needs and abilities.*

**T**eaching programs are sometimes broad in scope and appropriate for only one learning level. However, "Learning to Count" allows the parent or instructor to tailor the tutor to the child. Colorful graphics and exciting sound make it entertaining as well as educational.

## Selecting a Range

When you run the program, you're first asked to input a number from 1 to 4 to set the range of objects to be counted. Choosing the lowest range displays a random number of objects from 2 to 10, while the highest level gives groups ranging from 2 to 50 objects.

The child is asked to count the objects and type in the number. A correct answer is rewarded with a smiling face and short melody. A wrong answer elicits a *SORRY! TRY AGAIN* response. After three wrong responses, the correct answer is given.

The program continues until a zero is typed. That allows the parent or instructor to control the length of the program or to move to a higher level. When a zero is entered, the screen displays the number of tries, the number right, and the number wrong. Then, after a short graphics display, the program asks if you want to continue and at what level.

## Countable Graphics

The objects counted by the child include some of the special graphics characters, such as hearts and crosses. They're displayed in various colors and accompanied by a short tone. The DATA statements at the end of the program contain the codes for the characters, colors, and tones in groups of three.

Learning to Count can easily be modified or enhanced with custom characters. You can also include custom graphics or sound subroutines as rewards for correct answers.

### Program 1. Learning to Count, VIC Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

8 Z=7680:V=36878:S1=36876:BC=36879:COL=30720:S2=36
  874 :rem 240
9 PRINTCHR$(147):POKEV,15:POKE808,114 :rem 208
15 POKE214,7:PRINT:POKE211,4:PRINT"{PUR}LEARN TO C
  OUNT{BLU}":POKE214,17:PRINT:POKE211,6 :rem 209
25 FORI=0TO21:READA,B,C:POKEZ+I,A:POKEZ+I+COL,B:PO
  KES1,C:FORT=1TO75:NEXT:POKES1,0 :rem 224
30 IFC=236THENRESTORE :rem 96
35 NEXT :rem 167
40 FORI=0TO21:READA,B,C:POKEZ+484+I,A:POKEZ+484+I+
  COL,B:POKES1,C:FORT=1TO75 :rem 177
42 NEXT:POKE S1,0 :rem 238
45 IFC=236THENRESTORE :rem 102
50 NEXT :rem 164
55 FORI=0TO22:READA,B,C:POKEZ+22*I,A:POKEZ+22*I+CO
  L,B:POKES1,C:FORT=1TO75:NEXT:POKES1,0 :rem 0
60 IFC=236THENRESTORE :rem 99
61 NEXT :rem 166
62 FORI=0TO22:READA,B,C:POKEZ+21+22*I,A:POKEZ+21+2
  2*I+COL,B:POKES1,C :rem 102
63 FORT=1TO 75:NEXT:POKE S1,0 :rem 227
64 IFC=236THENRESTORE :rem 103
65 NEXT:FORT=1TO2500:NEXT :rem 112
70 PRINTCHR$(147):PRINT"{DOWN}{PUR}{2 SPACES}{RVS}
  LEARN TO COUNT{OFF}{BLK} CAN":PRINT"{DOWN}
  {2 SPACES}HELP YOU LEARN TO :rem 115
71 PRINT"{DOWN}{2 SPACES}COUNT UP TO 50.":PRINT"
  {3 DOWN}{2 SPACES}PRESS {RVS}1{OFF}, {RVS}2
  {OFF}, {RVS}3{OFF}, OR {RVS}4{OFF}." :rem 38
75 PRINT"{2 DOWN}{3 SPACES}UP TO 10---{RVS}1{OFF}"
  :PRINT"{DOWN}{3 SPACES}UP TO 25---{RVS}2{OFF}":
  PRINT"{DOWN}{3 SPACES}UP TO 35---{RVS}3{OFF}"
  :rem 182
76 PRINT"{DOWN}{3 SPACES}UP TO 50---{RVS}4{OFF}
  {2 RIGHT}{3 UP}{BLU}"; :rem 162
77 GETD$:IFD$=""THEN77 :rem 5
78 ONVAL(D$)GOTO81,82,83,84 :rem 15
79 GOTO77 :rem 23
81 DL=10:GOTO100 :rem 155
82 DL=25:GOTO100 :rem 162
83 DL=35:GOTO100 :rem 164
84 DL=50 :rem 158
100 POKEV,15:R=0:W=0:N=0:POKEBC,27:X=DL:PRINTCHR$(
  147) :rem 43
206 A=(INT(X*RND(1)))*2:IFA/2+1=1THEN206 :rem 227
210 N=N+1:RESTORE:SC=7834 :rem 9

```

```

220 FORH=0TOASTEP2:C=0:READL,M,K:IFK=236THENRESTOR
    E                                     :rem 61
230 POKESC+H,L:POKESC+COL+H,M:POKES1,K:FORT=1TO75:
    NEXT:POKES1,0:FORT=1TO350:NEXT      :rem 232
232 IFH=20ANDL=38THENSC=SC+22         :rem 251
233 IFH=42THENSC=SC+22                 :rem 57
234 IFH=64THENSC=SC+22                 :rem 62
235 IFH=86THENSC=SC+22                 :rem 67
236 IFH=108THENSC=SC+22                :rem 111
237 NEXT                                :rem 219
243 POKE214,19:PRINT:POKE211,0:PRINT"ENTER {RVS}0
    {OFF} OR A LETTER TOSTART OVER.    :rem 157
244 PRINT"{HOME}":FORT=0TO110:PRINT" ";:NEXT:PRINT
    "{HOME}":INPUT"HOW MANY";Y$       :rem 91
245 Y=VAL(Y$):IFY=0THENN=N-1:GOTO3000  :rem 13
260 IFVAL(Y$)=A/2+1THENGOSUB1000:R=R+1:PRINTCHR$(1
    47):GOTO206                         :rem 186
270 C=C+1:IFC=2THEN2500                :rem 41
280 PRINT"{HOME}{3 DOWN}{2 SPACES}{RVS}WRONG! TRY
    {SPACE}AGAIN.":GOSUB500:FORT=1TO900:NEXT:GOTO2
    44                                    :rem 242
500 POKES2,128:FORT=0TO300:NEXT:POKES2,0:RETURN
                                           :rem 5
1000 U=INT(RND(1)*3)+1:W=INT(RND(1)*8)+24:POKEBC,W
    :ONUGOSUB1200,2000,1200:PRINTCHR$(31) :rem 75
1001 PRINTCHR$(147):RETURN              :rem 87
1200 PRINT"{CLR}":PRINTCHR$(U+155):Q=INT(RND(1)*61
    )+161:,:FORT=0TO205:PRINTCHR$(Q);:NEXT:rem 141
1201 PRINT"{RED}RIGHT!"CHR$(U+155);:FORT=0TO249:PR
    INTCHR$(Q);:NEXT:GOSUB2006:RETURN   :rem 214
2000 PRINT"{CLR}{DOWN}{BLK}{2 SPACES}QQQQQ
    {8 SPACES}QQQQQ":PRINT" Q{5 SPACES}Q
    {6 SPACES}Q{5 SPACES}Q"           :rem 64
2001 PRINT"{BLU}{3 SPACES}{3 +}{10 SPACES}{3 +}":P
    RINT"{3 SPACES}{3 +}{10 SPACES}{3 +}":PRINT"
    {3 SPACES}{3 +}{10 SPACES}{3 +}"   :rem 117
2002 PRINT"{4 DOWN}{9 SPACES}{4 +}{18 SPACES}{4 +}
    {18 SPACES}{4 +}"                 :rem 161
2003 PRINT"{PUR} [+]{18 SPACES}{+}{2 SPACES}{+}
    {18 SPACES}{+}{3 SPACES}{+}{16 SPACES}{+}
    {5 SPACES}{+}";                   :rem 247
2004 PRINT"{14 SPACES}{+}{7 SPACES}{+}{DOWN}{+}
    {DOWN}{+}{DOWN}{8 +}{UP}{+}{UP}{+}{UP}{+}
    {BLU}"                              :rem 86
2005 REM CORRECT ANSWER TUNE           :rem 201
2006 Q=INT(RND(1)*2)+36875:FORH=235TO241:POKEQ,H:F
    ORT=1TO125:NEXT:NEXT               :rem 68
2007 FORH=241TO235STEP-1:POKEQ,H:FORT=1TO125:NEXT:
    NEXT:POKEQ,0:POKEBC,27:RETURN      :rem 148

```

## 2: Education

---

```
2500 GOSUB5000:PRINT"{3 DOWN} SORRY! WRONG AGAIN!
                                     :rem 246
2501 PRINT"{3 SPACES}THERE WERE";A/2+1"{2 UP}":GOS
UB500:FORT=0TO2000:NEXT:PRINTCHR$(147):GOTO20
6                                     :rem 178
3000 PRINT "{CLR}{GRN}"SPC(178)"YOU HAD: ";N;"TRIES
      :PRINT"Q{10 SPACES}";R;"RIGHT"      :rem 183
3002 PRINT"{DOWN}{2 RIGHT}{8 SPACES}"      :rem 225
3005 FORT=1TO4000:NEXT:PRINTCHR$(31):GOTO25
                                     :rem 168
5000 PRINT"{CLR}{2 SPACES}QQQQQ{8 SPACES}QQQQQ":PR
INT" Q{5 SPACES}Q{6 SPACES}Q{5 SPACES}Q"
                                     :rem 162
5001 PRINT"{BLK}{3 SPACES}[3 +]{10 SPACES}[3 +]:P
RINT"{3 SPACES}[3 +]{10 SPACES}[3 +]:PRINT"
{3 SPACES}[3 +]{10 SPACES}[3 +]"      :rem 233
5002 PRINT"{2 DOWN}{9 SPACES}[4 +]{18 SPACES}[4 +]
{18 SPACES}[4 +]:PRINT:PRINT      :rem 16
5003 PRINT"{9 SPACES}[4 +]:PRINT"{8 SPACES}[+}
{4 SPACES}[+}:PRINT"{7 SPACES}[+}{6 SPACES}
[+}      :rem 189
5004 PRINT"{6 SPACES}[+}{8 SPACES}[+}:PRINT"
{5 SPACES}[+}{10 SPACES}[+}:PRINT"{4 SPACES}
[+}{12 SPACES}[+}{BLU}":RETURN      :rem 205
8999 REM      :rem 199
9000 DATA81,0,219,65,2,221,83,3,223,90,4,225,88,5,
227,90,6,228,102,7,229,42,0,231,35,2 :rem 169
9001 DATA232,36,3,233,38,4,235,0,5,236      :rem 116
```

### Program 2. Learning to Count, 64 Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
28 POKE788,52:POKE53281,0:POKE53280,0:S=54272
                                     :rem 48
30 PRINT"{CLR}{12 DOWN}"TAB(11)"{WHT}LEARNING TO C
OUNT"      :rem 226
35 FORL=STOS+24:POKEL,O:NEXT:POKES+5,14:POKES+9,24
O:POKES+24,15:HF=S+1:LF=S      :rem 217
40 Z=1024:C=0:COL=S:DL(1)=10:DL(2)=25:DL(3)=35:DL(
4)=50      :rem 19
50 FORI=0TO39:READA:READB:POKEZ+I,A:POKEZ+I+COL,B:
GOSUB5000:FORT=1TO75:NEXT      :rem 124
52 IFB=5 THENRESTORE      :rem 253
53 NEXT      :rem 167
54 FORI=0TO39:READA:READB:POKEZ+960+I,A:POKEZ+960+
I+COL,B:GOSUB5000      :rem 169
55 FORT=1TO75:NEXT:IFB=5 THENRESTORE      :rem 107
56 NEXT      :rem 170
57 FORI=0TO24:READA:READB:POKEZ+40*I,A:POKEZ+40*I+
COL,B:GOSUB5000      :rem 46
```



```

58 FORT=1TO75:NEXT:IFB=5THENRESTORE           :rem 110
59 NEXT                                         :rem 173
60 FORI=0TO24:READA:READB:POKEZ+39+40*I,A:POKEZ+39
+40*I+COLL,B:GOSUB5000                         :rem 162
61 FORT=1TO75:NEXT:IFB=5THENRESTORE           :rem 104
62 NEXT:FORT=1TO2500:NEXT                       :rem 109
70 PRINTCHR$(147):PRINT"{2 DOWN}{4 SPACES}{RVS}LEA
RNING TO COUNT{OFF} CAN HELP YOU"             :rem 39
72 PRINT"{DOWN}{4 SPACES}LEARN TO COUNT UP TO 50."
:rem 196
73 PRINT"{3 DOWN}{4 SPACES}ENTER {RVS}1{OFF},
{RVS}2{OFF}, {RVS}3{OFF}, OR {RVS}4{OFF}."
:rem 153
75 PRINT"{4 DOWN}{5 SPACES}UP TO 10---{RVS}1{OFF}"
:PRINT"{DOWN}{5 SPACES}UP TO 25---{RVS}2{OFF}"
:rem 174
76 PRINT"{DOWN}{5 SPACES}UP TO 35---{RVS}3{OFF}":P
RINT"{DOWN}{5 SPACES}UP TO 50---{RVS}4{OFF}
{2 RIGHT}{3 UP}";                             :rem 173
80 INPUTD$:D=VAL(D$):IFD<1ORD>4THEN70          :rem 30
100 R=0:W=0:N=0:X=DL(D):PRINT"{CLR}"         :rem 213
206 A=(INT(X*RND(1)))*2:IFA/2+1=1THEN206      :rem 227
210 N=N+1:RESTORE:SCR=1304                     :rem 77
220 FORH=0TOASTEP2:C=0                         :rem 108
225 READL:M=INT(RND(0)*15)+1                   :rem 82
226 IFL=5THENRESTORE                           :rem 58
230 POKESCR+H,L:POKESCR+COL+H,M:FORT=1TO75:NEXT:GO
SUB5000:FORT=1TO350:NEXT                       :rem 94
232 IFH=39THENSCR=SCR+80                       :rem 230
238 NEXT                                         :rem 220
239 PRINT"{19 DOWN}{10 SPACES}ENTER {RVS}0{OFF} TO
START OVER.":POKE198,0                         :rem 6
240 PRINT"{HOME}{2 DOWN}{16 SPACES}";:INPUT"{HOME}
{2 DOWN}{2 SPACES}HOW MANY";Y$                :rem 245
245 IFY$=""0"THENN=N-1:GOTO3000              :rem 244
250 Y=VAL(Y$)                                   :rem 222
260 IFY=H/2THENGOSUB2000:R=R+1:PRINTCHR$(147):GOTO
206                                             :rem 14
270 C=C+1:IFC=3THENGOTO2500                   :rem 99
280 PRINT"{HOME}{3 DOWN}{RVS}SORRY! TRY AGAIN.":FO
RT=1TO1700:NEXT:GOSUB4000:GOTO240            :rem 94
2000 PRINT"{CLR}{4 DOWN}{WHT}"TAB(6)"{4 SPACES}QQQ
QQ{8 SPACES}QQQQQ "                          :rem 59
2001 PRINTTAB(6)"{3 SPACES}Q{5 SPACES}Q{6 SPACES}Q
{5 SPACES}Q"                                  :rem 54
2002 PRINTTAB(6)"{BLU}{5 SPACES}{3 +}{10 SPACES}
{3 +}{2 SPACES}"                             :rem 246
2003 PRINTTAB(6)"{5 SPACES}{3 +}{10 SPACES}{3 +}
{6 DOWN}"                                     :rem 62

```

## 2: Education

---

```
2004 PRINTTAB(6)"{RED}{11 SPACES}[4 +]{8 SPACES}"
                                         :rem 169
2005 PRINTTAB(6)"{CYN}{2 SPACES}[+]{7 SPACES}{RED}
   [4 +]{8 SPACES}{CYN}[+]"
                                         :rem 52
2006 PRINTTAB(6)"{2 SPACES}[+]{20 SPACES}[+]"
                                         :rem 67
2007 PRINTTAB(6)"{3 SPACES}[+]{18 SPACES}[+]"
                                         :rem 68
2008 PRINTTAB(6)"{4 SPACES}[+]{16 SPACES}[+]{
   {2 SPACES}"
                                         :rem 69
2009 PRINTTAB(6)"{5 SPACES}[+]{14 SPACES}[+]{
   {3 SPACES}"
                                         :rem 70
2010 PRINTTAB(6)"{6 SPACES}[+]{12 SPACES}[+]{
   {4 SPACES}"
                                         :rem 62
2011 PRINTTAB(6)"{7 SPACES}[12 +]{4 SPACES}{WHT}"
                                         :rem 192
2020 GOSUB5010:RETURN
                                         :rem 36
2500 PRINT"{CLR}{10 DOWN}"TAB(16)"{RVS}WRONG!{OFF}"
                                         :rem 250
2510 PRINT"{2 DOWN}"TAB(9)"{RVS}THERE WERE";H/2;"
   {LEFT} OBJECTS{OFF}"
                                         :rem 185
2520 FORT=1TO800:NEXT:FORT=1TO3500:NEXT:PRINTCHR$(
   147):W=W+1:GOTO206
                                         :rem 46
3000 PRINT "{CLR}{10 DOWN}"TAB(10)"YOU HAD:";N;"TR
   IES:PRINT"Q"TAB(18);R;"RIGHT"
                                         :rem 134
3010 PRINT"{DOWN}{2 RIGHT}"TAB(18);W;"WRONG":FORT=
   1TO4000:NEXT:RESTORE:GOTO50
                                         :rem 8
4000 PRINT"{HOME}{3 DOWN}{20 SPACES}";:RETURN
                                         :rem 48
5000 POKES+4,17:POKEHF,INT(RND(0)*50)+80:POKELF,25
   0:POKES+4,16:RETURN
                                         :rem 166
5010 POKES+4,17:FORM=70TO116STEP2:POKEHF,M:POKELF,
   INT(M/2):FORDL=1TO40:NEXT
                                         :rem 22
5020 NEXT:POKES+4,16:RETURN
                                         :rem 206
9000 DATA1,1,65,2,83,3,90,4,88,5,90,6,102,7,42,1,
   35,2,36,3,38,4,1,5
                                         :rem 41
```

# Robot Math

---

Bob Stewart

*Arithmetic becomes an exciting visual delight when children use this educational program. For the unexpanded VIC or the 64.*

**A**lthough the popular use of computers in schools and homes has created a barrage of educational software, much of it fails to take into account many factors which make a learning program truly valuable. Is the program flexible? Is it easy to use? Are there options for children of various levels? And perhaps most importantly, is it fun for the child? "Robot Math" answers each of these questions with *yes*.

## Easy-to-Use Menus

Robot Math is designed to let students practice addition or subtraction problems involving numbers of up to six digits. When entering answers, students should start with the rightmost digit of the answer, just as they would on paper. For instance, consider the following problem:

$$\begin{array}{r} 123 \\ +456 \\ \hline 579 \end{array}$$

When typing in the answer, the student would enter the 9 first, then the 7, and finally the 5. When the no-carry option is selected, that format gives students a chance to practice math problem solving while getting ready for carrying; when the carry option is chosen, it makes it easy to learn the principles of carrying.

After typing in and running the program, you'll see the main menu on your screen. Cursor up or down to choose one of the menu items: operation (+ or -); number of digits (up to six); carry/borrow (yes or no); and number of problems (up to nine).

Simply press RETURN to change the operation or carry/borrow options after you've cursoried to those items. You can also change the number of digits or number of problems. When you're satisfied with the menu choices, press B to begin.

### The Rambling Robot

After the first problem is presented, the timer begins. The problem number appears at the upper-right corner of the screen, directly across from a robot. A limited time and no more than three tries are allowed for each problem. A correct answer is rewarded by the robot, who toddles across the screen and introduces the next problem by updating the number.

If time runs out or if three incorrect answers are entered, the right answer is revealed and a new problem is shown. You can return to the menu at any time by pressing M, or you can delete any digits in your answer with the DELEte key.

### A Tight Fit

In the VIC version, very little memory is available after the program is run. In fact, you'll need to use abbreviations to get some of the lines to fit. In line 9, for instance, use the abbreviation T SHIFT-H for THEN.

The program is self-modifying. This means that once you have configured the program with the menu and have entered the drill mode by pressing B, you may interrupt the program using RUN/STOP and then save the program along with the selections you've made. This self-modifying feature is provided by lines 75 and 76; they change the data contained in line 91 by printing a new line 91 on the screen in white letters (which aren't visible) followed by the command RUN1.

Line 76 POKes three RETURNS (CHR\$(13)) into the keyboard buffer, followed by an END. The program is actually stopped by the END statement, which then causes BASIC to look into the keyboard buffer for further instructions. The first RETURN encountered by BASIC enters the new version of line 91 previously placed on the screen by line 75. The second RETURN skips one line, and the third enters the RUN1 command just as if you entered it from the keyboard. This causes the program to start at the beginning.

### Program 1. Robot Math, VIC Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
1 READP1$,P2,P3$,P4:GOTO62           :rem 254
2 POKEB,Ø:POKEB+1,Ø:POKEB+2,Ø:POKEB+4,Ø:RETURN
                                     :rem 7Ø
3 POKEB+4,15:POKEB+1,18Ø:FORM=1TOC:NEXT:GOTO2
                                     :rem 65
```

```

4 F=INT(RND(1)*9):RETURN                                :rem 214
5 POKED,3:POKED+1,3:POKEB+4,15:FORL=99TO255:POKEB+
  2,L:POKEB,L:NEXT:GOSUB2:GOTO92                        :rem 76
6 POKEB+4,15:POKEB+2,160:FORM=1TO400:NEXT:GOTO2
                                                         :rem 148
7 PRINTM$;" {RVS}{GRN}PRESS RTN TO CHANGE":RETURN
                                                         :rem 200
8 PRINTM$;"{5 SPACES}{RVS}{RED}ENTER DIGIT{OFF}
  {4 SPACES}":RETURN                                    :rem 126
9 IFF>5THENL=126:IFF>9THENL=108:M=1:IFF>30THENL=90
  :M=0:IFF>50THENL=108:M=1:IFF>99THENF=0              :rem 252
10 POKEE,L:POKEE+1,L:POKED+21,M:POKED+24,M:GOTO36
                                                         :rem 22
11 PRINT"{HOME}":PRINTTAB(L);M$;O$;:GOSUB3:PRINTN$
  :RETURN                                                :rem 86
12 M$=" N[2 T]M {DOWN}{6 LEFT} MZZN {DOWN}{6 LEFT}
  {F}[L][D] {DOWN}{6 LEFT} £[Q][W][*] {DOWN}
  {6 LEFT}{2 SPACES}OP{2 SPACES}{DOWN}{6 LEFT}
  {RVS}£{2 SPACES}{*}[OFF] "                          :rem 222
13 N$="{6 LEFT} WWW " :O$="{DOWN}{6 LEFT} ZZZ " :B
  =36874:READP1$,P2,P3$,P4:POKEB+5,30:POKE649,1
                                                         :rem 170
14 PRINT"{CLR}{9 DOWN}{RVS}{GRN} M=MENU-----DEL=ER
  ASE {HOME}":L8=48:POKE143,PEEK(162):GOTO92
                                                         :rem 72
15 POKE651,255:PRINT"{HOME}":R=38649:S=7929:Y=P2:D
  =38446:E=7726:IFP1$="-"ORP2=1ORP2>3THENY=2
                                                         :rem 115
16 FORI=P2TO1STEP-1:A(I)=0:S(I)=0:FORK=1TOY:GOSUB3
  :GOSUB4                                                :rem 139
17 A(I)=A(I)+F:S(I)=S(I)-F                              :rem 144
18 M=(K*22)+I:POKER+M,4:POKES+M,F+48:IFK=1THENL1=F
                                                         :rem 18
19 NEXT:IFP1$="-"THENGOSUB57:GOTO21                    :rem 182
20 L=A(I):N=9:GOTO22                                     :rem 155
21 S(I)=S(I)+2*L1:L=S(I):N=0                            :rem 35
22 GOSUB49:A(I)=L                                       :rem 239
23 NEXTI                                                :rem 237
24 FORK=1TOY:FORI=1TOP2                                 :rem 255
25 M=K*22+I:IFPEEK(S+M)>48THEN28                        :rem 139
26 IFI=P2THENV=1                                        :rem 242
27 POKES+M,32:NEXT                                     :rem 109
28 NEXTK:IFV=1THENV=0:GOTO15                            :rem 76
29 A=0:U=-1:FORI=P2TO1STEP-1:U=U+1:IFP1$="-"THENA=
  A+S(I)*10↑U:GOTO31                                    :rem 142
30 A=A+A(I)*10↑U                                       :rem 102
31 NEXT                                                :rem 163
32 A=INT(A):L2=0:IFA<0THEN15                            :rem 144
33 PRINT"{11 DOWN}":FORI=2TOY:PRINTTAB(7);P1$:NEXT
  :POKE160,0:POKE161,0:POKE162,0                      :rem 93

```

## 2: Education

```
34 PRINT"{4 UP}":FORK=0TOP2:PRINTTAB(7+K);"  
  {3 DOWN}C{DOWN}{LEFT} {5 UP}":NEXT      :rem 135  
35 PRINT"{3 DOWN}":U=LEN(STR$(A))-2:I=0:L1=0:FORK=  
  P2TOP2-USTEP-1                             :rem 246  
36 FORM=6TO8:POKEM+E+154,ASC(MID$(TI$,M-2))+128:NE  
  XT:IFTI$="000400"THENGOSUB6:GOTO46       :rem 66  
37 GETA$:IFA$=""THENL=124:F=F+1:GOTO9       :rem 184  
38 IFASC(A$)=20THENPRINTTAB(7);"{7 SPACES}";"  
  {5 UP}":GOTO35                             :rem 239  
39 IFA$="M"THENPOKEB+5,27:GOTO62           :rem 158  
40 IFA$<"0"ORA$>"9"THEN37                 :rem 98  
41 L1=INT(L1+VAL(A$)*10↑I):I=I+1:PRINTTAB(7+K);A$:  
  PRINT"{2 UP}":NEXT                         :rem 21  
42 IFL1=ATHENGOSUB5                          :rem 212  
43 IFL1<>ATHENGOSUB6                          :rem 19  
44 L2=L2+1:IFL2>2THEN46                     :rem 77  
45 PRINT"{2 UP}":GOTO34                     :rem 54  
46 V=0:AN$=STR$(A):L=LEN(AN$):IFL>P2+1THENV=1  
                                              :rem 208  
47 IFL-1<P2THENV=L-1-P2                     :rem 125  
48 PRINTTAB(8-V);"{RVS}";MID$(AN$,2,8):FORK=1TO350  
  0:NEXT:GOTO15                               :rem 3  
49 IFP3$="N"ANDP1$="+"THEN52                 :rem 44  
50 GOSUB4:X=1:IFL<N+FTHEN54                 :rem 193  
51 RETURN                                     :rem 70  
52 IFL>NTHENX=-1:GOTO54                     :rem 204  
53 RETURN                                     :rem 72  
54 L=0:FORK=1TOY:M=(K*22)+I:F=PEEK(S+M)+X:IFF<48TH  
  ENF=48                                       :rem 176  
55 IFF>57THENF=57                           :rem 7  
56 POKES+M,F:L=L+(F-48):NEXT:GOTO49         :rem 154  
57 IFP3$="N"THEN60                           :rem 6  
58 IFI=1ORF>=L1THENRETURN                   :rem 161  
59 GOTO61                                     :rem 14  
60 IFF<L1THENRETURN                          :rem 3  
61 POKES+M,L1+48:POKES+M-22,F+48:S(I)=(-F)-L1:L1=F  
  :RETURN                                       :rem 149  
62 M$="{HOME}{16 DOWN}":PRINT"{CLR}{3 DOWN}OPERATI  
  ON (+/-).... ";P1$                          :rem 111  
63 PRINT"{DOWN}# DIGITS (MAX=6)...";P2:PRINT"CARRY  
  /BORROW..... ";P3$                          :rem 221  
64 PRINT"{DOWN}# PROBLEMS (MAX=9).";P4:PRINT"  
  {DOWN}{4 SPACES}EEEEEEEEEEEEEEEEEE       :rem 45  
65 GOSUB7:PRINT"{3 DOWN}{5 SPACES}{CYN}{RVS}(B TO  
  {SPACE}BEGIN)";"{GRN}{HOME}SELECT:USE CRSR(UP/D  
  N){BLK}                                       :rem 162  
66 M=7746                                       :rem 206  
67 IFM1=7878THENM=7746                       :rem 38  
68 FORI=MTOM+20:POKEI,PEEK(I)+128:NEXT      :rem 129  
69 GETA$:IFA$=""THEN69:A=A+128:POKEI,A:NEXT:rem 88
```

```

70 IFVAL(A$)<10ANDVAL(A$)>0THEN82           :rem 159
71 IFA$="+ORAS$="-ORAS$="Y"ORAS$="N"ORPEEK(197)=15T
  HENA$="1":GOTO82                           :rem 239
72 IFA$="{DOWN}"THEN77                       :rem 140
73 IFA$="{UP}"THEN81                         :rem 8
74 IFA$<>"B"THEN69                           :rem 253
75 PRINT"{WHT}{CLR}{3 DOWN}91 DATA";P1$;"",P2$;"",
  ;P3$;"",";P4:PRINT"RUN12";"{HOME}"       :rem 158
76 :POKE198,3:POKE631,13:POKE632,13:POKE633,13:END
                                           :rem 164

77 M2=M2-1:K=44                              :rem 49
78 M=M+K:FORI=M-KTOM-K+20:POKEI,PEEK(I)-128:NEXT:I
  FM>7878THENM=7746                          :rem 190
79 IFM<7746THENM=7878                       :rem 247
80 ON(M-7702)/44GOSUB7,8,7,8:GOTO67         :rem 249
81 M2=M2-1:K=-44:GOTO78                    :rem 59
82 ON(M-7746)/44GOTO85,87,90:IFP1$="+THENP1$="-":
  GOTO84                                       :rem 146
83 P1$="+":                                  :rem 188
84 POKEM+20,ASC(P1$)+128:GOTO69             :rem 22
85 IFVAL(A$)>6THENA$="6"                    :rem 84
86 P2=VAL(A$):POKEM+20,P2+176:GOTO69       :rem 98
87 IFP3$="N"THENP3$="Y":GOTO89             :rem 8
88 P3$="N":                                  :rem 230
89 POKEM+20,64+ASC(P3$):GOTO69             :rem 236
90 P4=VAL(A$):POKEM+20,P4+176:GOTO69       :rem 97
91 DATA-, 3 ,Y, 2                          :rem 243
92 C=0:PRINT"{HOME}{BLK}":FORL=0TO15:GOSUB11:NEXT
                                           :rem 65
93 FORL=14TO0STEP-1:GOSUB11:NEXT:PRINT"{BLK}":C=40
                                           :rem 252
94 L8=L8+1:M=34816+8*L8:PRINT"{HOME}":IFL8-48>P4TH
  EN98                                         :rem 79
95 FORM1=MTOM+6:X=PEEK(M1):FORL=1TO7:C=32:X=X*2:IF
  X>255THENX=X-256:C=L8                       :rem 231
96 PRINTTAB(13){"CYN";CHR$(C);:NEXT:PRINT"{BLK}":
  NEXT:IFL8-48>P4THEN98                       :rem 75
97 GOTO15                                     :rem 15
98 POKEB+5,27:PRINT"{CLR}";SPC(176);"{RVS}PLAY ANO
  THER GAME(Y/N){OFF}"                       :rem 48
99 GETZ$:IFZ$="OR(Z$<>"Y"ANDZ$<>"N")THEN99:rem 29
100 IFZ$="N"THENEND                          :rem 115
101 RUN1                                     :rem 184

```

## Program 2. Robot Math, 64 Version

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

10 PRINT"{CLR}":POKE53281,1:POKE53280,5:READP1$,P2
  ,P3$,P4:GOTO710                            :rem 145
12 READP1$,P2,P3$,P4:B=54272                 :rem 18

```

## 2: Education

```

15 FORI=BTOB+24:POKEI,0:NEXT:VO=B+24:AD=B+5:SR=AD+
   1:HF=B+1:LF=B:POKEAD,20          :rem 6
16 POKESR,200:SO=B+4:GOTO130        :rem 186
20 POKESO,32:RETURN                  :rem 222
30 POKEHF,50:POKELF,40:POKESO,33:FORM=1TOC:NEXT:GO
   TO20                                :rem 233
40 F=INT(RND(1)*9):RETURN           :rem 6
50 POKESO,33:FORL=99TO255:POKEHF,L:POKELF,50:NEXT:
   GOSUB20:GOTO1050                  :rem 236
60 POKESO,33:POKEHF,60:POKELF,50:FORM=1TO400:NEXT:
   GOTO20                              :rem 63
70 PRINTM$;"{8 SPACES}{RVS}{BLU}PRESS RETURN TO CH
   ANGE":RETURN                       :rem 229
80 PRINTM$;"{13 SPACES}{RVS}{BLU}ENTER DIGIT{OFF}
   {8 SPACES}":RETURN               :rem 177
90 IFF>5THENL=126:IFF>9THENL=108:M=1:IFF>30THENL=9
   0:M=0:IFF>50THENL=108:M=1       :rem 139
100 IFF>99THENF=0                   :rem 248
110 POKEE,L:POKEE+1,L:GOTO430       :rem 10
120 PRINT"{HOME}":PRINTTAB(L);M$;O$;:POKEVO,15:GOS
   UB30:POKEVO,O:PRINTN$:RETURN     :rem 224
130 POKE649,1:M$=" N[2 T]M {DOWN}{6 LEFT} MZZN
   {DOWN}{6 LEFT} [F]L[D] {DOWN}{6 LEFT} [Q]
   [W][*]{DOWN}{6 LEFT}{2 SPACES}OP" :rem 141
140 M$=M$+"{2 SPACES}{DOWN}{6 LEFT}{RVS}[
   {2 SPACES}[*]{OFF} "           :rem 14
150 N$="{6 LEFT} WWW ":O$="{DOWN}{6 LEFT} ZZZZ "
                                       :rem 232
160 PRINT"{CLR}{9 DOWN}{RVS}{GRN} M=MENU-----
   -----DEL=ERASE {OFF}{HOME}" :rem 181
165 POKE214,23:PRINT:POKE211,15     :rem 73
170 L8=48:POKE143,PEEK(162):GOTO1050 :rem 58
180 POKE651,255:PRINT"{HOME}":R=54272:S=1561:Y=P2:
   E=1106:RW=16:WR=RW-4:POKEVO,15  :rem 213
190 IFP2=3THENRW=17:WR=RW-5        :rem 195
200 IFP1$="-"ORP2=1ORP2>3THENY=2    :rem 144
210 POKE214,RW:PRINT:POKE211,17:PRINT"{7 SPACES}"
                                       :rem 145
220 FORI=P2TO1STEP-1:A(I)=0:S(I)=0:FORK=1TOY:GOSUB
   30:GOSUB40                        :rem 24
230 A(I)=A(I)+F:S(I)=S(I)-F        :rem 189
240 M=(K*40)+I:POKER+S+M,0:POKES+M,F+48:IFK=1THENL
   1=F                                :rem 185
250 NEXT:IFP1$="-"THENGOSUB660:GOTO270 :rem 73
260 L=A(I):N=9:GOTO280              :rem 7
270 S(I)=S(I)+2*L1:L=S(I):N=0      :rem 89
280 GOSUB580:A(I)=L                 :rem 85
290 NEXTI                            :rem 35
300 FORK=1TOY:FORI=1TOP2            :rem 44
310 M=K*40+I:IFPEEK(S+M)>48THEN340  :rem 229

```



```

320 IFI=P2THENV=1 :rem 31
330 POKES+M, 32:NEXT :rem 154
340 NEXTK:IFV=1THENV=0:GOTO180 :rem 172
350 A=0:U=-1:FORI=P2TO1STEP-1:U=U+1:IFP1$="-"THENA
=A+S(I)*10↑U:GOTO370 :rem 241
360 A=A+A(I)*10↑U :rem 156
370 NEXT :rem 217
380 A=INT(A):L2=0:IFA<0THEN180 :rem 249
390 FORI=2TOY:POKE214,WR+I:PRINT:POKE211,17:PRINT"
{BLK}"P1$:NEXT :rem 145
400 POKE160,0:POKE161,0:POKE162,0 :rem 113
410 FORK=0TOP2:POKE214,RW-1:PRINT:POKE211,17+K:PRI
NT"C":NEXT :rem 161
420 U=LEN(STR$(A))-2:I=0:L1=0:FORK=P2TOP2-USTEP-1
:rem 230
430 FORM=15TO17:POKEM+E+R+280,0:POKEM+E+280,ASC(MI
D$(TI$,M-11))+128:NEXT :rem 211
440 IFTI$="000400"THENGOSUB60:GOTO550 :rem 175
450 GETA$:IFA$=" "THENL=124:F=F+1:GOTO90 :rem 23
460 IFASC(A$)=20THENPOKE214,RW:PRINT:POKE211,17:PR
INT"{7 SPACES}":GOTO420 :rem 139
470 IFA$="M"THEN710 :rem 36
480 IFA$<"0"ORA$>"9"THEN440 :rem 200
490 PRINT"{DOWN}":L1=INT(L1+VAL(A$)*10↑I):I=I+1
:rem 135
500 POKE214,RW:PRINT:POKE211,17+K:PRINTA$:NEXT
:rem 163
510 IFL1=ATHENGOTO50 :rem 237
520 IFL1<>ATHENGOSUB60 :rem 115
530 L2=L2+1:IFL2>2THEN550 :rem 173
540 GOTO410 :rem 103
550 V=0:AN$=STR$(A):L=LEN(AN$):IFL>P2+1THENV=1
:rem 0
560 IFL-1<P2THENV=L-1-P2 :rem 173
570 POKE214,RW:PRINT:POKE211,18-V:PRINT"{RVS}";MID
$(AN$,2,8):FORK=1TO3500:NEXT :rem 237
575 GOTO 180 :rem 115
580 IFP3$="N"ANDP1$="+"THEN610 :rem 140
590 GOSUB40:X=1:IFL<N+FTHEN630 :rem 90
600 RETURN :rem 118
610 IFL>NTHENX=-1:GOTO630 :rem 44
620 RETURN :rem 120
630 L=0:FORK=1TOY:M=(K*40)+I:F=PEEK(S+M)+X:IFF<48T
HENF=48 :rem 224
640 IFF>57THENF=57 :rem 55
650 POKES+M,F:L=L+(F-48):NEXT:GOTO580 :rem 250
660 IFP3$="N"THEN690 :rem 111
670 IFI=1ORF>=L1THENRETURN :rem 209
680 GOTO700 :rem 110
690 IFF<L1THENRETURN :rem 60

```

## 2: Education

---

```
700 POKES+M,L1+48:POKES+M-40,F+48:S(I)=(-F)-L1:L1=
F:RETURN :rem 197
710 M$="{HOME}{16 DOWN}" :rem 173
720 PRINT"{CLR}{BLK}{3 DOWN}{8 RIGHT}OPERATION (+/
-)... ";P1$ :rem 201
730 PRINT"{DOWN}{8 RIGHT}# DIGITS (MAX=6)...";P2
:rem 60
740 PRINT"{DOWN}{8 RIGHT}CARRY/BORROW..... ";P3$
:rem 20
750 PRINT"{DOWN}{8 RIGHT}# PROBLEMS (MAX=9).";P4
:rem 135
760 PRINT"{DOWN}{8 RIGHT}{4 SPACES}EEEEEEEEEEEEEEEE
:rem 11
770 GOSUB70:PRINT"{3 DOWN} {8 RIGHT}{4 SPACES}
{BLU}{RVS}(B TO BEGIN)"; :rem 178
780 PRINT"{GRN}{HOME}{8 RIGHT}SELECT:USE CRSR(UP/D
N){BLK}" :rem 241
790 M=1152 :rem 243
800 IFM1=1392THENM=1152 :rem 51
810 FORI=MTOM+20:X=PEEK(I):POKEI,X+128:NEXT
:rem 211
820 GETA$:IFA$=" "THEN820 :rem 87
830 IFVAL(A$)<10ANDVAL(A$)>0THEN950 :rem 7
840 IFA$="+ "ORAS$="- "ORAS$="Y"ORAS$="N"ORPEEK(197)=1T
HENA$="1":GOTO950 :rem 34
850 IFA$="{DOWN}"THEN900 :rem 235
860 IFA$="{UP}"THEN940 :rem 112
870 IFA$<>"B"THEN820 :rem 92
880 PRINT"{WHT}{CLR}{3 DOWN}1040 DATA";P1$;",";P2;
",";P3$;",";P4:PRINT"RUN12";"{HOME}" :rem 79
890 :POKE198,3:POKE631,13:POKE632,13:POKE633,13:EN
D :rem 216
900 M2=M2-1:K=80 :rem 92
910 M=M+K:FORI=M-KTOM-K+20:X=PEEK(I):POKEI,X-128:N
EXT:IFM>1392THENM=1152 :rem 242
920 IFM<1152THENM=1392 :rem 4
930 ON(M-1064)/80GOSUB70,80,70,80:GOTO800 :rem 19
940 M2=M2-1:K=-80:GOTO910 :rem 154
950 ON(M-1152)/80GOTO980,1000,1030:IFP1$="+ "THENP1
$="- ":GOTO970 :rem 204
960 P1$="+ " :rem 240
970 POKEM+20,ASC(P1$)+128:GOTO820 :rem 117
980 IFVAL(A$)>6THENA$="6" :rem 136
990 P2=VAL(A$):POKEM+20,P2+176:GOTO820 :rem 193
1000 IFP3$="N"THENP3$="Y":GOTO1020 :rem 172
1010 P3$="N" :rem 56
1020 POKEM+20,64+ASC(P3$):GOTO820 :rem 105
1030 P4=VAL(A$):POKEM+20,P4+176:GOTO820 :rem 231
1040 DATA-, 2 ,N, 2 :rem 66
```

```
1050 C=0:PRINT"{HOME}{BLK}":FORL=0TO34:GOSUB120:NE
XT                                     :rem 206
1060 FORL=33TO0STEP-1:GOSUB120:NEXT:PRINT"{BLU}":C
=40                                     :rem 24
1070 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)AND2
51                                     :rem 233
1080 L8=L8+1:IFL8-48>P4THEN1120      :rem 5
1090 M=53247+8*L8:PRINT"{HOME}";    :rem 195
1100 FORM1=MTOM+7:X=PEEK(M1):FORL=1TO7:C=32:X=X*2:
IFX>255THENX=X-256:C=209             :rem 83
1110 PRINTTAB(30)"{BLK}"CHR$(C);:NEXT:PRINT"
{7 LEFT}{DOWN}";:NEXT              :rem 19
1120 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1
                                       :rem 179
1130 IFL8-48>P4THEN1150              :rem 41
1140 GOTO180                          :rem 152
1150 PRINT"{CLR}":POKE214,12:PRINT:POKE211,4
                                       :rem 222
1155 PRINT"{RVS}{BLK}HOW ABOUT ANOTHER GAME (Y/N)?
{OFF}"                                :rem 203
1160 GETZ$:IFZ$="OR(Z$<>"Y"ANDZ$<>"N")THEN1160
                                       :rem 201
1170 IFZ$="Y"THENRESTORE:CLR:GOTO10  :rem 242
1180 END                              :rem 161
```

# Homonym Practice

---

Michael A. Tyborski

*This educational program, designed by a schoolteacher, drills young people on the use of homonyms. It works on both the unexpanded VIC-20 and Commodore 64. A screen reformatter is included for the 64.*

**M**y VIC-20 computer is used in a crowded fifth-grade classroom. Since I am busy teaching, I need programs that do not require teacher assistance. To meet this need, I have developed "Homonym Practice" and other educational programs.

Homonym Practice drills students on the homonyms *to*, *two*, and *too* and on *there*, *their*, and *they're*. It also illustrates some of the features that enhance such programs. A standard format allows students to easily work with any one of a series of such programs I have written.

## Friendly Features

Push-button reset is the most important feature. It involves checking the f1 special function key whenever the keyboard is read. If pressed, the program restarts for the next student. This allows many students to use the program without supervision.

In addition, function key f3 turns the program into a learning guide. It recalls examples of properly used homonyms. This is done by the subroutine at line 42. The student can press RETURN to continue the drill. For this type of lesson, the student must type in the correct answer—a feature that helps students learn spelling too.

Unfortunately, typing is an error-prone activity. That made it necessary to use the simulated INPUT routine in lines 29–35. It uses the GET statement to ignore unwanted keys and prevents data entry errors from crashing the program. It even lets students type in apostrophes without using the SHIFT key.

The name entry routine (lines 2–9) also uses the GET statement. It capitalizes the student's name even if the SHIFT key was not used.

Lines 10–20 display directions on a series of screens. It uses more memory, but it's definitely worth it. The subroutine

at line 51 holds the text on the screen until the student presses a key.

### **Random But Not Repetitious**

This program is a tight fit in an unexpanded VIC. Be sure not to type any extra spaces, or you may run out of memory.

This program allows for 16 sentences for each set of homonyms. The first version of the program used random selection. Unfortunately, many repeats occurred. The present method provides better results. It starts at a random point in the list and walks through it in a read-two-skip-one pattern. This assures no repeats in a lesson, and few repeats in any two consecutive lessons. These features have made Homonym Practice an effective classroom aid.

### **Commodore 64 Notes**

The same program (Program 1) works on both the VIC-20 and Commodore 64. However, because the VIC has a 22-column screen and the Commodore 64 supports 40 columns, the screen formatting will appear to be messed up on the 64.

To avoid that problem, Commodore 64 users should type in Program 2. This is a 22-column screen formatter that allows the 64 to emulate the VIC screen. It creates a machine language program which forces the 64 to PRINT within 22 columns; in addition, it centers the image for an attractive display and automatically handles line wraparound. VIC users should not type in Program 2.

This screen formatter first appeared in the November 1983 issue of *COMPUTE!'s Gazette*, with the text-adventure game "Martian Prisoner." If you typed in the formatter for Martian Prisoner, you needn't type it again for Homonym Practice.

Save Program 2 before running it for the first time. When you type RUN, it activates itself. If you ever need to reactivate it (after pressing RUN/STOP-RESTORE, for instance), enter SYS 828.

To use the screen formatter, first load and run it. Then type NEW and load the main program.

**Program 1. Homonym Practice for the VIC and 64**

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

2 PRINTCHR$(14):CH$="1":PRINT"{CLR}{DOWN}HI, I'M M
  S. ENGLISH.", "{DOWN} WHAT'S YOUR NAME?{4 DOWN}"
:rem 226
3 GETC$:IFC$=""THEN3 :rem 141
4 N$=CHR$(ASC(C$)OR128) :rem 0
5 PRINTN$: :rem 111
6 GETC$:IFC$=""THEN6 :rem 147
7 IFASC(C$)=13THENN$=N$+"{4 SPACES}":GOTO10
:rem 137
8 IFASC(C$)=133THEN2 :rem 139
9 N$=N$+C$:PRINTC$:GOTO6 :rem 254
10 PRINT"{CLR}{DOWN} HI, ";N$:PRINT"{3 DOWN} TODAY
  WE'LL PRACTICE":PRINT"{DOWN} SOME HOMONYMS."
:rem 157
11 GOSUB 51:PRINT "{CLR}{DOWN} WOULD YOU LIKE TO",
  "{DOWN} PRACTICE USING", "{2 DOWN}{2 SPACES}1) T
  O{2 SPACES}TWO"; :rem 221
12 PRINT"{2 SPACES}TOO", "{3 DOWN} OR", "{3 DOWN}
  {2 SPACES}2) THERE{3 SPACES}THEIR {DOWN}
  {11 SPACES}THEY'RE" :rem 145
13 PRINTTAB(12)"{2 DOWN}? "; :rem 92
14 GET CH$:IF CH$=""THEN14 :rem 129
15 PRINT CH$:GOSUB51:PRINT"{CLR}{DOWN} IF YOU WANT
  TO SEE", "{DOWN} EXAMPLES"; :rem 82
16 PRINT" OF EACH", "{DOWN} WORD USED IN A", "{DOWN}
  SENTENCE," :rem 166
17 PRINT"{DOWN} JUST PRESS THE", "{DOWN} BROWN BUTT
  ON", "{DOWN} MARKED {RVS} F3 {OFF} .":GOSUB51
:rem 131
18 PRINT"{CLR}{DOWN} YOU MAY USE THE", "{DOWN} BROW
  N {RVS} F3 {OFF} BUTTON", "{DOWN} ANYTIME YOU NE
  ED IT." :rem 142
19 GOSUB51:PRINT"{CLR}{DOWN} YOU MUST TYPE ", "
  {DOWN} THE WORD THAT", "{DOWN} GOES IN THE *** .
  " :rem 169
20 PRINT"{3 DOWN} PRESS {RVS} RETURN {OFF}", "
  {DOWN} AFTER EACH ANSWER.":GOSUB51 :rem 25
21 SC=0:G=0:S=INT((RND(1)*10)+2) :rem 249
22 W=S:IFASC(CH$)=50THENW=S+16 :rem 53
23 RESTORE:FORT=1TOW:READA$,B$:NEXTT :rem 128
24 READA$,B$:S=S+1:IFS>17THENS=1:GOTO22 :rem 123
25 C=C+1:IFC>2THENC=0:GOTO24 :rem 186
26 IFASC(C$)=134THENGOSUB42 :rem 112
27 PRINT"{CLR}{5 DOWN}";A$: :rem 194
28 PRINT"{HOME}{14 DOWN}{4 SPACES}*** = "; :rem 50
29 GET C$:IF C$=""THEN 29 :rem 253
30 IFASC(C$)=55THENC$="" :rem 102

```

```

31 IFASC(C$)=13THEN36 :rem 187
32 IFASC(C$)=133THEN2 :rem 184
33 IFASC(C$)=134THEN26 :rem 240
34 IFASC(C$)=20THENAN$=LEFT$(AN$,LEN(AN$)-1):PRINT
C$;:GOTO29 :rem 74
35 PRINTC$;:AN$=AN$+C$:GOTO29 :rem 228
36 IFAN$=B$THENPRINT"{HOME}{DOWN}VERY GOOD, ";N$:S
C=SC+1:FORT=1TO800:NEXTT:AN$="":GOTO39 :rem 113
37 PRINT"{HOME}{DOWN}SORRY, TRY AGAIN.":AN$="":SC=
SC-1 :rem 166
38 PRINT"{HOME}{14 DOWN}{19 SPACES}":GOTO28:rem 26
39 G=G+1:IFG<10THEN24 :rem 213
40 PRINT"{CLR}{DOWN} ";N$:PRINT"{2 DOWN} YOU GOT "
SC" RIGHT", "{DOWN}{2 SPACES}OUT OF TEN."
:rem 149
41 PRINT"{4 DOWN}{3 SPACES}THAT'S{2 SPACES}";100-(
(10-SC)*10);"%":GOSUB51:GOTO 2 :rem 88
42 IFASC(CH$)=50THEN47 :rem 8
43 PRINT"{CLR}{DOWN} TWO",, "{DOWN}{4 SPACES}I HAVE
TWO TOYS." :rem 231
44 PRINT"{2 DOWN} TOO",, "{DOWN}{4 SPACES}HE ATE TO
O MUCH." :rem 46
45 PRINT"{2 DOWN} TOO",, "{DOWN}{4 SPACES}I WANT SO
ME, TOO." :rem 126
46 PRINT"{2 DOWN} TO",, "{DOWN}{4 SPACES}GO TO THE
{SPACE}STORE.", "{DOWN}{4 SPACES}I WANT TO SEE I
T.":GOTO50 :rem 169
47 PRINT"{CLR}{2 DOWN} THERE",, "{DOWN}{4 SPACES}TH
E BOOK IS OVER{10 SPACES}THERE." :rem 32
48 PRINT"{2 DOWN} THEIR",, "{DOWN}{4 SPACES}THEY LO
ST THEIR{12 SPACES}HATS." :rem 94
49 PRINT"{2 DOWN} THEY'RE",, "{DOWN}{4 SPACES}THEY'
RE GOING HOME{8 SPACES}NOW." :rem 60
50 AN$="":GOSUB51:RETURN :rem 211
51 PRINT"{HOME}{21 DOWN}{4 SPACES}{RVS} PRESS RETU
RN {OFF}{2 SPACES}" :rem 192
52 GETT$:IFT$=""THEN52 :rem 23
53 IFASC(T$)=134THENGOSUB42 :rem 129
54 IFASC(T$)=133THEN2 :rem 205
55 RETURN :rem 74
56 DATA1,1 :rem 19
57 DATA"THAT'S WAY *** MUCH!",TOO :rem 208
58 DATA"I HAD *** MUCH TO EAT{2 SPACES}{DOWN}LAST
{SPACE}NIGHT.",TOO :rem 148
59 DATA"WE'RE GOING *** FAST!",TOO :rem 5
60 DATA"LET'S GO OVER *** MY{2 SPACES}{DOWN} HOUSE
.",TOO :rem 43
61 DATA"MARY WANTS *** COME{4 SPACES}{DOWN} OVER H
ERE.",TOO :rem 27

```

## 2: Education

---

62 DATA "I DON'T KNOW HOW \*\*\*{2 SPACES}{DOWN} DO TH  
IS ONE.",TO :rem 81

63 DATA "PETER THINKS THAT IT'S {DOWN} \*\*\* FAR TO W  
ALK.",TOO :rem 102

64 DATA "THERE ARE \*\*\* TIGERS{2 SPACES}{DOWN} IN TH  
E ZOO.",TWO :rem 185

65 DATA "WHAT IS \*\*\* TIMES{5 SPACES}{DOWN} SIXTY-FO  
UR?.",TWO :rem 199

66 DATA "WHERE IS TRUDY GOING{3 SPACES}{DOWN}\*\*\* LO  
OK FOR IT?",TO :rem 67

67 DATA "LATONIA WOULD LIKE{4 SPACES}{DOWN} SOME IC  
E CREAM, \*\*\*.",TOO :rem 119

68 DATA "LITTLE JIM CAN COME{3 SPACES}{DOWN} ALONG,  
\*\*\*.",TOO :rem 232

69 DATA "I HOPE THERE WON'T{4 SPACES}{DOWN} BE \*\*\*  
{SPACE}MANY.",TOO :rem 64

70 DATA "HOW MUCH WOULD \*\*\*{4 SPACES}{DOWN} HAMBURG  
ERS COST?",TWO :rem 8

71 DATA "THIS WORK IS \*\*\* HARD {DOWN} FOR ALISA.",T  
OO :rem 40

72 DATA "CAN MARK GO TO THE{4 SPACES}{DOWN} PARTY,  
'SPACE'\*\*\* ?",TOO :rem 134

73 DATA "ARE THOSE YOUR BOOKS{2 SPACES}{DOWN} OVER  
{SPACE}\*\*\*?",THERE :rem 30

74 DATA "CAN WE PLAY AT \*\*\*{4 SPACES}{DOWN} HOUSE?"  
,THEIR :rem 124

75 DATA "I'M SURE THAT \*\*\* NOT {DOWN} HOME YET.",TH  
EY'RE :rem 100

76 DATA "THE CHILDREN PUT \*\*\*{2 SPACES}{DOWN} BOOKS  
AWAY.",THEIR :rem 131

77 DATA "TOM AND SUE SAID \*\*\*{2 SPACES}{DOWN} COMIN  
G LATER.",THEY'RE :rem 179

78 DATA "THE BOYS LOST \*\*\*{5 SPACES}{DOWN} BALL.",T  
HEIR :rem 45

79 DATA "IS KIM SURE THAT \*\*\*{2 SPACES}{DOWN} COMIN  
G TONIGHT?",THEY'RE :rem 135

80 DATA "IS \*\*\* A DRAGON IN{4 SPACES}{DOWN} THE CLO  
SET?",THERE :rem 146

81 DATA "CAN YOU SEE \*\*\* BIG{3 SPACES}{DOWN} BLUE E  
YES?",THEIR :rem 153

82 DATA "BILL AND TOM ARE ON{3 SPACES}{DOWN} \*\*\* WA  
Y.",THEIR :rem 121

83 DATA "I THINK THAT \*\*\* TOO{2 SPACES}{DOWN} HIGH  
{SPACE}TO REACH.",THEY'RE :rem 58

84 DATA "THE BOYS LEFT \*\*\*{5 SPACES}{DOWN} JUNK ALL  
OVER",THEIR :rem 56

85 DATA "LOOK OVER \*\*\*.",THERE :rem 12

86 DATA "I THINK \*\*\* GONE.",THEY'RE :rem 12

87 DATA "GIVE ME \*\*\* ADDRESS.",THEIR :rem 100



```

88 DATA "CAN MOLLY TAKE ***{4 SPACES}{DOWN} PLACE?"
    ,THEIR                                     :rem 45
89 DATA 1,1                                   :rem 25
91 RETURN                                       :rem 74

```

## Program 2: 64 Screen Formatter

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

100 PRINT "{CLR}{4 SPACES}{RVS}22-COLUMN PRINT FOR
    MATTER FOR C64":PRINT                       :rem 2
110 PRINT "READING DATA"                       :rem 119
120 FOR I=828 TO 881:READ A:CK=CK+A:POKE I,A:
    NEXT I:POKE I,79,883 AND 255                 :rem 92
130 IF CK <> 6032 THEN PRINT "ERROR IN DATA:CHECK TY
    PING.":END                                   :rem 227
140 PRINT "{DOWN}BEFORE...":SYS 828:PRINT "AFTER..."
    :rem 150
150 PRINT "{DOWN}PRESS RUN/STOP-RESTORE";:PRINT "TO
    REGAIN 40 COLUMNS"                         :rem 228
160 PRINT "{DOWN}ENTER {RVS}SYS 828{OFF} TO":PRINT
    "REACTIVATE, IF":PRINT "NECESSARY."        :rem 115
170 PRINT "{DOWN}DO NOT EDIT ANY":PRINT "LINES WHIL
    E IN 22 COL-UMN MODE."                     :rem 84
1000 DATA 169,71,141,38,3,169,3,141           :rem 180
1010 DATA 39,3,96,72,152,72,138,72           :rem 141
1020 DATA 56,32,240,255,192,9,176,3         :rem 185
1030 DATA 76,100,3,192,31,144,15,169       :rem 226
1040 DATA 13,32,202,241,56,32,240,255      :rem 9
1050 DATA 160,9,24,32,240,255,104,170      :rem 14
1060 DATA 104,168,104,76,202,241           :rem 30

```

# French Tutor

Michael Quigley

*"French Tutor" is a helpful study aid designed for students who are learning or strengthening French vocabulary and translation skills. For the VIC or 64.*

**W**ith two children studying French in elementary school, one of my reasons for buying the VIC-20 was to create some French instructional programs. "French Tutor" was suggested by Steve Steinberg's "Language Lab" (*COMPUTE!*, July 1982), which provided for both vocabulary drill and translation practice.

## Dummy Words and Custom Accents

It was relatively easy to adapt to the VIC, with a few minor modifications. For example, a dummy word (XX) is needed as the last item in the DATA statements to prevent the program from running out of DATA if a particular word is not in the list.

Another modification involves the use of accents, which Language Lab did not include. The solution was to create the accents with programmable characters, as described in "Custom Characters for the VIC" by David Malmberg (*COMPUTE!'s First Book of VIC*).

In addition to the familiar accented vowels, this program includes some which are used less frequently—the umlauted *e* (as in Noël), *u* (as in Saül), and *o* (for words of German origin). Also included are the combined *æ* for words like *œuvre* and *æ* as in *Cæsar*.

## Language Drills

The program is made up of four sections: a French-to-English vocabulary drill, an English-to-French vocabulary drill, a French-to-English translator, and an English-to-French translator. Because of the VIC's memory restraints, there are only 101 words (most of which employ accents and are no longer than five letters). With memory expansion, that total could be increased, although that will necessitate relocating the programmable characters if more than 8K of additional RAM is added.

If you do have more memory and decide to expand the list of words, make the corresponding change in line 43 of Program 2 (line 1610 in Program 3 for the 64), the line that randomly selects the words. If desired, the selection process can be changed so that words will not be repeated.

### Typing In the Programs

Since Program 2 almost fills the unexpanded VIC, do not include unnecessary spaces when typing it in. In addition, in order to make some lines fit, you will have to use abbreviated keywords. PRINT becomes ?, GOSUB becomes GO followed by SHIFT-S, DATA becomes D SHIFT-A, and so on. In particular, lines 1, 3, 4, 5, 60 and 62 require abbreviations.

### Adding Words

The maximum number of words allowed for each vocabulary drill is nine. This is because the computer recognizes only the first integer with the GET A\$ statement in line 10, which doesn't require the user to hit the RETURN key. In other words, if you should type in 20 words, the computer would see it as two words. In order to increase that number to 10 or more, eliminate the question mark from line 9. Also, delete lines 10-11 and 18-20 and replace them with the following:

```
10 INPUTN:IFN<1THEN10
11 IFCO=NTHEN14
```

```
18 INPUTN:IFN<1THEN10
20 IFCO=NTHEN14
```

That will allow numbers larger than nine to be used, but you will have to press RETURN after the number is typed in.

For the 64 version, delete the question mark and (MAX. 9) from line 1270. In addition, delete lines 1280-1290 and 1360-1380 and replace them as follows:

```
1280 INPUTN:IFN<1THEN1280
1290 IFCO=NTHEN1320
```

```
1360 INPUTN:IFN<1THEN1280
1380 IFCO=NTHEN1320
```

VIC French Tutor is in two parts. Program 1 will automatically load and run Program 2. Tape users should type in and save Program 1, then type in and save Program 2 as the next program on the tape. To make the VIC version work with a disk

## 2: Education

---

drive, give Program 2 the name "F". Then, in Program 1, delete line 555 and make the following changes:

```
390 IFA$="N"THENPOKE36869,255:GOTO560      :rem 163
580 POKE7993,34:POKE7994,6:POKE7995,34:POKE7996,44
      :POKE7997,56                          :rem 121
590 POKE198,1:POKE631,131:END              :rem 161
```

### Program 1. VIC French Tutor, Part 1: Redefined Characters

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
10 POKE36879,237                          :rem 105
20 PRINT"{CLR}{4 RIGHT}{4 DOWN}{BLK}"CHR$(122)
                                           :rem 245
30 FORT=1TO8                                :rem 231
40 PRINTTAB(4)"{BLK}"CHR$(125)"{BLU}{RVS}
   {4 SPACES}{WHT}{4 SPACES}{RED}{4 SPACES}"
                                           :rem 161
50 NEXT                                     :rem 164
60 PRINTTAB(4)"{BLK}"CHR$(125)            :rem 13
70 PRINTTAB(4)"{BLK}"CHR$(125)            :rem 14
80 S1=36876:V=36878:POKEV,10              :rem 96
90 READN,D                                  :rem 67
100 IFN=-2THEN170                          :rem 209
110 POKES1,(ABS(N))                        :rem 55
120 FORT=1TO(ABS(D))                      :rem 155
130 NEXTT                                   :rem 39
140 POKES1,0                               :rem 164
150 FORN=1TO20:NEXTN                       :rem 5
160 GOTO90                                  :rem 57
170 FORT=1TO2000:NEXTT                    :rem 115
180 GOTO210                                 :rem 101
190 DATA-201,-125,-201,-187,-201,-62,-215,-250,-21
   5,-250,-219,-250,-219,-250,-228,-375   :rem 112
200 DATA-223,-125,-215,-1000,-2          :rem 247
210 POKE36879,26:PRINT"{CLR}{BLK}{9 DOWN}"TAB(6)"V
   IC FRENCH":PRINTTAB(7)"TUTORIAL"      :rem 70
220 PRINTTAB(7)"[8 T]"                    :rem 220
230 PRINT"{7 DOWN}{RIGHT}DEFINING CHARACTERS"
                                           :rem 30
260 X=PEEK(56)-2:POKE52,X:POKE56,X:POKE51,PEEK(55)
      :CLR                                  :rem 15
270 CS=256*PEEK(52)+PEEK(51)              :rem 23
280 FORI=CSTOCS+511:POKEI,PEEK(I+32768-CS):NEXT
                                           :rem 166
290 READX                                  :rem 15
300 IFX=-1THEN370                          :rem 222
310 IFX<0THEN290                          :rem 177
320 FORI=XTOX+7:READJ                     :rem 87
```

```

330 IFJ<ØTHEN320 :rem 159
340 POKEI,J:NEXT :rem 254
350 GOTO290 :rem 108
370 PRINT"{CLR}{BLK}{10 DOWN}"SPC(5)"INSTRUCTIONS?
      ":PRINT:PRINTSPC(7)"{RED}Y{BLK}ES OR {RED}N
      {BLK}O" :rem 163
380 GETA$:IFA$<>"N"ANDA$<>"Y"THEN380 :rem 51
390 IFA$="N"THENPOKE36869,255:GOTO555 :rem 167
400 PRINT"{CLR}{BLK}{6 DOWN}IN ORDER TO CREATE","F
      RENCH ACCENTS IN","THIS PROGRAM, CERTAIN"
      :rem 8
410 PRINT"LETTERS HAVE BEEN","RE-DEFINED USING","P
      ROGRAMMABLE","CHARACTERS." :rem 215
420 PRINT:PRINT"PRESS {RED}C{BLK} TO CONTINUE."
      :rem 224
430 GETA$:IFA$<>"C"THEN430 :rem 209
440 PRINT"{CLR}{DOWN}{BLK}THE FRENCH CHARACTERS":P
      RINT"ARE {BLU}BLUE{BLK}, THEIR VIC"; :rem 147
450 PRINT"{3 SPACES}EQUIVALENTS {GRN}GREEN{BLK}. "
      :PRINT :rem 207
460 POKE36869,255 :rem 161
470 PRINTTAB(4)"{BLU}← {BLK}{RVS}= {GRN}←{OFF}"SPC
      (4)"{BLU}# {BLK}{RVS}= {GRN}#" :rem 69
480 PRINT:PRINTTAB(4)"{BLU}$ {BLK}{RVS}= {GRN}$
      {OFF}"SPC(4)"{BLU}% {BLK}{RVS}= {GRN}% "
      :rem 155
490 PRINT:PRINTTAB(4)"{BLU}& {BLK}{RVS}= {GRN}&
      {OFF}"SPC(4)"{BLU}+ {BLK}{RVS}= {GRN}+"
      :rem 172
500 PRINT:PRINTTAB(4)"{BLU}£ {BLK}{RVS}= {GRN}£
      {OFF}"SPC(4)"{BLU}@ {BLK}{RVS}= {GRN}@":rem 58
510 PRINT:PRINTTAB(4)"{BLU}* {BLK}{RVS}= {GRN}*
      {OFF}"SPC(4)"{BLU}↑ {BLK}{RVS}= {GRN}↑":rem 19
520 PRINT:PRINTTAB(4)"{BLU}[ {BLK}{RVS}= {GRN}[
      {OFF}"SPC(4)"{BLU}] {BLK}{RVS}= {GRN}]"
      :rem 116
530 PRINT:PRINTTAB(4)"{BLU}= {BLK}{RVS}= {GRN}=
      {OFF}"SPC(4)"{BLU}< {BLK}{RVS}= {GRN}<"
      :rem 247
535 PRINT:PRINTTAB(4)"{BLU}> {BLK}{RVS}= {GRN}>
      {OFF}"SPC(4)"{BLU}/ {BLK}{RVS}= {GRN}/"
      :rem 228
540 PRINT:PRINT"{BLK}PRESS {RED}C{BLK} TO CONTINUE
      ." :rem 115
550 GETA$:IFA$<>"C"THEN550 :rem 215
555 POKE36879,237 :rem 167
560 PRINT"{CLR}{8 DOWN}{2 RIGHT}{BLK}ONE MOMENT PL
      EASE." :rem 235
570 PRINT:PRINTTAB(6)"UN MOMENT,":PRINTTAB(3)"S'IL
      VOUS PLA{T.{WHT}" :rem 180

```

## 2: Education

---

```
580 POKE198,5:POKE631,78:POKE632,69:POKE633,87:POK
    E634,13:POKE635,131:END :rem 27
600 DATA7168,8,16,126,64,126,64,126,0 :rem 84
620 DATA7384,24,36,0,60,66,66,60,0 :rem 187
630 DATA7392,28,34,64,64,34,28,8,16 :rem 253
640 DATA7400,8,20,0,62,8,8,62,0 :rem 25
650 DATA7408,16,8,126,64,126,64,126,0 :rem 86
660 DATA7416,30,40,72,78,72,40,30,0 :rem 231
670 DATA7448,16,8,66,66,66,66,60,0 :rem 211
680 DATA7456,24,36,0,66,66,66,60,0 :rem 199
690 DATA7464,36,0,60,66,66,66,60,0 :rem 199
695 DATA7472,30,40,72,126,72,72,78,0 :rem 44
700 DATA7504,16,8,60,66,126,66,66,0 :rem 243
720 DATA7512,24,36,0,60,66,126,66,0 :rem 232
725 DATA7544,36,0,66,66,66,66,60,0 :rem 203
730 DATA7648,36,0,126,64,126,64,126,0 :rem 85
740 DATA7656,24,36,126,64,126,64,126,0 :rem 139
750 DATA7664,20,0,62,8,8,8,62,0 :rem 39
760 DATA-1 :rem 21
```

### Program 2. VIC French Tutor, Part 2: Vocabulary Drill and Translator

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
1 POKE36879,27:PRINT"{CLR}":L$="FRENCH":K$="ENGLIS
  H":R$="{23 SPACES}":POKE36869,240 :rem 37
2 PRINT"{CLR}{BLU}{RVS}"R$"ENTER DESIRED NUMBER"R$
  :rem 62
3 PRINT:PRINT"{GRN}1){BLK} "L$" TO "K$SPC(5)"VOCAB
  ULARY DRILL":PRINT:PRINT"{GRN}2) {BLK}"K$" TO "L
  $SPC(5)"VOCABULARY"; :rem 60
4 PRINT" DRILL":PRINT:PRINT"{GRN}3) {BLK}"L$" TO "
  K$SPC(5)"TRANSLATOR":PRINT:PRINT"{GRN}4) {BLK}"K
  $" TO "L$SPC(5)"TRANSLATOR" :rem 53
5 PRINT:PRINT:PRINT"{3 RIGHT}{RVS}{CYN}PUSH {RED}R
  ETURN{CYN} AFTER"SPC(5)" EACH WORD INPUT "SPC(5)
  " IN THIS PROGRAM {OFF}" :rem 159
6 GETMQ$:IFVAL(MQ$)<1ORVAL(MQ$)>4THEN6 :rem 114
8 POKE36869,255:ONVAL(MQ$)GOTO9,18,27,35 :rem 60
9 CO=0:SC=0:PRINT"{CLR}{BLK}{DOWN}HOW MANY WORDS?"
  :rem 165
10 GETA$:IFVAL(A$)<1THEN10 :rem 5
11 LETN=VAL(A$):IFCO=NTHEN14 :rem 229
12 PRINT"{CLR}{DOWN}{GRN}TRANSLATE":PRINT"INTO "K$
  ; :rem 147
13 PRINT"{HOME}{2 DOWN}{RED}"TAB(13)CO+1"{LEFT} OF
  "N:GOSUB43 :rem 229
14 IFCO=NTHENGOSUB44:GOTO9 :rem 212
15 PRINT"{HOME}{BLK}{5 DOWN}"W$;:INPUTT$ :rem 39
```

```

16 IFT$=E$THENSC=SC+1:CO=CO+1:PRINT"{BLU}CORRECT":
FORL=1TO1500:NEXTL:GOTO11      :rem 25
17 IFT$<>E$THENCO=CO+1:PRINT"{BLU}WRONG! IT'S
{PUR}";E$:FORT=1TO1500:NEXTT:GOTO11      :rem 91
18 CO=0:SC=0:PRINT"{CLR}{BLK}{DOWN}HOW MANY WORDS?
"                                     :rem 213
19 GETA$:IFVAL(A$)<1THEN19      :rem 23
20 LETN=VAL(A$):IFCO=NTHEN23    :rem 229
21 PRINT"{CLR}{DOWN}{GRN}TRANSLATE":PRINT"INTO "L$
;                                     :rem 148
22 PRINT"{HOME}{2 DOWN}{RED}"TAB(13)CO+1"{LEFT} OF
"N:GOSUB43                          :rem 229
23 IFCO=NTHENGOSUB44:GOTO18    :rem 4
24 PRINT"{HOME}{BLK}{5 DOWN}"E$;:INPUTT$ :rem 21
25 IFT$=W$THENSC=SC+1:CO=CO+1:PRINT"{BLU}CORRECT":
FORL=1TO1500:NEXTL:GOTO20      :rem 43
26 IFT$<>W$THENCO=CO+1:PRINT"{BLU}WRONG! IT'S
{PUR}";W$:FORT=1TO1500:NEXTT:GOTO20    :rem 127
27 PRINT"{CLR}{DOWN}{BLK}ENTER "L$" WORD{5 SPACES}
OR {GRN}M{BLK} TO GO TO MENU"      :rem 137
28 X$="XX":PRINT"{BLK}":INPUTT$      :rem 244
29 IFT$="M"THENRUN              :rem 100
30 READE$,W$                    :rem 143
31 IFW$=T$THENPRINT"{2 RIGHT}{BLU}"E$:FORT=1TO1500
:NEXTT:PRINT:RESTORE:GOTO27      :rem 255
32 IFW$=X$THENPRINT"{2 RIGHT}{BLU}TRY AGAIN.":REST
ORE:GOTO28                        :rem 73
33 IFE$<>T$THEN30                :rem 226
34 PRINT"{BLU}{2 RIGHT}TRY AGAIN.":RESTORE:GOTO28
                                     :rem 89
35 PRINT"{CLR}{DOWN}{BLK}ENTER "K$" WORD{4 SPACES}
OR {GRN}M{BLK} TO GO TO MENU"      :rem 135
36 X$="XX":PRINT"{BLK}":INPUTT$      :rem 243
37 IFT$="M"THENRUN              :rem 99
38 READE$,W$                    :rem 151
39 IFE$=T$THENPRINT"{2 RIGHT}{BLU}"W$:FORT=1TO1500
:NEXTT:PRINT:RESTORE:GOTO35      :rem 6
40 IFE$=X$THENPRINT"{2 RIGHT}{BLU}TRY AGAIN.":REST
ORE:GOTO36                        :rem 53
41 IFW$<>T$THEN38                :rem 251
42 PRINT"{BLU}{2 RIGHT}TRY AGAIN.":RESTORE:GOTO36
                                     :rem 87
43 X=INT(RND(1)*101)+1:RESTORE:FORM=1TOX:READE$,W$
:NEXTM:RETURN                     :rem 49
44 PRINT:PRINT"{CLR}{BLK}{4 DOWN}OUT OF";N;"WORDS
{SPACE}YOU {3 RIGHT}HAVE CORRECTLY"SPC(8)"TRANS
LATED";SC;"{LEFT}."              :rem 122
45 PRINT:PRINT"YOUR SCORE IS";INT((SC/N)*100);SPC(
6)"PER CENT."                     :rem 41

```

## 2: Education

```
46 PRINT:PRINT"GO AGAIN?":PRINT"{DOWN}{2 RIGHT}
  {RED}Y{BLK} - YES":PRINT"{DOWN}{2 RIGHT}{RED}M
  {SPACE}{BLK}- RETURN TO MENU"           :rem 228
47 GETQ$:IFQ$<>"Y"ANDQ$<>"M"THEN47         :rem 2
48 IFQ$="Y"THENRETURN                       :rem 89
49 IFQ$="M"THENRUN                          :rem 99
51 DATASUMMER,@T@,APPLE,POMME,HERE,ICI,THERE,L*,NE
  ST,NID                                     :rem 165
52 DATAHOUSE,MAISON,FARM,FERME,WHERE,O#,SAME,M=ME,
  BOX,BO]TE,FRENCH,FRAN£AIS,CAKE,G+TEAU   :rem 7
53 DATACOW,VACHE,HORSE,CHEVAL,BIRD,OISEAU,CHRISTMA
  S,NO<L,EGG,£UF,EYE,£IL,WORK,£UVRE      :rem 132
54 DATACOST,CO$T,TASTE,GO$T,RATHER,PLUT[T,BELIEVE,
  CRO]RE,HEAD,T=TE,BEAST,B=TE,KEY,CL@    :rem 154
55 DATANAME,NOM,YES,OUI,NO,NON,NOSE,NEZ,COFFEE,CAF
  @,BOY,GAR£ON,DAY,JOUR,CASTLE,CH+TEAU   :rem 217
56 DATABLACK,NOIR,BLUE,BLEU,RED,ROUGE,GREEN,VERT,W
  HITE,BLANC,PURPLE,VIOLET,YELLOW,JAUNE  :rem 144
57 DATAFEBRUARY,F£VRIER,KNOT,N£UD,TASK,T+CHE,PUPIL
  ,£L£VE,PASTE,P+TE,FOREST,FOR=T,OR,OU   :rem 1££
58 DATACHOIR,CH£UR,BONE,OS,BEAR,OURS,GOAT,CH£VRE,C
  ITY,CIT@,NUT,NOIX,MOON,LUNE,BEEF,B£UF  :rem 148
59 DATAFATHER,P£RE,MOTHER,M£RE,BABY,B£B£,FAIRY,F£E
  ,IRON,FER,FIRE,FEU,WINDOW,FEN=TRE      :rem 48
6£ DATARULE,R£GLE,RICE,RIZ,CORN,MA>S,MASTER,MA]TRE
  ,WHEAT,BL@,VERY,TR£S,SOON,T[T,WINE,VIN :rem 37
61 DATALIFE,VIE,JUNE,JUIN,TAIL,QUEUE,FOOT,PIED,ARM
  ,BRAS,WORD,MOT,LEG,JAMBE,CHILD,ENFANT  :rem 21
62 DATASTRONG,FORT,BUILD,B+TIR,AT,*,SWORD,@P£E,FIN
  GER,DOIGT,HEART,C£UR,SKY,CIEL,BEAK,BEC  :rem 61
63 DATAHOUSE,MAISON,DOOR,PORTE,SOAP,SAVON,CUT,COUP
  ,LIP,L£VRE,SCHOOL,@COLE,SUN,SOLEIL     :rem 1£
64 DATAMILK,LAIT,TEA,TH@,WATER,EAU,ARROW,FL£CHE,EN
  D,FIN,AUNT,TANTE,TOOTH,DENT,XX,XX      :rem 79
```

### Program 3. French Tutor, 64 Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
1££ DATA SUMMER,@T@,APPLE,POMME,HERE,ICI,THERE,L*,
  NEST,NID                                     :rem 2££
11£ DATAHOUSE,MAISON,FARM,FERME,WHERE,O#,SAME,M=ME
  ,BOX,BO]TE                                  :rem 183
12£ DATAFRENCH,FRAN£AIS,CAKE,G+TEAU         :rem 252
13£ DATACOW,VACHE,HORSE,CHEVAL,BIRD,OISEAU,CHRISTM
  AS,NO<L                                     :rem 32
14£ DATAEGG,£UF,EYE,£IL,WORK,£UVRE        :rem 19
15£ DATACOST,CO$T,TASTE,GO$T,RATHER,PLUT[T,BELIEVE
  ,CRO]RE                                     :rem 68
16£ DATAHEAD,T=TE,BEAST,B=TE,KEY,CL@       :rem 8
```



```

170 DATANAME, NOM, YES, OUI, NO, NON, NOSE, NEZ, COFFEE, CA
    F@, BOY, GARÇON :rem 152
180 DATA DAY, JOUR, CASTLE, CH+TEAU :rem 246
190 DATABLACK, NOIR, BLUE, BLEU, RED, ROUGE, GREEN, VERT,
    WHITE, BLANC :rem 21
200 DATAPURPLE, VIOLET, YELLOW, JAUNE :rem 42
210 DATAFEBRUARY, F@VRIER, KNOT, N<UD, TASK, T+CHE, PUPI
    L, @L↑VE :rem 98
220 DATAPASTE, P+TE, FOREST, FOR=T, OR, OU :rem 171
230 DATACHOIR, CH<UR, BONE, OS, BEAR, OURS, GOAT, CH↑VRE
    :rem 19
240 DATACITY, CIT@, NUT, NOIX, MOON, LUNE, BEEF, B<UF
    :rem 45
250 DATAFATHER, P↑RE, MOTHER, M↑RE, BABY, B@B@, FAIRY, F@
    E, IRON :rem 193
260 DATAFER, FIRE, FEU, WINDOW, FEN=TRE :rem 30
270 DATARULE, R↑GLE, RICE, RIZ, CORN, MA>S, MASTER, MA]TR
    E, WHEAT, BL@ :rem 81
280 DATAVERY, TR↑S, SOON, T[T, WINE, VIN :rem 143
290 DATALIFE, VIE, JUNE, JUIN, TAIL, QUEUE, FOOT, PIED, AR
    M, BRAS :rem 190
300 DATAWORD, MOT, LEG, JAMBE, CHILD, ENFANT :rem 12
310 DATASTRONG, FORT, BUILD, B+TIR, AT, *, SWORD, @P@E, FI
    NGER, DOIGT :rem 179
320 DATAHEART, C<UR, SKY, CIEL, BEAK, BEC :rem 57
330 DATAHOUSE, MAISON, DOOR, PORTE, SOAP, SAVON, CUT, COU
    P, LIP, L↑VRE :rem 158
340 DATASCHOOL, @COLE, SUN, SOLEIL :rem 30
350 DATAMILK, LAIT, TEA, TH@, WATER, EAU, ARROW, FL↑CHE, E
    ND, FIN, AUNT, TANTE :rem 180
360 DATATOOTH, DENT, XX, XX :rem 80
365 POKE53280, 6 :rem 49
370 POKE53281, 12:CH=54272:FORT=CHTOCH+24:POKET,0:N
    EXT:PRINTCHR$(142) :rem 159
375 POKECH+24, 15 :rem 124
380 POKECH+5, 17:POKECH+6, 241:POKECH, 100 :rem 18
390 PRINT"{CLR}{8 RIGHT}{6 DOWN}{BLK}"CHR$(122)
    :rem 197
400 FORT=1T010 :rem 65
410 PRINTTAB(8)"{BLK}"CHR$(125)"{BLU}{RVS}
    {8 SPACES}{WHT}{8 SPACES}{RED}{8 SPACES}"
    :rem 214
420 NEXT :rem 213
430 PRINTTAB(8)"{BLK}"CHR$(125) :rem 66
440 PRINTTAB(8)"{BLK}"CHR$(125) :rem 67
450 FORT=0T01STEP 0 :READA$:IF A$="XX"THEN T=1:READ
    A$ :rem 71
451 NEXT :rem 217
460 READN, D :rem 116
470 IFD=-2THEN 560 :rem 212

```

## 2: Education

---

```
480 POKECH+1,(ABS(N)):POKECH+4,33           :rem 137
490 FORT=1TO(ABS(D)):NEXT                   :rem 30
510 POKECH+4,32                             :rem 64
520 FORN=1TO20:NEXTN                        :rem 6
530 GOTO460                                  :rem 107
560 PRINTCHR$(14):GOTO590                   :rem 240
570 DATA-16,-125,-16,-187,-16,-62,-22,-250,-22,-25
    0,-25,-250,-25,-250,-33,-375           :rem 230
580 DATA-28,-93,-22,-375,-2,-2           :rem 222
590 PRINT"{CLR}{BLK}{11 DOWN}"TAB(15)"64 FRENCH
    {DOWN}":PRINTTAB(15)"TUTORIAL"         :rem 93
600 PRINTTAB(15)"[8 T]"                    :rem 13
603 IFPEEK(12288)=8THENFORII=1TO2500:NEXT:GOTO740
                                           :rem 31
605 PRINT"{3 DOWN}{3 RIGHT}PLEASE WAIT...DEFINING
    {SPACE}CHARACTERS"                    :rem 19
610 PRINT:PRINT                             :rem 235
620 X=48:POKE56,X                           :rem 241
630 CS=12288                                 :rem 113
640 POKE56334,PEEK(56334)AND254:POKE1,PEEK(1)AND25
    1                                       :rem 187
650 FORI=CSTOCS+4095:POKEI,PEEK(I+40960):NEXT
                                           :rem 24
660 READX                                   :rem 16
670 IFX=-1 THEN740                          :rem 233
680 IFX<0 THEN660                          :rem 188
690 FORI=XTOX+7:READJ                      :rem 97
700 IFJ<0 THEN690                          :rem 170
710 POKEI+5120,J:NEXT                      :rem 242
720 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1
                                           :rem 136
730 GOTO660                                  :rem 111
740 PRINT"{CLR}{BLK}{10 DOWN}"SPC(13)"INSTRUCTIONS
    ?{DOWN}":PRINTSPC(15)"{RVS}Y{OFF}E $\bar{S}$  OR {RVS}N
    {OFF}O"                                :rem 188
750 GETA$:IFA$<>"N"ANDA$<>"Y" THEN750     :rem 53
760 IFA$="N" THEN1150                       :rem 86
770 PRINT"{CLR}{BLK}{6 DOWN}{2 RIGHT}IN ORDER TO C
    REATE FRENCH ACCENTS IN{DOWN}"         :rem 123
780 PRINT"{8 RIGHT}THIS PROGRAM, CERTAIN{DOWN}"
                                           :rem 235
790 PRINT"{2 RIGHT}LETTERS HAVE BEEN RE-DEFINED US
    ING{DOWN}"                             :rem 86
800 PRINT"{8 RIGHT}PROGRAMMABLE CHARACTERS.{DOWN}"
                                           :rem 233
810 PRINT"{10 RIGHT}{5 DOWN}PRESS {RVS}C{OFF} TO C
    ONTINUE."                              :rem 139
820 GETA$:IFA$<>"C" THEN820                :rem 215
830 PRINT"{CLR}{BLK}{4 RIGHT}{BLK}THE FRENCH CHAR
    ACTERS ARE BLACK,"                    :rem 236
```

```

840 PRINT"{4 RIGHT}THEIR 64 EQUIVALENTS ARE {WHT}W
HITE{BLK}."{2 DOWN}" :rem 86
850 POKE53272,29 :rem 102
860 PRINTTAB(12)"< {RVS}={OFF} {RVS}{WHT}<{OFF}
{BLK}"SPC(4)"# {RVS}={OFF} {RVS}{WHT}#{OFF}
{BLK}" :rem 225
870 PRINT:PRINTTAB(12)"$ {RVS}={OFF} {RVS}{WHT}$
{OFF}{BLK}"SPC(4)"% {RVS}={OFF} {RVS}{WHT}%
{OFF}{BLK}" :rem 55
880 PRINT:PRINTTAB(12)"& {RVS}={OFF} {RVS}{WHT}&
{OFF}{BLK}"SPC(4)" + {RVS}={OFF} {RVS}{WHT}+
{OFF}{BLK}" :rem 72
890 PRINT:PRINTTAB(12)"£ {RVS}={OFF} {RVS}{WHT}£
{OFF}{BLK}"SPC(4)"@ {RVS}={OFF} {RVS}{WHT}@
{OFF}{BLK}" :rem 223
900 PRINT:PRINTTAB(12)"* {RVS}={OFF} {RVS}{WHT}*
{OFF}{BLK}"SPC(4)"↑ {RVS}={OFF} {RVS}{WHT}↑
{OFF}{BLK}" :rem 175
910 PRINT:PRINTTAB(12)"[ {RVS}={OFF} {RVS}{WHT}[
{OFF}{BLK}"SPC(4)" ] {RVS}={OFF} {RVS}{WHT}]
{OFF}{BLK}" :rem 16
920 PRINT:PRINTTAB(12)"= {RVS}={OFF} {RVS}{WHT}=
{OFF}{BLK}"SPC(4)"< {RVS}={OFF} {RVS}{WHT}<
{OFF}{BLK}" :rem 147
930 PRINT:PRINTTAB(12)"> {RVS}={OFF} {RVS}{WHT}>
{OFF}{BLK}"SPC(4)"/ {RVS}={OFF} {RVS}{WHT}/
{OFF}{BLK}" :rem 124
940 PRINTTAB(9)"{2 DOWN}{BLK}PRESS {RVS}C{OFF} TO
{SPACE}CONTINUE." :rem 43
950 GETA$:IFA$<>"C"THEN950 :rem 223
952 DATA7168,8,16,126,64,126,64,126,0 :rem 94
960 DATA7384,24,36,0,60,66,66,60,0 :rem 194
965 DATA7392,28,34,64,64,34,28,8,16 :rem 8
970 DATA7400,8,20,0,62,8,8,62,0 :rem 31
975 DATA7408,16,8,126,64,126,64,126,0 :rem 96
980 DATA7416,30,40,72,78,72,40,30,0 :rem 236
985 DATA7448,16,8,66,66,66,66,60,0 :rem 220
990 DATA7456,24,36,0,66,66,66,60,0 :rem 203
995 DATA7464,36,0,60,66,66,66,60,0 :rem 207
1000 DATA7472,30,40,72,126,72,72,78,0 :rem 73
1005 DATA7504,16,8,60,66,126,66,66,0 :rem 34
1015 DATA7512,24,36,0,60,66,126,66,0 :rem 22
1020 DATA7544,36,0,66,66,66,66,60,0 :rem 240
1025 DATA7648,36,0,126,64,126,64,126,0 :rem 131
1030 DATA7656,24,36,126,64,126,64,126,0 :rem 180
1035 DATA7664,34,0,62,8,8,8,62,0 :rem 89
1040 DATA-1 :rem 61
1150 POKE53272,29:CLR:RESTORE :rem 9
1160 PRINT"{CLR}" :L$="FRENCH":K$="ENGLISH" :rem 65
1165 R$="{23 SPACES}" :rem 196

```

## 2: Education

---

```
1170 POKE53280,7 :rem 93
1180 PRINT"{CLR}{DOWN}{WHT}{RVS}"SPC(10)"ENTER DES
      IRED NUMBER" :rem 121
1190 PRINT"{2 DOWN}{WHT}1}{BLK}"L$" TO "K$" {WHT}
      -{BLK} VOCABULARY DRILL" :rem 232
1200 PRINT"{2 DOWN}{WHT}2) {BLK}"K$" TO "L$" {WHT}
      -{BLK} VOCABULARY DRILL" :rem 225
1210 PRINT"{2 DOWN}{WHT}3) {BLK}"L$" TO "K$" {WHT}
      -{BLK} TRANSLATOR" :rem 126
1220 PRINT"{2 DOWN}{WHT}4) {BLK}"K$" TO "L$" {WHT}
      -{BLK} TRANSLATOR" :rem 128
1225 PRINT"{2 DOWN}{WHT}5) {BLK}END THE PROGRAM"
      :rem 128
1230 PRINT"{4 DOWN}{3 RIGHT}{RVS}{WHT}PUSH {BLK}RE
      TURN{WHT} AFTER EACH WORD INPUT{DOWN}";
      :rem 153
1240 PRINTSPC(16)"IN THIS PROGRAM" :rem 29
1250 GETMQ$:IFVAL(MQ$)<1ORVAL(MQ$)>5THEN1250
      :rem 151
1255 IFVAL(MQ$)=5THENSYS2048 :rem 192
1260 POKE53272,29:ONVAL(MQ$)GOTO1270,1360,1450,153
      0 :rem 67
1270 CO=0:SC=0:PRINT"{CLR}{BLK}{3 DOWN}{4 RIGHT}HO
      W MANY WORDS? (MAX. 9)" :rem 106
1280 GETA$:IFVAL(A$)<1THEN1280 :rem 217
1290 N=VAL(A$):IFCO=NTHEN1320 :rem 203
1300 PRINT"{CLR}{4 DOWN}{3 RIGHT}TRANSLATE":PRINT"
      {3 RIGHT}INTO "K$" :rem 124
1310 PRINT"{2 UP}"TAB(24)CO+1"{LEFT} OF"N:GOSUB161
      0 :rem 122
1320 IFCO=NTHENGOSUB1620:GOTO1270 :rem 39
1330 PRINT"{3 DOWN}{7 RIGHT}"W$;:INPUTT$ :rem 142
1340 IFT$=E$THENSC=SC+1:CO=CO+1:PRINT"{7 RIGHT}
      {2 DOWN}{WHT}CORRECT 1{BLK}":FORL=1TO2E2:NEXT
      L :rem 12
1345 IFT$=E$THENT$="":GOTO1290 :rem 225
1350 CO=CO+1:PRINT"{7 RIGHT}{2 DOWN}{WHT}WRONG 1
      {2 SPACES}IT'S {BLK}";E$:FORT=1TO1500:NEXTT:G
      OTO1290 :rem 212
1360 CO=0:SC=0:PRINT"{CLR}{BLK}{3 DOWN}{4 RIGHT}HO
      W MANY WORDS?{SHIFT-SPACE}(MAX. 9)" :rem 10
1370 GETA$:IFVAL(A$)<1THEN1370 :rem 217
1380 LETN=VAL(A$):IFCO=NTHEN1410 :rem 176
1390 PRINT"{CLR}{4 DOWN}{3 RIGHT}TRANSLATE":PRINT"
      {3 RIGHT}INTO "L$; :rem 193
1400 PRINT"{UP}"TAB(24)CO+1"{LEFT} OF"N:GOSUB1610
      :rem 233
1410 IFCO=NTHENGOSUB1620:GOTO1360 :rem 39
```

```
1420 PRINT"{3 DOWN}{7 RIGHT}"E$;:INPUTT$ :rem 124
1430 IFT$=W$THENSC=SC+1:CO=CO+1:PRINT"{WHT}
{2 DOWN}{7 RIGHT}CORRECT !{BLK}":FORL=1TO1500
:NEXTL :rem 59
1435 IFT$=W$THEN1380 :rem 135
1440 CO=CO+1:PRINT"{WHT}{2 DOWN}{7 RIGHT}WRONG ! I
T'S {BLK}";W$:FORT=1TO1500:NEXTT:GOTO1380
:rem 230
1450 PRINT"{CLR}{DOWN}{RIGHT}{BLK}ENTER "L$" WORD
{SPACE}OR {RVS}M{OFF} TO GO TO MENU" :rem 253
1460 X$="XX":PRINT"{DOWN}{RIGHT}";:INPUTT$ :rem 46
1470 IFT$="M"THEN1160 :rem 152
1480 READE$,W$ :rem 249
1490 IFW$=T$THENPRINT"{DOWN}{2 RIGHT}"E$:FORT=1TO1
500:NEXTT:PRINT:RESTORE:GOTO1450 :rem 188
1500 IFW$=X$THENPRINT"{DOWN}{2 RIGHT}TRY AGAIN.":R
ESTORE:GOTO1460 :rem 253
1510 IFE$<>T$THEN1480 :rem 173
1520 PRINT"{DOWN}{2 RIGHT}TRY AGAIN.":RESTORE:GOTO
1460 :rem 13
1530 PRINT"{CLR}{DOWN}{RIGHT}{BLK}ENTER "K$" WORD
{SPACE}OR {RVS}M{OFF} TO GO TO MENU" :rem 251
1540 X$="XX":PRINT"{DOWN}{RIGHT}";:INPUTT$ :rem 45
1550 IFT$="M"THEN1170 :rem 152
1560 READE$,W$ :rem 248
1570 IFE$=T$THENPRINT"{DOWN}{2 RIGHT}"W$:FORT=1TO1
500:NEXTT:PRINT:RESTORE:GOTO1530 :rem 186
1580 IFE$=X$THENPRINT"{DOWN}{2 RIGHT}TRY AGAIN.":R
ESTORE:GOTO1540 :rem 242
1590 IFW$<>T$THEN1560 :rem 198
1600 PRINT"{DOWN}{2 RIGHT}TRY AGAIN.":RESTORE:GOTO
1540 :rem 11
1610 X=INT(RND(1)*101)+1:RESTORE:FORM=1TOX:READE$,
W$:NEXTM:RETURN :rem 146
1620 PRINT:PRINT"{CLR}{BLK}{4 DOWN}{3 RIGHT}OUT OF
";N;"WORDS, YOU HAVE CORRECTLY" :rem 71
1625 PRINT"{12 RIGHT}TRANSLATED";SC;"{LEFT}."
:rem 8
1630 PRINT:PRINT"{6 RIGHT}YOUR SCORE IS";INT((SC/N
)*100);"PER CENT." :rem 203
1640 PRINTTAB(15)"{2 DOWN}GO AGAIN?" :rem 129
1645 PRINT"{DOWN}"TAB(16)"{RVS}Y{OFF} - YES":PRINT
"{DOWN}"TAB(11)"{RVS}M{OFF} - RETURN TO MENU"
:rem 96
1650 GETQ$:IFQ$<>"Y"ANDQ$<>"M"THEN1650 :rem 196
1660 IFQ$="Y"THENRETURN :rem 186
1670 IFQ$="M"THEN1170 :rem 152
```

# Up or Down?

---

C. Regena

*“Up or Down” is a program designed to help beginning music students learn to read music. It illustrates several ways that sound can be used to enhance a program. For the unexpanded VIC or 64.*

**O**ne of the most valuable applications of Commodore sound is in the field of educational programming. This music tutorial takes advantage of the excellent sound capabilities of the VIC and 64 to help beginning students learn to read musical notation.

## Up or Down?

This program is designed for students who are just beginning to read music. The notes are shown on a musical staff, and the student is asked whether the second note is higher, lower, or the same as the first note. In other words, to get from the first note to the second one, do you step up, step down, or stay where you are?

The student is asked to press f1 for up, f3 for no change, or f5 for down. Ten problems are included in each drill. After each correct answer, both notes are played. Incorrect answers produce the “uh-oh” sound. At the end of the drill, the student’s score is displayed, along with the option to try again.

## How It Works

This explanation covers both the VIC and 64 versions of the program.

Line 10 branches past subroutines. Line 10 in the 64 version also POKES 53281,1 to change to a white screen.

Lines 20–40 contain several subroutines. Lines 20–26 print the message to PRESS RETURN, then wait for the student to respond before continuing the program.

Line 30 is a short delay for playing tones for the audible prompt and the “uh-oh” sound for an incorrect response. Line 40 is a delay used in playing the notes shown after the student has pressed the correct answer. The notes are played so the student can hear as well as see the interval.

Lines 100–130 print the title and instruction screen. Line 140 defines L\$ for use in printing the musical staff. To type

this line, use SHIFT and \* to get a horizontal line. For the VIC type 22 lines, and for the 64 type 40 of them.

Lines 150–160 define the tone numbers for playing the notes. The numbers are read in as an array. Two numbers are necessary for each tone in the 64 version.

Line 170 defines the B array. The three numbers are the ASCII codes of the keys f1, f3, and f5. Line 175 POKES values necessary to play music. Line 180 calls the subroutine to wait for the student.

Lines 190–380 present the quiz of ten problems. SC is the score. Line 200 prints the musical staff. Note that a blank line is printed after L\$, because L\$ ends in the last column. You should see five horizontal lines with blank lines between them if you have typed L\$ correctly.

Line 210 chooses a random number for the first note. There are nine possible positions, so  $\text{INT}(9*\text{RND}(0))$  chooses a number from 0 to 8. P1 is the screen memory location calculated, so line 220 can POKE a red circle (representing the note) in the chosen position. Similarly, lines 230–240 choose the second note.

Line 250 calculates the answer. The SGN function returns a value of +1, 0, or -1, depending on whether the number is positive, zero, or negative. Subtracting N2 from N1 determines whether the second note is up from, the same as, or down from the first note. I added 2 to the SGN to get an answer (A). B(A) will be the ASCII code of the correct function key pressed. Line 250 also sets a flag FL to zero.

Line 260 plays the audible prompt (a short, high-pitched tone). Lines 270–290 then receive the student's answer, accepting only the f1, f3, and f5 keys.

Line 300 checks the key pressed. If the answer is incorrect, FL is set to 1, the computer plays an "uh-oh" sound, and the program branches back to line 280 for another answer. If the answer is correct, then lines 350–360 play the notes shown, and line 370 increments the score (if this is the first response).

After ten problems, line 390 prints the score. Although a student must get the correct answer for the program to continue, the score reflects only those answers that were correct on the first try.

Lines 400–420 give the student the option to try the drill again, and the program branches appropriately. Line 430 clears the screen and ends the program.

### Other Possibilities

There are many ways that your Commodore computer could be used to teach musical skills. For instance, a program could be developed to teach the difference between half steps and whole steps or to teach the names of chords. Programs could also teach chord inversions or types of chords, and the computer could then play the chords (a note at a time or together) to reinforce the learning. There are dozens of possibilities.

Music composition can also be enjoyable on the computer. I have seen several programs for nonprogrammers in which a student designs a line by choosing notes and rests and placing them on the staff; then the computer plays what the student composed. You could create similar programs to play one or several notes at a time, allowing even non-composers to experiment with creating musical lines.

These are just a few ideas for using the music capabilities of computers. I'm sure you have other ideas ready to try.

### Program 1. Up or Down?, VIC Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
5 REM STEPPING UP OR DOWN :rem 1
10 GOTO 100 :rem 43
20 PRINT"{DOWN}{BLK}PRESS RETURN{BLU}" :rem 96
22 GETA$:IFA$=" "THEN22 :rem 235
24 IF ASC(A$)<>13THEN22 :rem 243
26 RETURN :rem 72
30 FOR D=1 TO 80:NEXT D:POKE S,0:RETURN :rem 246
40 FOR D=1 TO 500:NEXT D:POKE S,0:RETURN :rem 36
100 PRINT"{CLR}{BLU}":PRINT"STEPPING UP OR DOWN"
:rem 7
110 PRINT"{2 DOWN}TWO NOTES ARE SHOWN.":PRINT"FROM
THE FIRST ONE," :rem 83
120 PRINT"DO YOU GO UP, GO DOWN":PRINT"OR STAY THE
SAME":PRINT"TO PLAY THE SECOND?" :rem 221
130 PRINT"{DOWN}PRESS{2 SPACES}F1 FOR UP":PRINTTAB
(7)"F3 FOR SAME":PRINTTAB(7)"F5 FOR DOWN"
:rem 250
140 L$="*****" :rem 6
150 FOR I=0TO8:READF(I):NEXT :rem 189
160 DATA 232,231,228,225,223,219,215,209,207
:rem 114
```



```

170 B(1)=135:B(2)=134:B(3)=133           :rem 218
175 POKE 36878,15:S=36876                :rem 70
180 GOSUB20                               :rem 123
190 SC=0:FOR T=1 TO 10                   :rem 132
200 PRINT"{CLR}{4 DOWN}{BLK}":FOR I=1TO5:PRINTL$:N
    EXT                                     :rem 42
210 N1=INT(9*RND(0)):P1=7796+N1*22       :rem 96
220 POKE P1,81:POKE P1+30720,2          :rem 72
230 N2=INT(9*RND(0)):P2=7802+N2*22      :rem 89
240 POKE P2,81:POKE P2+30720,2          :rem 76
250 A=SGN(N1-N2)+2:FL=0                 :rem 16
260 POKE S,237:GOSUB 30                 :rem 255
270 PRINT"{3 DOWN}{BLU}F1{2 SPACES}UP":PRINT"F3
    {2 SPACES}SAME":PRINT"F5{2 SPACES}DOWN":rem 64
280 GET A$:IF A$=""THEN 280             :rem 87
290 IF ASC(A$)<133 OR ASC(A$)>135 THEN 280 :rem 88
300 IF ASC(A$)=B(A) THEN 350            :rem 135
310 FL=1:POKE S,159:GOSUB 30           :rem 56
320 POKE S,135:GOSUB 30:GOTO 280       :rem 6
350 POKE S,F(N1):GOSUB 40              :rem 122
360 POKE S,F(N2):GOSUB 40              :rem 124
370 IF FL=0 THEN SC=SC+1               :rem 28
380 NEXT T                               :rem 46
390 PRINT "{2 DOWN}SCORE = ";SC;"OUT OF 10"
                                           :rem 134
400 PRINT "{DOWN}{BLK}TRY AGAIN? (Y/N)" :rem 203
410 GET A$:IF A$="Y" THEN 190           :rem 171
420 IF A$<>"N" THEN 410                 :rem 90
430 PRINT"{CLR}{BLU}":END              :rem 43

```

## Program 2. Up or Down?, 64 Version

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

5 REM STEPPING UP OR DOWN                :rem 1
10 POKE53281,1:GOTO 100                 :rem 244
20 PRINT"{DOWN}{GRN}PRESS RETURN{BLU}" :rem 238
22 GETA$:IFA$=""THEN22                  :rem 235
24 IF ASC(A$)<>13THEN22                   :rem 243
26 RETURN                                 :rem 72
30 FORD=15TO1STEP-1:POKEW,33:POKEW+20,D:FORL=1TO80
    :NEXT:POKE W,0:NEXT:RETURN           :rem 188
35 POKE 54296,15:POKE W,33:FOR D=1 TO 125:NEXT:POK
    E W,0:RETURN                         :rem 66
40 POKE 54296,15:POKE W,33:FOR D=1 TO 500:NEXT D:P
    OKE W,0:RETURN                       :rem 127
100 PRINT"{CLR}{BLU}":PRINTTAB(10)"STEPPING UP OR
    {SPACE}DOWN"                         :rem 144
110 PRINT"{2 DOWN}YOU WILL SEE TWO NOTES.":PRINT"F
    ROM THE FIRST ONE, DO YOU MOVE UP," :rem 150

```

## 2: Education

---

```
120 PRINT"MOVE DOWN, OR STAY THE SAME TO PLAY THE
    {SPACE}SECOND NOTE?" :rem 211
130 PRINT"{DOWN}PRESS{2 SPACES}F1 FOR UP":PRINTTAB
    (7)"F3 FOR SAME":PRINTTAB(7)"F5 FOR DOWN" :rem 250
140 L$="*****" :rem 134
150 FOR I=0TO8:READHF(I),LF(I):NEXT :rem 93
160 DATA 44,193,42,62,37,162,33,135,31,165,28,49,2
    5,30,22,96,21,31 :rem 173
170 B(1)=135:B(2)=134:B(3)=133 :rem 218
175 V1=54273:V2=54272:W=54276:POKE 54277,64:POKE 5
    4278,128 :rem 129
180 GOSUB20 :rem 123
190 SC=0:FOR T=1 TO 10 :rem 132
200 PRINT"{CLR}{5 DOWN}{BLK}":FOR I=1TO5:PRINTL$:N
    EXT :rem 59
210 N1=INT(9*RND(0)):P1=1280+N1*40 :rem 78
220 POKE P1,81:POKE P1+54272,2 :rem 80
230 N2=INT(9*RND(0)):P2=1287+N2*40 :rem 90
240 POKE P2,81:POKE P2+54272,2 :rem 84
250 A=SGN(N1-N2)+2:FL=0 :rem 16
260 POKE V1,56:POKE V2,99:GOSUB 30 :rem 145
270 PRINT"{3 DOWN}{BLU}F1{2 SPACES}UP":PRINT"F3
    {2 SPACES}SAME":PRINT"F5{2 SPACES}DOWN":rem 64
280 GET A$:IF A$=""THEN 280 :rem 87
290 IF ASC(A$)<133 OR ASC(A$)>135 THEN 280 :rem 88
300 IF ASC(A$)=B(A) THEN 350 :rem 135
310 FL=1:POKE V1,10:POKE V2,143:GOSUB 35 :rem 232
320 POKE V1,8:POKE V2,97:GOSUB 35:GOTO 280:rem 107
350 POKE V1,HF(N1):POKE V2,LF(N1):GOSUB 40:rem 117
360 POKE V1,HF(N2):POKE V2,LF(N2):GOSUB 40:rem 120
370 IF FL=0 THEN SC=SC+1 :rem 28
380 NEXT T :rem 46
390 PRINT "{2 DOWN}SCORE = ";SC;"OUT OF 10" :rem 134
400 PRINT "{DOWN}{GRN}TRY AGAIN? (Y/N)" :rem 89
410 GET A$:IF A$="Y" THEN 190 :rem 171
420 IF A$<>"N" THEN 410 :rem 90
430 PRINT"{CLR}{BLU}":END :rem 43
```

# Build a Quiz

---

Clark and Kathy Kidd

*"Build a Quiz" makes it easy to create multiple-choice tests on any subject you choose. For any VIC or 64.*

**"Build a Quiz"** is a versatile educational tool that turns your VIC or 64 into a tutor in virtually any subject you choose. It lets you create a quiz on any subject and then save it to tape or disk.

Assume that your child isn't doing well in civics and that there's a big test coming up. You can create a sample test covering anything from the U.S. Constitution to your local government, using multiple-choice, true/false, fill-in-the-blank questions, or any combination thereof.

On the Commodore 64, to save to tape or disk, simply connect your tape drive or disk drive and press the appropriate buttons when the program instructs you to do so. On the VIC, however, you'll need to change the value of DV in line 100 of the VIC version to reflect the storage device you're using. If your system includes a disk drive, change DV to 8. That signifies that you will be reading from and writing to device 8, the disk drive. For VIC tape operation, DV should equal 1.

You are then asked if you want to create a new quiz or answer the questions from an existing quiz. Creating a new quiz is a simple process involving naming the quiz, choosing the type of question for each question, writing the questions, and providing the correct answers.

The computer will prompt you at each step, asking for the question and then for the answer. For instance, when you write a multiple-choice question, you'll be asked to give four possible answers. After all four have been typed in, the computer will ask you to show the correct answer by pressing A, B, C, or D on the keyboard.

You can write as many questions as you like, or you can quit by pressing 4. At that point you'll return to the initial screen and can save the program by pressing 3.

Later you can load the program and take the quiz by typing in the quiz name. The questions you entered will be dis-

played in the order in which they were typed. Correct answers are greeted with a musical tone, while incorrect answers are noted by a buzzing sound and a black screen.

When the quiz is completed, your final score—including the number of questions asked, the number correct, and your percentile score—is shown. If you want, you can continue with another quiz or write a new one.

Build a Quiz is a valuable educational tool, but it can be a lot of fun too. For instance, it's ideal for trivia games. But however you decide to use it, it's sure to make teaching—and learning—more fun than ever before.

### Program 1. Build a Quiz, VIC Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
1 REM VIC BUILD A QUIZ :rem 241
100 POKE36879,190:Z$=CHR$(13):DV=1 :rem 132
120 M=4*(PEEK(36866)AND128)+64*(PEEK(36869)AND120)
:C=37888+4*(PEEK(36866)AND128) :rem 80
200 PRINT"{BLK}{CLR}{DOWN}{5 SPACES}{RVS}BUILD A Q
UIZ{OFF}" :rem 53
210 PRINT"{DOWN}OPTION 1 OF THIS{6 SPACES}PROGRAM
{SPACE}WILL BUILD A{2 SPACES}QUIZ AND WRITE IT
TO" :rem 52
215 PRINT"TAPE OR DISK." :rem 141
220 PRINT"{DOWN}OPTION 2 ALLOWS YOU TOTEST YOURSEL
F WITH A{2 SPACES}PREVIOUSLY WRITTEN{4 SPACES}
QUIZ." :rem 239
230 PRINT"{2 DOWN} ENTER OPTION:" :rem 25
240 PRINT"{DOWN} {RVS}1{OFF} CREATE A QUIZ":PRINT"
{DOWN} {RVS}2{OFF} TAKE A QUIZ":PRINT"{DOWN}
{RVS}3{OFF} END PROGRAM" :rem 14
250 GETX$:IFX$=""THEN250 :rem 127
260 X=VAL(X$):ONXGOTO300,700,280 :rem 97
270 GOTO250 :rem 105
280 POKE36879,27:PRINT"{CLR}":END :rem 30
300 NQ=0:GOSUB7500 :rem 37
310 OPEN9,DV,1,QN$:GOSUB6500 :rem 150
320 NQ=NQ+1:PRINT"{CLR}{DOWN} ENTER QUESTION TYPE:
{2 DOWN}" :rem 175
330 PRINT"{DOWN} {RVS}1{OFF} TRUE/FALSE":PRINT"
{DOWN} {RVS}2{OFF} MULTIPLE CHOICE":PRINT"
{DOWN} {RVS}3{OFF} COMPLETION" :rem 29
340 PRINT"{DOWN} {RVS}4{OFF} (ALL DONE)" :rem 161
350 GETX$:IFX$=""THEN350 :rem 129
360 IFX$="4"THEN550 :rem 34
```

```

365 IFX$<"1"ORX$>"3"THEN350 :rem 243
370 GOSUB6500:PRINT"{CLR} QUESTION #";NQ; :rem 51
380 X=VAL(X$):ONXGOTO400,450,500 :rem 98
390 GOTO350 :rem 109
400 PRINT"(T/F)":O=88:L=66:GOSUB8000:GOSUB6500 :rem 235
410 PRINT"{8 DOWN} ENTER {RVS}T{OFF} OR {RVS}F {OFF}" :rem 239
420 GETX$:IFX$=""THEN420 :rem 125
430 IFX$<"T"ANDX$<"F"THEN420 :rem 204
440 GOSUB6500:GOTO320 :rem 235
450 PRINT"(M.C.):O=44:L=66:GOSUB8000:GOSUB6500:PRINT"{5 DOWN}A.":O=135:L=63:GOSUB8000 :rem 231
460 GOSUB6500:PRINT"{3 DOWN}B.":O=223:GOSUB8000:GOSUB6500:PRINT"{3 DOWN}C.":O=311:GOSUB8000 :rem 131
470 GOSUB6500:PRINT"{3 DOWN}D.":O=399:GOSUB8000:GOSUB6500 :rem 8
480 PRINT"{2 DOWN}{2 SPACES}(PRESS {RVS}A{OFF},{RVS}B{OFF},{RVS}C{OFF} OR {RVS}D{OFF})"; :rem 59
485 GETX$:IFX$=""THEN485 :rem 147
490 IFX$<"A"ORX$>"D"THEN485 :rem 28
495 GOSUB6500:GOTO320 :rem 245
500 PRINT"(COMP.):O=66:L=69:GOSUB8000:GOSUB6500:PRINT"{7 DOWN}ENTER CORRECT ANSWER" :rem 97
510 O=242:L=63:GOSUB8000:GOSUB6500:GOTO320:rem 245
550 GOSUB6500:CLOSE9:GOTO200 :rem 211
700 GOSUB7500 :rem 227
730 OPEN9,DV,0,QN$:GOSUB6600 :rem 156
740 IFLEN(X$)>21THEN760 :rem 67
750 A$="" +X$:X$=A$+" ":GOTO740 :rem 254
760 PRINT"{CLR}{2 DOWN}":FORX=1TO10:PRINTX$;:PRINT " ":NEXTX:GOSUB7000:FORX=1TO1000:NEXTX :rem 5
770 NQ=0:CQ=0 :rem 229
800 GOSUB6600 :rem 228
810 IFX$="4"THEN2000 :rem 74
820 IFX$<"1"ORX$>"3"THEN800 :rem 239
830 NQ=NQ+1:X=VAL(X$) :rem 241
840 PRINT"{CLR} QUESTION #";NQ :rem 117
850 ONXGOTO900,1000,1100 :rem 63
860 GOTO800 :rem 111
900 GOSUB6600:PRINT"{DOWN}";X$ :rem 184
910 PRINT"{2 DOWN} ENTER {RVS}T{OFF} FOR TRUE" :rem 42
920 PRINT"{2 DOWN} ENTER {RVS}F{OFF} FOR FALSE" :rem 72
930 GOSUB6600 :rem 232
940 GETA$:IFA$=""THEN940 :rem 93
950 IFA$<"T"ANDA$<"F"THEN940 :rem 172

```

## 2: Education

---

```
960 IFX$<>A$THEN1200 :rem 123
970 GOTO1300 :rem 157
1000 GOSUB6600:PRINT"{DOWN}";X$:GOSUB6600:PRINT"
      {DOWN}A. ";X$:GOSUB6700 :rem 47
1010 GOSUB6600:PRINT"B. ";X$:GOSUB6700:GOSUB6600:P
      RINT"C. ";X$:GOSUB6700 :rem 7
1020 GOSUB6600:PRINT"D. ";X$:GOSUB6700:GOSUB6600
      :rem 80
1030 PRINT"{3 SPACES}ENTER {RVS}A{OFF},{RVS}B{OFF}
      ,{RVS}C{OFF} OR {RVS}D{OFF}"; :rem 225
1040 GETA$:IFA$=""THEN1040 :rem 173
1050 IFA$<"A"ORA$>"D"THEN1040 :rem 59
1060 IFX$<>A$THEN1200 :rem 163
1070 GOTO1300 :rem 197
1100 GOSUB6600 :rem 14
1110 PRINT"{DOWN}";X$:GOSUB6600 :rem 226
1120 INPUT"{3 DOWN}";A$ :rem 107
1130 IFX$<>A$THEN1200 :rem 161
1140 GOTO1300 :rem 195
1200 GOSUB7200:PRINT"{CLR}{DOWN} ANSWER = ";X$
      :rem 127
1210 FORX=1TO1500:NEXTX:GOTO800 :rem 182
1300 CQ=CQ+1:GOSUB7000:GOTO800 :rem 17
2000 PRINT"{CLR}{DOWN}{6 SPACES}{RVS}QUIZ OVER!
      {OFF}" :rem 129
2010 PRINT"{2 DOWN} # QUESTIONS =";NQ :rem 187
2020 PRINT"{2 DOWN} # CORRECT{3 SPACES}=";CQ
      :rem 248
2030 IFNQ=0THENNQ=1 :rem 156
2040 X=INT((CQ*100)/NQ) :rem 5
2050 PRINT"{2 DOWN} YOUR SCORE{2 SPACES}=";X;"
      {LEFT}%" :rem 150
2060 PRINT"{3 DOWN}{3 SPACES}(PRESS ANY KEY)"
      :rem 123
2070 CLOSE 9 :rem 120
2080 GETX$:IFX$=""THEN2080 :rem 229
2090 GOTO200 :rem 150
6500 PRINT#9,X$;Z$;:RETURN :rem 106
6600 X$="" :rem 201
6610 GET#9,QQ$ :rem 251
6620 IFQQ$=""THEN6610 :rem 160
6630 IFASC(QQ$)=13THENRETURN :rem 252
6640 X$=X$+QQ$ :rem 246
6650 GOTO6610 :rem 215
6700 X=LEN(X$):IFX=19ORX=41ORX=63THENRETURN:rem 31
6710 PRINT"{SHIFT-SPACE}":RETURN :rem 89
7000 POKE36878,15:FORX=110TO190STEP16:POKE36879,X:
      POKE36875,X+30:FORY=1TO100:NEXTY,X :rem 184
7010 POKE36878,0:POKE36875,0:RETURN :rem 79
```

```

7200 POKE36879,8:POKE36878,15:POKE36874,130:FORX=1
      TO500:NEXTX                               :rem 157
7210 POKE36879,190:POKE36878,0:POKE36874,0:RETURN
      :rem 144
7500 PRINT"{CLR}{3 DOWN}{3 SPACES}ENTER QUIZ NAME:
      ":PRINT"{DOWN}{4 SPACES}(1-22 LETTERS)"
      :rem 215
7510 O=176:L=23:GOSUB80000:IFLEN(X$)>22THEN7500
      :rem 179
7520 QN$=X$:IFLEN(X$)>11THENQN$=LEFT$(X$,11)
      :rem 56
7530 QN$="QUIZ/"+QN$                             :rem 121
7540 PRINT"{6 DOWN}{2 SPACES}PREPARE TAPE/DISK":PR
      INT"{2 SPACES}THEN PRESS {RVS}RETURN{OFF}"
      :rem 229
7550 GETES:IFES$=""THEN7550                       :rem 205
7560 IFASC(ES)<>13THEN7550                         :rem 208
7570 RETURN                                       :rem 179
8000 X$="":POKEM+O,160:POKEC+O,0                :rem 58
8010 GETY$:IFY$=""THEN8010                       :rem 229
8020 X=ASC(Y$):IFX=13THEN8150                    :rem 195
8030 IFX=20THEN8100                              :rem 73
8040 Y=LEN(X$):X$=X$+Y$:IFX>63THENX=X-64       :rem 160
8050 POKEM+O+Y,X:POKEC+O+Y,0:POKEM+O+Y+1,160:POKEC
      +O+Y+1,0:IFLEN(X$)<LTHEN8010                :rem 251
8060 GOTO8150                                     :rem 213
8100 Y=LEN(X$):IFY<1THEN8010                     :rem 146
8110 POKEM+O+Y,32:POKEC+O+Y,1:POKEM+O+Y-1,160:POKE
      C+O+Y-1,0                                     :rem 21
8120 Y$=LEFT$(X$,Y-1):X$=Y$:GOTO8010            :rem 48
8150 Y=LEN(X$):POKEM+O+Y,32:POKEC+O+Y,1:RETURN
      :rem 118

```

## Program 2. Build a Quiz, 64 Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

1 REM BUILD A QUIZ                               :rem 15
100 DI=8:TA=1:REM ** DEVICE NUMBERS **         :rem 98
110 VC=53248:POKEVC+32,0:POKEVC+33,11:PRINTCHR$(14
      7)                                           :rem 50
120 Z$=CHR$(13):DP=0:PS=0                       :rem 137
130 DIM NT%(30)                                  :rem 233
140 MC=54272:FORX=MCTO54296:POKEX,0:NEXTX     :rem 71
150 FORX=0TO30:READZ:NT%(X)=Z:NEXTX           :rem 10
160 M=((PEEK(53272)AND240)/16)*1024            :rem 44
170 C=55296                                       :rem 35
200 PRINT"{WHT}{2 DOWN} {RVS}{6 SPACES}B U I L D
      {4 SPACES}A{4 SPACES}Q U I Z{7 SPACES}{OFF}"
      :rem 40
210 PRINT"{2 DOWN} OPTION 1 OF THIS PROGRAM WILL B
      UILD A"                                     :rem 94

```

## 2: Education

---

```
215 PRINT" QUIZ AND SAVE IT ON TAPE OR DISK."
                                     :rem 18
220 PRINT "{2 DOWN} OPTION 2 LETS YOU RECALL AN EX
      ISTING"                         :rem 116
225 PRINT" QUIZ AND TEST YOURSELF."   :rem 109
227 GOSUB9000                          :rem 228
230 PRINT"{2 DOWN}{3 SPACES}ENTER OPTION:" :rem 25
240 PRINT"{DOWN}{3 SPACES}{RVS}1{OFF} CREATE A QUI
      Z":PRINT"{DOWN}{3 SPACES}{RVS}2{OFF} TAKE A QU
      IZ"                               :rem 44
245 PRINT"{DOWN}{3 SPACES}{RVS}3{OFF} END PROGRAM"
                                     :rem 67
250 GETX$:IFX$=""THEN250               :rem 127
260 X=VAL(X$):ONXGOTO300,700,280      :rem 97
270 GOTO250                             :rem 105
280 PS=0:GOSUB9000:PRINTCHR$(147)    :rem 226
290 END                                  :rem 114
300 NQ=0:GOSUB9200:GOSUB7500:PRINTCHR$(147):rem 95
310 OPEN9,DV,1,QN$:PRINTCHR$(147):PRINT#9,X$:Z$;
                                     :rem 133
320 NQ=NQ+1:PRINT"{CLR}{2 DOWN}{3 SPACES}ENTER QUE
      STION TYPE:{2 DOWN}"            :rem 192
330 PRINT"{2 DOWN}{3 SPACES}{RVS}1{OFF} TRUE/FALSE
      ":PRINT"{2 DOWN}{3 SPACES}{RVS}2{OFF} MULTIPLE
      CHOICE"                          :rem 82
335 PRINT"{2 DOWN}{3 SPACES}{RVS}3{OFF} COMPLETION
      "                                :rem 95
340 PRINT"{2 DOWN}{3 SPACES}{RVS}4{OFF} (ALL DONE)
      "                                :rem 178
350 GETX$:IFX$=""THEN350              :rem 129
360 IFX$="4"THENT$=X$:GOTO550         :rem 198
365 IFX$<"1"ORX$>"3"THEN350         :rem 243
370 PRINT"{CLR}{DOWN}{3 SPACES}QUESTION #";NQ;
                                     :rem 191
380 X=VAL(X$):T$=X$:ONXGOTO400,450,500 :rem 205
390 GOTO350                             :rem 109
400 PRINT"(TRUE/FALSE)":O=240:L=80:GOSUB8000:A$=X$
                                     :rem 241
410 PRINT"{10 DOWN}{3 SPACES}ENTER CORRECT ANSWER
      {SPACE}({RVS}T{OFF} OR {RVS}F{OFF})" :rem 68
420 GETX$:IFX$=""THEN420              :rem 125
430 IFX$<"T"ANDX$<"F"THEN420        :rem 204
435 PRINTCHR$(147)                     :rem 23
440 PRINT#9,T$:Z$:A$:Z$:X$:Z$:;GOTO320 :rem 234
450 PRINT"(MULTIPLE CHOICE)":O=240:L=80:GOSUB8000
                                     :rem 219
455 Q$=X$:PRINT"{8 DOWN} A.":O=404:L=76:GOSUB8000
                                     :rem 222
460 A$=X$:PRINT"{2 DOWN} B.":O=524:GOSUB8000:B$=X$
                                     :rem 145
```



```

465 PRINT"{2 DOWN} C.":O=644:GOSUB8000:C$=X$
:rem 67
470 PRINT"{2 DOWN} D.":O=764:GOSUB8000:D$=X$
:rem 68
480 PRINT"{3 DOWN}{4 SPACES}ENTER CORRECT ANSWER (
{RVS}A{OFF},{RVS}B{OFF},{RVS}C{OFF} OR {RVS}D
{OFF})"
:rem 228
485 GETX$:IFX$=""THEN485
:rem 147
490 IFX$<"A"ORX$>"D"THEN485
:rem 28
495 PRINTCHR$(147)
:rem 29
497 PRINT#9,T$;Z$;Q$;Z$;A$;Z$;B$;Z$;C$;Z$;D$;Z$;X$
;Z$;
:rem 104
499 GOTO320
:rem 116
500 PRINT"(COMPLETION)":O=240:L=80:GOSUB8000:Q$=X$
:rem 34
505 PRINT"{10 DOWN}{3 SPACES}ENTER CORRECT ANSWER:
"
:rem 175
510 O=600:L=80:GOSUB8000
:rem 101
520 PRINTCHR$(147)
:rem 18
530 PRINT#9,T$;Z$;Q$;Z$;X$;Z$;
:rem 242
540 GOTO320
:rem 103
550 PRINTCHR$(147)
:rem 21
560 PRINT#9,T$;Z$;
:rem 28
570 CLOSE9
:rem 75
580 GOTO200
:rem 104
700 GOSUB9200:GOSUB7500:PRINTCHR$(147)
:rem 29
730 OPEN9,DV,0,QN$:GOSUB6000
:rem 150
740 IFLEN(X$)>39THEN760
:rem 76
750 A$=" "+X$:X$=A$+" ":GOTO740
:rem 254
760 PRINT"{CLR}{2 DOWN}":FORX=1TO11:PRINTX$;:PRINT
" ":NEXTX:GOSUB7000:FORX=1TO1500:NEXTX
:rem 11
770 PRINTCHR$(147):NQ=0:CQ=0
:rem 154
800 INPUT#9,X$
:rem 44
810 IFX$="4"THEN2000
:rem 74
820 IFX$<"1"ORX$>"3"THEN800
:rem 239
830 NQ=NQ+1:X=VAL(X$)
:rem 241
850 ONXGOTO900,1000,1100
:rem 63
860 GOTO800
:rem 111
900 GOSUB6000:Q$=X$:GOSUB6000:R$=X$
:rem 48
902 PRINT"{CLR}{DOWN}{3 SPACES}QUESTION #";NQ
:rem 133
904 PRINT"{3 DOWN}";Q$
:rem 81
910 PRINT"{4 DOWN}{3 SPACES}ENTER {RVS}T{OFF} FOR
{SPACE}TRUE"
:rem 76
920 PRINT"{2 DOWN}{3 SPACES}ENTER {RVS}F{OFF} FOR
{SPACE}FALSE"
:rem 72
940 GETA$:IFA$=""THEN940
:rem 93
950 IFA$<>"T"ANDA$<>"F"THEN940
:rem 172
960 IFA$<>R$THEN1200
:rem 117
970 GOTO1300
:rem 157

```

## 2: Education

---

```
1000 GOSUB6000:Q$=X$:GOSUB6000:A$=X$:GOSUB6000:B$=
      X$:GOSUB6000:C$=X$                               :rem 250
1002 GOSUB6000:D$=X$:GOSUB6000:R$=X$                   :rem 77
1006 PRINT"{CLR}{DOWN}{3 SPACES}QUESTION #";NQ
                                             :rem 177
1008 PRINT"{3 DOWN}";Q$:PRINT"{2 DOWN} A. ";A$:X$=
      A$:GOSUB6700                                       :rem 152
1010 PRINT" B. ";B$:X$=B$:GOSUB6700:PRINT" C. ";C$
      :X$=C$:GOSUB6700                                   :rem 131
1020 PRINT" D. ";D$:X$=D$:GOSUB6700                   :rem 139
1030 PRINT"{12 SPACES}ENTER {RVS}A{OFF},{RVS}B
      {OFF},{RVS}C{OFF} OR {RVS}D{OFF}"               :rem 166
1040 GETA$:IFA$=""THEN1040                             :rem 173
1050 IFA$<"A"ORA$>"D"THEN1040                         :rem 59
1060 IFA$<>R$THEN1200                                   :rem 157
1070 GOTO1300                                           :rem 197
1100 GOSUB6000:Q$=X$:GOSUB6000:R$=X$                   :rem 89
1102 PRINT"{CLR}{DOWN}{3 SPACES}QUESTION #";NQ
                                             :rem 174
1110 PRINT"{3 DOWN}";Q$                               :rem 119
1120 O=480:L=80:GOSUB8000                               :rem 153
1130 IFX$<>R$THEN1200                                   :rem 178
1140 GOTO1300                                           :rem 195
1200 GOSUB7200:PRINT"{CLR}{3 DOWN}CORRECT ANSWER:
      {3 DOWN}":PRINTR$                                 :rem 105
1210 FORX=1TO1500:NEXTX:PRINTCHR$(147):GOTO800
                                             :rem 107
1300 CQ=CQ+1:GOSUB7000                                   :rem 6
1310 PRINTCHR$(147)                                     :rem 64
1320 GOTO800                                           :rem 151
2000 PRINT"{CLR}{2 DOWN}{15 SPACES}{RVS}QUIZ OVER!
      {OFF}"                                             :rem 146
2010 PRINT"{2 DOWN}{3 SPACES}NUMBER OF QUESTIONS =
      ";NQ                                               :rem 246
2020 PRINT"{2 DOWN}{3 SPACES}NUMBER CORRECT
      {6 SPACES}=";CQ                                   :rem 158
2030 IFNQ=0THENNQ=1                                     :rem 156
2040 X=INT((CQ*100)/NQ)                                 :rem 5
2050 PRINT"{2 DOWN}{3 SPACES}YOUR SCORE{10 SPACES}
      =";X;"{LEFT}%"                                     :rem 150
2060 PRINT"{3 DOWN}{12 SPACES}(PRESS ANY KEY)"
                                             :rem 123
2070 GETX$:IFX$=""THEN2070                             :rem 227
2080 PRINTCHR$(147)                                     :rem 69
2090 CLOSE 9                                           :rem 122
2100 GOTO200                                           :rem 142
6000 X$=""                                             :rem 195
6010 GET#9,R$                                           :rem 165
6020 IFR$=""THEN6010                                    :rem 68
6030 IFASC(R$)=13THEN6060                              :rem 146
```

```

6040 X$=X$+R$ :rem 160
6050 GOTO6010 :rem 203
6060 RETURN :rem 172
6700 X=LEN(X$):IFX=36ORX=76THEN6720 :rem 118
6710 PRINT"{SHIFT-SPACE}" :rem 63
6720 RETURN :rem 175
7000 POKEMC+0,0:POKEMC+1,0:POKEMC+5,15:POKEMC+6,15
:POKEMC+24,10 :rem 220
7010 FORX=2TO11 :rem 123
7020 POKEVC+33,X:POKEMC+1,X*5:POKEMC+4,33 :rem 200
7030 FORY=1TO50:NEXTY :rem 82
7040 POKEMC+4,32:FORY=1TO10:NEXTY,X :rem 188
7050 POKEMC+24,0 :rem 120
7060 RETURN :rem 173
7200 POKEMC+0,0:POKEMC+1,30:POKEMC+4,33:POKEMC+5,1
5:POKEMC+6,15 :rem 228
7210 POKEVC+33,0:POKEMC+24,10 :rem 150
7220 FORX=1TO500:NEXTX :rem 129
7230 POKEMC+4,32:FORX=1TO20:NEXTX :rem 56
7240 POKEMC+24,0:POKEVC+33,11 :rem 154
7250 RETURN :rem 174
7500 PRINT"{CLR}{3 DOWN}{3 SPACES}ENTER QUIZ NAME:
":PRINT"{DOWN}{3 SPACES}(1-37 LETTERS)"
:rem 221
7510 O=323:L=38:GOSUB8000:IFLEN(X$)>37THEN7500 :rem 185
7520 QN$=X$:IFLEN(X$)>11THENQN$=LEFT$(X$,11)
:rem 56
7530 QN$="QUIZ/"+QN$ :rem 121
7540 PRINT"{6 DOWN}{3 SPACES}PREPARE ";DV$:PRINT"
{DOWN}{3 SPACES}THEN PRESS {RVS}RETURN{OFF}"
:rem 107
7550 GETES:IFES=""THEN7550 :rem 205
7560 IFASC(E$)<>13THEN7550 :rem 208
7570 RETURN :rem 179
8000 X$="" :POKEM+0,160:POKEC+0,1 :rem 59
8010 GETY$:IFY$=""THEN8010 :rem 229
8020 X=ASC(Y$):IFX=13THEN8150 :rem 195
8030 IFX=20THEN8100 :rem 73
8040 Y=LEN(X$):X$=X$+Y$:IFX>63THENX=X-64 :rem 160
8050 POKEM+O+Y,X:POKEC+O+Y,1:POKEM+O+Y+1,160:POKEC
+O+Y+1,1:IFLEN(X$)<LTHEN8010 :rem 253
8060 GOTO8150 :rem 213
8100 Y=LEN(X$):IFY<1THEN8010 :rem 146
8110 POKEM+O+Y,32:POKEC+O+Y,11:POKEM+O+Y-1,160:POK
EC+O+Y-1,1 :rem 71
8120 Y$=LEFT$(X$,Y-1):X$=Y$:GOTO8010 :rem 48
8150 Y=LEN(X$):POKEM+O+Y,32:POKEC+O+Y,11 :rem 141
8160 RETURN :rem 175
9000 IFPS=1THEN9080 :rem 105

```

## 2: Education

---

```
9005 POKEMC+0,0:POKEMC+1,0:POKEMC+5,79:POKEMC+6,12
    9:POKEMC+24,15 :rem 40
9010 FORX=0TO30 :rem 124
9020 Y=INT(NT%(X)/256) :rem 217
9030 POKEMC+0,NT%(X)-(Y*256) :rem 32
9040 POKEMC+1,Y:POKEMC+4,17 :rem 89
9050 FORY=1TO70:NEXTY :rem 88
9060 POKEMC+4,16:FORY=1TO10:NEXTY,X :rem 194
9070 POKEMC+24,0:PS=1 :rem 199
9080 RETURN :rem 177
9200 IFDP=1THEN9280 :rem 94
9205 PRINTCHR$(147);"{4 DOWN} DO YOU WANT TO USE D
    ISK OR TAPE FOR" :rem 69
9210 PRINT"{DOWN} SAVING/LOADING QUIZZES?" :rem 29
9220 PRINT"{5 DOWN} ENTER {RVS}D{OFF} OR {RVS}T
    {OFF}" :rem 242
9230 GETX$:IFX$=""THEN9230 :rem 237
9240 IFX$="D"THENDV=DI:DV$="DISK":GOTO9270 :rem 27
9250 IFX$="T"THENDV=TA:DV$="TAPE":GOTO9270 :rem 51
9260 GOTO9230 :rem 216
9270 DP=1 :rem 212
9280 RETURN :rem 179
9900 DATA6430,6430,6430,4817,8101,8101,8101,6430,6
    430,8101,9634,9634 :rem 121
9910 DATA8583,8101,7217,0,7217,8101,8583,8583,8101
    ,7217 :rem 2
9920 DATA8101,6430,6430,8101,7217,4817,6069,7217,6
    430 :rem 148
```

# Chapter 3

---

# Applications

□ □ □ □ □

□ □ □ □ □

# SpeedScript Customizer

---

J. Blake Lambert

*SpeedScript is a full-featured word processor for the VIC and 64 which appeared in COMPUTE!'s Second Book of Commodore 64 and COMPUTE!'s Third Book of VIC. Many SpeedScript users would like to change its default settings and formatting commands to suit their own preferences, and this short program shows how. It lets you modify SpeedScript with any values you choose, creating a new version that can be saved to tape or disk. For the VIC (with at least 8K expansion) and the 64.*

**I**f you use *SpeedScript* with a VIC or 64, "*SpeedScript Customizer*" may be a real timesaver. It lets you predefine background and character color; left, right, top, and bottom margins; page length; and line spacing. It also lets you select single sheet or fanfold (continuous pinfeed) paper. You can also change or add values for the predefined formatting codes used for printing. In addition, it fixes the new page command in *SpeedScript 1.0* (January 1984) and corrects an error in the predefined values of the version of *SpeedScript* printed in *COMPUTE!'s Second Book of Commodore 64*.

Using the Customizer will let you make personalized copies of *SpeedScript*. For example, you may prefer to print single-spaced documents with margins at 10 and 70, using single sheets of paper. You may also need to send special codes to your printer to access all of its features. *SpeedScript* allows you to assign formatting codes at the beginning of a document, but you have to define them every time you want to use them.

It's possible to set up format files and save them if you like. However, it is simpler to use the Customizer to save your personalized version(s) of the program. The values can be re-defined just as before; you're only changing the default values that *SpeedScript* thinks are "normal." You could, for example, have one version of *SpeedScript* for writing business letters, one for personal letters, and another for writing reports.

## How to Use *SpeedScript* Customizer

First, type in and save Programs 1 and 2. Be sure that you name Program 2 "CUST.SS". Then load and run Program 1,

### 3: Applications

---

the Customizer Boot, which automatically loads and runs Program 2. The Customizer will prompt you to insert a version of *SpeedScript* and then ask for its filename. Type in the filename of the *SpeedScript* version on your tape or disk and press RETURN. Press D (for disk) or T (for tape) at the prompt. When loading is complete, your screen will show which version was found. That message is followed by the color selection screen.

Not all monitors have perfect picture resolution. Thus it's nice to be able to select the color of the background and characters, which *SpeedScript* allows with the CTRL-B and CTRL-L commands. But if you CLEAR ALL TEXT, the program returns to the default colors (the colors that were there when you first ran the program). The color selection screen in the Customizer allows you to flip through the background colors with the f1 key and through the character colors with the f3 key.

That allows you to select any color combination you want. For instance, some people like to use a dark gray or black background with light green characters, to emulate a green screen monitor. When you find a combination that suits you, press RETURN.

### Changing Default Values

After you've set the letter and background colors, another menu appears. It asks you to enter the default values. If you choose not to change a setting, simply press RETURN and the original default will remain unaltered.

Here are a few tips on setting the values correctly.

*Left margin.* Sets the default value for *SpeedScript's* [l] function (obtained by holding down the CTRL key and pressing the £ key, then pressing l). The left margin sets the distance (number of spaces) from the left edge of the page to the point where the first character is printed. It should be at least 1. For a one-inch margin with normal (pica, ten characters per inch) type, set this value to 10. With other print sizes, multiply the margin width you want (in inches) by the number of characters per inch.

*Right margin.* Sets the default value for the [r] function. This is the preferred distance from the last character on a line to the right edge of the paper, subtracted from the number 80. You can also think of this as the left margin plus the number



of characters per line. With 8-1/2-inch-wide paper and [l] set at 10, make [r] 70 to get a one-inch right margin.

*Page length.* This value has no corresponding function in *SpeedScript*. It is the number of lines that you want to fit on a page, and it is preset at 66 (since most printers print six lines per inch and standard paper is 11 inches long). If you want to use nonstandard stationery or legal-size paper, change the value accordingly (inches of length multiplied by six).

Some printers and interfaces allow you to change the spacing between lines to print eight lines per inch. Once you've set the printer into that mode (you may have to flip a switch on the interface or send a special code to the printer), change the page-length value in *SpeedScript* to 88 (lines per inch times length of paper in inches). Remember to change the bottom margin, too.

*Top margin.* Sets the default for [t], the number of blank lines at the top of the page. Should be 5 or more.

*Bottom margin.* Sets the [b] default. This is the page length minus the number of lines you would like at the bottom of the page. You can think of this as the top margin added to the number of lines you want to print. It should be 58 or less when using standard paper; it should always be at least 8 less than the page length.

*Spacing.* *SpeedScript's* [s] function. Use a 1 for single-spacing between lines of text, a 2 for double-spacing, and so forth.

*Paper style selection.* Works like the [w] command. Answer 0 and *SpeedScript* will wait for you to press RETURN after printing each page of text. That makes it easy to use single sheets of paper. The default value, 1, signals continuous pinfeed paper, but you can still use the [w] command when you wish.

## Other Options

The user-definable reverse-video numbers can also be preset in this section of the program. The first four probably should not be redefined. If you often share files with friends, you should consider standardizing your use of predefined numbers. We'll give some tips on setting the user-definable codes in a moment.

After setting the values, the program will ask if you wish to continue or rerun. Check the values and press R if you find

any errors; that will cause the program to start over from the beginning. Otherwise, press C to continue, and enter the filename you want to use for your new customized version of *SpeedScript*. Then press RETURN.

Make sure to give the new *SpeedScript* a unique name, so you'll know which version to load later on. *SpeedScript* Customizer doesn't allow the SAVE with Replace option, so you can't destroy the original *SpeedScript* while using the Customizer. Remember that no matter what version you use, the default values can still be changed using the CTRL-£ commands in *SpeedScript*.

When the program finishes, it resets the BASIC pointers and saves your modified *SpeedScript*. If all goes well, the program will automatically run your new version. Disk users should check the error channel by pressing the up-arrow key while holding down CTRL, then pressing RETURN.

Next, look at the directory using *SpeedScript*'s CTRL-4 command. Tape users can recover from errors (for example, if RECORD was not down during the SAVE) by pressing RUN/STOP-RESTORE and typing SAVE "new filename",1 followed by RETURN.

If the program does not execute properly, remember to turn the computer off and then on again before doing other programming. That will reset the memory pointers to prevent problems and free up the memory space used by the Customizer.

### **How *SpeedScript* Customizer Works**

When you use the Customizer, you actually have two programs in memory at the same time and use one program to modify the other. This technique is described in *COMPUTE!'s Mapping the Commodore 64*.

Program 1 (line 8) determines whether the computer in use is a VIC or 64 by using the Kernal SCREEN routine. It checks the number of columns (22 for a VIC and 40 for a 64) and adjusts the start of BASIC to a point above where *SpeedScript* normally resides in memory. The boot program prints the necessary commands on the screen, then fills the keyboard buffer (a small area of memory that temporarily stores character information) with a HOME character, two RETURNS, an exclamation point, and the code for LOAD and RUN. Because of the exclamation point, the computer ignores

the LOAD command and performs the RUN. That is how it automatically loads and runs Program 2.

The Customizer again checks which computer is in use and sets the values of several variables. Line 50 of Program 2 loads *SpeedScript* into its usual place in memory. That explains the extra ,1 at the end of the LOAD command. The computer ignores *SpeedScript*, though, since it is below the current start of BASIC.

Next, it tests to see which version of *SpeedScript* is currently in memory (by PEEKing a designated memory location). It then tells you what it has found (lines 56–64). Lines 66–86 handle the default color selection, and INPUT statements allow you to change the normal values for print formatting (lines 88–122).

To make the program work with all versions of *SpeedScript*, Program 2 contains its own definition tables. Three of these tables are located in lines 128–132. Depending on what version of *SpeedScript* is in memory, one of these tables will be used to point to the location in *SpeedScript* that holds the background color (BL), letter color (LL), and the start of *SpeedScript*'s definition table (DT). Line 134 POKEs these locations with the values you have assigned in Customizer.

Should future versions of *SpeedScript* become available, the pointers in the Customizer can be changed so that it will modify the new versions. A simple machine language monitor, a BASIC PEEKing loop, or even an MLX listing would be enough to find the definition table; just look for consecutive memory locations that hold 5, 75, 66, 5, 58, 2, 1, 27, 14, 15, 18, 0, 0, 0, 0, 0 (the values that are predefined). The variable DT in the Customizer would need to be set equal to the memory location that holds the first value (5) in the list above. The locations referenced by the variables BL and LL might need to be readjusted, as well.

Line 150 of Program 2 determines which table should be used for POKeing the BASIC pointers to the right values before saving the modified version of *SpeedScript*. When a SAVE is performed in BASIC, the start address of the block of memory to be saved is contained in locations 43 and 44 (in standard low-byte/high-byte form). The top of the block to be saved is one position below the value contained in locations 45 and 46 (called the start-of-BASIC variables, stored in the same format).

Lines 160–168 print the statements to perform the POKEs and to save and run the new *SpeedScript*; they also fill the keyboard buffer with a HOME character, three RETURNS, an exclamation point, and the code for LOAD and RUN. The Customizer vanishes from sight as it is replaced by *SpeedScript*. (Actually, it's still high in memory but is unavailable for use.)

#### More on Sending Printer Codes

One of the biggest benefits of using *SpeedScript* Customizer is that it lets you incorporate specific printer codes. Most printer codes are easy to send and are listed in printer and interface manuals. Gemini Star and Epson (Grafrax) owners, for example, can send the ESCape code (CTRL-£ 1, represented in this article by [1]) followed by a 4, in the text of the *SpeedScript* file on the screen, to cause the printer to print in italics. To turn the italics print off, send [1]5. Some interfaces, including the Tymac Connection, require sending the ESCape code twice when using emulation mode. (If you have problems, refer to your printer/interface manual.)

Some printer features require three codes to be sent, however. On the Gemini Star, for example, the code that triggers continuous underlining is ESC-1. But sending this to *SpeedScript* as [1]-1 doesn't work. To send the codes properly, you need only define a reverse-video number to the value 1. Since [1] is already used by *SpeedScript*, use [8] instead. From within *SpeedScript*, this would be [8]=1 (the Customizer allows you to set default values for the reverse-video numbers, so they don't have to be defined on the screen). Then, simply insert [1]-[8] immediately before the text you wish to underline.

Now let's turn it off. The code sequence for turning off the Gemini's continuous underline feature is ESC-0. Unless it is defined otherwise, the default value of [9] in *SpeedScript* is zero. Thus, you should place the characters [1]-[9] on the screen after the word or phrase you want underlined.

Any three-character code sequence can be sent in this manner to the printer, so the Gemini's foreign character sets can be accessed by *SpeedScript*. The table lists printer codes for the Gemini; if you have another printer, refer to your manual. *SpeedScript* can even be used with letter-quality printers if you

redefine the codes to match those that the interface and printer will accept.

### **SpeedScript Formats to Access Selected Gemini and Epson Features**

(This table uses these preset values in addition to the predefined default settings: [5] = 20 [8] = 1 [9] = 0.)

- [2] enlarged (double-width) print (cleared when a carriage return character is sent)
- [3] condensed print (use [5] instead with some interfaces)\*\*
- [4] pica print
- [5] cancel enlarged print (use [3] instead with some interfaces)\*\*
- [1]4 italics on\*
- [1]5 italics off\*
- [1]E emphasized on\*
- [1]F emphasized off\*
- [1]G double-strike on\*
- [1]H double-strike off\*
- [1]O disable skip-over perforation
- [1]S[8] subscripts on
- [1]S[9] superscripts on
- [1]T sub/superscripts and unidirectional printing off
- [1]U[8] unidirectional printing on
- [1]U[9] unidirectional off
- [1]W[8] double-wide printing on (alternate method, not cleared by a carriage return character)
- [1]W[9] double-wide printing off (alternate method)
- [1]Y[8] enable buzzer
- [1]Y[9] disable buzzer
- [1]-[8] underline on
- [1]-[9] underline off

\* Indicates this command works for Epson Graftrax.

\*\* Some interfaces, notably CARDCO and XETEC, swap these two codes, CHR\$(15) and CHR\$(20).

To access foreign character sets, send [1]7[7] after setting [7] to one of the following values:

- |              |             |
|--------------|-------------|
| 0 = American | 4 = French  |
| 1 = British  | 5 = Swedish |
| 2 = German   | 6 = Italian |
| 3 = Danish   | 7 = Spanish |

### 3: Applications

---

Some printers use DIP switches to invoke foreign character sets, so they won't take these codes. In those cases, simply flip the right switches and you're through.

After you've selected the desired character set, you may obtain some of the special characters from the keyboard. Others will require the use of the user-definable reverse-video numbers in *SpeedScript*. Check your printer manual and the Commodore ASCII chart in the *Programmer's Reference Guide* and experiment. For example, with the Spanish character set activated, a closed bracket (]) on the screen would cause an inverted question mark to be printed on a Gemini printer.

Even when you're using the normal character set, symbols obtained by pressing the Commodore logo key will cause the printer's (or the interface's) characters to be printed. In that way, you can access a number of graphics and special characters (most of the printer's characters with ASCII codes 161-191) from within *SpeedScript*. Just compare the ASCII charts in the printer and computer manuals.

Other features are available by defining the reverse-video numbers. For example, to have the printer backspace one character (allowing you to print accent marks), simply define a reverse-video number as 8. Some printer/interface combinations will interpret this value as a graphics command, so consult your manual and define the number as you need it. Then, whenever *SpeedScript* finds the reverse-video number in the text, it will backspace. Similarly, to activate the printer's internal buzzer during a printout, you could define one of the reverse-video numbers as 7 and place the defined number in the text wherever you wish, even in the footer.

If you get confused about the codes, remember to check your manuals. If things don't work right, keep trying. Note, too, that some printer functions will not work while others are in effect. For example, some printers will not print superscripts in the emphasized mode, but will automatically double-strike the superscript data. If you can't get signals through the interface at first, try using *SpeedScript*'s CTRL-P command and resetting the secondary address to the interface's *transparent* (no ASCII correction) mode. In most cases, once the printer is set it will stay in that mode until you send codes to change it (or until you turn the power off).

One final note: Whenever you want to include a memo about a file, use a SHIFT-SPACE (hold down SHIFT and type a space) to separate the filename from the memo. For example, you may want to save a note about City League Baseball with the name CLB and have a note *in the directory* that says SPDSOCR (to indicate it is a *SpeedScript* file) too. In *SpeedScript* enter the filename as below:

```
SAVE:CLB{SHIFT-SPACE}SPDSOCR
```

A small dot will appear where the {SHIFT-SPACE} was entered). Assuming that it is four blocks long, the file will LIST in the directory as follows:

```
4 "CLB"SPDSOCR PRG
```

You can then load the file with the short name (CLB) or the long name (CLB{SHIFT-SPACE}SPDSOCR). This trick can also be used when saving a BASIC program. For example, you could produce the same directory entry by entering the following line in BASIC immediate mode:

```
SAVE"CLB{SHIFT-SPACE}SPDSOCR",8
```

### Program 1. *SpeedScript* Customizer, Boot Program

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
2 DN=8:PRINT"{CLR} BOOT: {RVS}D{OFF}ISK OR {RVS}T
  {OFF}APE" :rem 131
4 GETZ$:IFZ$="T"THENDN=1:GOTO8 :rem 136
6 IFZ$<>"D"THEN4 :rem 168
8 SYS65517:IFPEEK(781)=22THENPRINT"{CLR}POKE44,39:
  POKE256*39,0:NEW":GOTO12 :rem 115
10 IFPEEK(781)=40THENPRINT"{CLR}POKE44,48:POKE256*
  48,0:NEW :rem 99
12 PRINT"{2 DOWN}LOAD"CHR$(34)"CUST.SS"CHR$(34)", "
  DN :rem 65
14 FORI=631TO635:READN:POKEI,N:NEXT:POKE198,5
  :rem 98
16 DATA19,13,13,33,131 :rem 94
```

### Program 2. *SpeedScript* Customizer, Main Program

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 IFFL=1THEN52 :rem 86
10 GOTO16 :rem 1
12 PRINT"{CLR}"X$"{RVS}SPEEDSCRIPT_CUSTOMIZER{OFF}
  " :rem 6
14 RETURN :rem 69
```

### 3: Applications

---

```
16 LC=0:BC=1:C$="VIC":P0=129:P1=4674:P2=132:P3=145
   :SYS65517                                     :rem 216
18 IFPEEK(781)=40THENC$="C64":X=1:P1=2062:P2=103:P
   3=106                                         :rem 211
20 DIMV(20):PRINTCHR$(14):PRINTCHR$(8):IFXTHENX$="
   {9 SPACES}"                                  :rem 97
22 GOSUB12                                       :rem 71
24 PRINTX$"{2 DOWN}CUSTOMIZER WILL NOT"        :rem 147
26 PRINTX$"DESTROY THE SOURCE "                :rem 145
28 PRINTX$"COPY OF SPEEDSCRIPT."              :rem 251
30 PRINTX$"{DOWN}INSERT SOURCE DISK{2 SPACES}"
                                               :rem 18
32 PRINTX$"OR TAPE VERSION OF                  :rem 22
34 PRINTX$"SPEEDSCRIPT 1.0 OR"                :rem 42
36 PRINTX$"2.0 FOR {RVS}"C$"{OFF}."          :rem 136
38 PRINTX$"{DOWN}ENTER SOURCE FILENAME."      :rem 7
40 PRINTX$;:INPUT"NAME";NF$                   :rem 74
42 PRINTX$"{DOWN}{RVS}D{OFF}ISK OR {RVS}T{OFF}APE?
   "                                             :rem 65
44 GETZ$:IFZ$<>"D"ANDZ$<>"T"THEN44           :rem 9
46 IFZ$="T"THEND$="{RVS}TAPE{OFF}":DN=1       :rem 108
48 IFZ$="D"THEND$="{RVS}DISK{OFF}":DN=8      :rem 102
50 FL=1:LOADNF$,DN,1                           :rem 146
52 IFXTHENLC=11:BC=12:POKE53280,BC:POKE53281,BC
                                               :rem 67
54 POKE646,LC:IFX=0THENPOKE36879,25           :rem 131
56 PE=PEEK(P1)                                  :rem 52
57 IFPE=P2ORPE=P0THENV$="V1":V=1:IFXANDPEEK(3585)=2
   7THENPOKE5755,133                           :rem 205
58 IFPE=P3THENV$="V2":V=2                     :rem 2
60 IFPE<>P0ANDPE<>P2ANDPE<>P3THENPRINT"READ ERROR.
   ":FORI=1TO2000:NEXT:RUN                     :rem 134
62 IFP1=4674THENV=3                            :rem 127
64 PRINTX$"{DOWN}{RVS}"C$" SPEEDSCRIPT {RVS}"V$:FO
   RI=1TO2000:NEXTI:PRINT"{CLR}"              :rem 244
66 PRINT"HOME"X$"{RVS}SPEEDSCRIPT CUSTOMIZER
   {OFF}"                                       :rem 143
68 PRINTX$"{DOWN}{RVS}F1{OFF} CHANGES BACKGROUND"
                                               :rem 192
70 PRINTX$"{DOWN}{RVS}F3{OFF} CHANGES LETTERS"
                                               :rem 254
72 PRINTX$"{DOWN}{RVS}RETURN{OFF} SETS THE COLORS
                                               :rem 27
74 PRINTX$"AS DEFAULT COLORS."                 :rem 47
76 GETZ$:IFZ$=CHR$(133)THENBC=BC+1AND15:IFX=1THENP
   OKE53281,BC:POKE53280,BC                     :rem 0
78 IFZ$=CHR$(133)ANDX=0THENBP=(BCAND15)*16+(BCAND7
   )+8:POKE36879,BP                             :rem 126
80 IFZ$=CHR$(133)THENFORI=0TO200:NEXT:GOTO76
                                               :rem 245
```



```

82 IFZ$=CHR$(134)THENLC=LC+1AND15:IFX=0THENLC=LCAN
   D7 :rem 223
84 IFZ$=CHR$(134)THENPOKE646,LC:GOTO66 :rem 56
86 IFZ$<>CHR$(13)THEN76 :rem 71
88 GOSUB12 :rem 83
90 PRINTX$"ORIGINAL DEFAULT{3 SPACES}" :rem 144
92 PRINTX$"SETTINGS ARE LISTED " :rem 198
94 PRINTX$"BELOW:" :rem 109
96 PRINTX$;:INPUT"LEFT MARGIN{7 SPACES}5{3 LEFT}";
   V(0) :rem 200
98 PRINTX$;:INPUT"RIGHT MARGIN{6 SPACES}75{4 LEFT}
   ";V(1) :rem 242
100 PRINTX$;:INPUT"PAGE LENGTH{7 SPACES}66{4 LEFT}
   ";V(2) :rem 182
102 PRINTX$;:INPUT"TOP MARGIN{8 SPACES}5{3 LEFT}";
   V(3) :rem 183
104 PRINTX$;:INPUT"BOTTOM MARGIN{5 SPACES}58
   {4 LEFT}";V(4) :rem 113
106 PRINTX$;:INPUT"SPACING{11 SPACES}2{3 LEFT}";V(
   5) :rem 14
108 PRINTX$;:INPUT"FANFOLD(N=0/Y=1){2 SPACES}1
   {3 LEFT}";V(6) :rem 7
110 FORI=1TO9:READJ$:K=I+6 :rem 60
112 PRINTX$"[CTRL] £ "I" ={4 SPACES}"J$;:INPUT"
   {4 LEFT}";V(K) :rem 76
114 NEXTI :rem 30
116 DATA27,14,15,18,00,00,00,00,00 :rem 143
118 PRINTX$"{RVS}C{OFF}ONTINUE OR {RVS}R{OFF}ERUN.
   " :rem 239
120 GETZ$:IFZ$="R"THENRUN :rem 47
122 IFZ$<>"C"THEN120 :rem 101
124 GOSUB12 :rem 122
126 PRINTX$;:INPUT"{DOWN}NEW FILENAME";NF$:rem 154
128 IFV=1THENBL=2408:LL=2417:DT=5200 :rem 105
130 IFV=2THENBL=2411:LL=2425:DT=5275 :rem 104
132 IFV=3THENBL=4979:LL=5031:DT=7750 :rem 124
134 POKEBL,BC:POKELL,LC:FORI=0TO15:POKEDT+I,V(I):N
   EXTI :rem 245
136 IFDN=1THENPRINTX$"{DOWN}{RVS}PRESS STOP ON TAP
   E{OFF}" :rem 116
138 PRINTX$"{DOWN}INSERT DESTINATION{2 SPACES}"
   :rem 145
140 PRINTX$D$" TO HOLD" :rem 20
142 PRINTX$"MODIFIED SPEEDSCRIPT" :rem 107
144 PRINTX$"AND PRESS {RVS}RETURN{OFF}." :rem 248
146 GETZ$:IFZ$<>CHR$(13)ANDZ$<>CHR$(141)THEN146
   :rem 237
148 IFX=0ANDV$="V2"THENV=4 :rem 61
150 ONVGOSUB152,154,156,158:GOTO160 :rem 3
152 HS=8:LE=162:HE=27:RETURN :rem 208

```

### 3: Applications

---

```
154 HS=8:LE=0:HE=40:RETURN           :rem 100
156 HS=18:LE=108:HE=37:RETURN        :rem 6
158 HS=18:LE=8:HE=38:RETURN          :rem 168
160 PRINT "{CLR}PO43,1:PO44,"HS"     :rem 136
162 PRINT "{2 DOWN}PO45,"LE":PO46,"HE" :rem 179
164 PRINT "{2 DOWN}SAVE"CHR$(34)NF$CHR$(34)","DN :rem 233
166 DATA19,13,13,13,33,131         :rem 36
168 POKE198,6:FORI=631TO636:READN:POKEI,N:NEXT :rem 158
```

# Memo Writer

---

Mark R. Brown

*Here's a mini word processor that's handy for memos, notes, or lists. Versions are included for the unexpanded VIC and for the 64.*

“**M**emo Writer” is a simple text-processing program written in BASIC. It allows you to edit text using the INST/DEL, CLR/HOME, and cursor control keys. Once you have filled the screen with text, you may send it to disk or to your printer. Since Memo Writer is written in BASIC, you can easily customize it to suit yourself.

Memo Writer is screen-oriented. This means that you work with only one screen of text at a time. The program will prevent you from doing anything which would cause the screen to scroll while you are typing, since that would cause you to lose some of your work. In addition to the normal editing keys and function keys, the 64 version makes use of CTRL key combinations to give you additional control over the text.

## VIC Memo

Since you are limited to one screen of text, the VIC version of the program prevents you from doing almost anything that would cause scrolling. However, there is one exception: If you use the INST key to insert characters on the bottom line, the screen *will* scroll, so avoid this if possible.

The function keys are used for tabs and for selecting print options. You can choose single- or double-spacing and expanded or normal print sizes. There are no set margins, but the tabs can be used to move the left margin.

When creating your memos, you can type in either uppercase only (the default mode) or in upper- and lowercase (by pressing the Commodore key and the SHIFT key). The PRINT subroutine PEEKs to see which shift mode you're in and sends the proper control characters to the printer.

You can save about half the work of typing in the program if you leave out line 9 and lines 500–780 and refer to the program listing for instructions. Line 500 sets the background and border colors; they can be changed to match your preference.

### 3: Applications

---

The start-of-line markers help you keep track of where you are on an 80-column line. Don't forget to erase them before you print or they'll appear in your printed output. You can eliminate or modify them in line 800.

A side effect of having repeating keys is a possible inconsistency when selecting (toggling) between uppercase and lowercase. That may be an aggravation. To turn off this function, delete POKE 650,128 in line 10.

There are a couple of tricks in the input routine. POKE 204,0 in line 10 turns the cursor on. Normally you wouldn't have one during a GET and PRINT sequence. POKE 205,3 in line 40 sets the cursor-blink countdown timer to a short count, to even out the timing jerks caused by the GET loop in line 20. Without this, typing is not smooth at all. WAIT 207,1 in line 40 waits for the cursor to blink off before printing and keeps the PRINT statement from leaving reverse characters during the cursor-blink phase.

The PEEKs in lines 35 and 50 check to see if you are on the last screen line, and keep you from doing anything which would cause the screen to scroll. It should be fairly easy to add any special features you want. This program supports the full graphics character set, but of course it will print properly only on a Commodore-compatible printer. Those with other printers may need to make some changes in the control codes in order to make Memo Writer compatible.

You will be unable to use quotes on the VIC version of Memo Writer. To do so could cause problems if you tried to use the cursor keys within the quotes.

## 64 Memo

In addition to the standard Commodore 64 screen-editing functions, the following commands are available on the 64 version of this program:

- f1 Tab 5 spaces.
- f2 Send text to the printer using the format you've specified. Unless you specify otherwise before pressing f2, it will assume you want normal, single-spaced output.
- f3 Print single-spaced text.
- f4 Print double-spaced text.
- f5 Turn off key repeat.
- f6 Turn on key repeat.
- f7 Print normal-sized text.
- f8 Print expanded text.

For the following commands you must hold down the CTRL key and then press the next key indicated.

**CTRL D.** Remove the caret (>) marks from the display. Whenever you clear the screen, a line of carets appears on the left-hand side. This helps you to keep track of where the 80-column lines start on the Commodore 64's 40-line display. If you don't want them there, CTRL D will get rid of them. Any PRINT command (f2) will automatically eliminate them.

**CTRL G.** GET a screen of text from disk. Whenever you use GET or PUT, the status line will disappear from the bottom of the screen and you will be asked to input a filename (15 characters or less). After you have entered the filename, the status line will be restored.

**CTRL P.** PUT the screen of text currently displayed to disk. You will be asked for a filename.

**CTRL C.** COPY this screen of text to the buffer in memory. There is a buffer in memory which can hold one screen's worth of text. The COPY command will copy what is on the screen into the memory buffer, overwriting whatever was in the buffer to start with. You can also copy a screen to the buffer if you want to save it while you GET and look at a screen from disk.

**CTRL X.** SWAP the displayed screen of text with the text screen in memory, so you can work on two screens at once. The border color will change from blue to red to remind you which screen you currently have displayed.

**CTRL Y.** YANK the text screen from memory onto the screen. YANK copies the buffer memory to the screen without erasing the buffer's contents.

*Other features.* The 64 version of Memo Writer will not go into quote mode or insert mode, so you can use quotation marks or the insert key without affecting the cursor keys. In addition, since the status line is at the bottom of the screen, you don't have to worry about the odd system lock-up bug that is sometimes encountered when trying to insert or delete characters on the last screen line. However, deleting text on the last available screen line can disturb the status line display.

Screens of text are saved to disk as screen code program files, which makes them disk compatible with *SpeedScript* files. This means you can read short *SpeedScript* files with Memo Writer (and short Memo Writer files with *SpeedScript*). This

will work if you indent two spaces at the beginning of the text. The two programs are not, however, printer-output compatible. You can't just load files from one into the other and send them to the printer without substantial modification.

Unlike *SpeedScript*, Memo Writer has no built-in DOS commands. However, it is compatible with the DOS wedge. Load and run the DOS wedge program first, and then load and run Memo Writer. When you need to perform a DOS-related operation, just use the RUN/STOP key to stop the program, then use the wedge. Run Memo Writer when you are through with disk operations. If you are in the middle of editing a screen of text when you need to use the wedge, remember to save a copy of the screen in progress so you can load it back in and pick up where you left off.

The files you create with Memo Writer are only 1K long, but they can be concatenated (joined together) to make longer files for use by *SpeedScript* or terminal programs by using the DOS COPY command. Consult your 1541 manual for the specifics on concatenating files.

*The status line.* At the bottom of the screen is the status line, which keeps you informed of what is going on. The message in the middle identifies disk, print, and buffer memory operations. Since it is often difficult to know if anything is actually going on when one of these operations is in progress, there is a flashing arrow after the message to reassure you that the program has not crashed. The symbols on the right side of the status line tell you whether you have selected single- or double-spacing (1 or 2), normal or expanded print (N or E), and if the key repeat is on or off (+ or -). When you select a disk GET or PUT operation, the status line temporarily disappears while you input a filename.

#### Program 1. Memo Writer, VIC Version

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
9 GOSUB500 / :rem 78
10 GOSUB800:POKE650,128:SP=1:POKE204,0 :rem 234
20 GETA$:IFA$=""THEN20LIST10 :rem 132
21 IFA$=CHR$(20)ANDPEEK(210)=31ANDPEEK(209)>205THE
    NA$=CHR$(157)+CHR$(32)+CHR$(157) :rem 58
22 IFA$=CHR$(34)THEN20 :rem 223
```

```

23 IFAŞ=CHR$(13)ANDPEEK(210)=31ANDPEEK(209)>205THE
    N20                                     :rem 19
25 IFAŞ="{CLR}"THENPOKE204,1:GOSUB800:POKE204,0:AŞ
    ="":GOTO20                               :rem 98
30 IFASC(AŞ)<141ANDASC(AŞ)>132THENGOSUB100:rem 197
32 IFAŞ=""THEN20                             :rem 107
35 IFASC(AŞ)=13AND(PEEK(210)=31)AND(PEEK(209)>226)
    THEN20                                   :rem 145
40 POKE205,3:WAIT207,1:PRINTAŞ;             :rem 85
50 IFPEEK(210)=31AND(PEEK(209)+PEEK(211)>227)THENP
    OKE205,3:WAIT207,1:PRINT"{UP}";        :rem 113
60 GOTO20                                     :rem 1
100 X=0                                       :rem 86
110 IFAŞ="{F1}"THENX=5                       :rem 133
120 IFAŞ="{F3}"THENX=10                     :rem 179
130 IFAŞ="{F5}"THENX=15                     :rem 186
140 IFAŞ="{F7}"THENX=20                     :rem 184
150 IFAŞ="{F2}"THENX=25                     :rem 191
160 IFX>0THENAŞ="{FORQ=1TOX:AŞ=AŞ+"{RIGHT}":NEXTQ
    :RETURN                                   :rem 141
170 IFAŞ="{F4}"THENIFSP=1THENSP=2:POKE8164,178:RET
    URN                                       :rem 201
175 IFAŞ="{F4}"THENSP=1:POKE8164,177:RETURN
                                           :rem 253
180 IFAŞ="{F6}"THENG1=7640:TF=14:RW=11:CL=40:REM E
    XPANDED                                   :rem 23
190 IFAŞ="{F8}"THENG1=7600:TF=15:RW=5:CL=80:REM NO
    RMAL                                       :rem 109
200 AŞ=""                                     :rem 120
210 GOSUB600000                              :rem 9
220 RETURN                                   :rem 116
500 POKE 36879,9                             :rem 58
510 PRINT"{CLR}{RVS}{WHT}{4 DOWN}{5 RIGHT}MEMO WRI
    TER"                                       :rem 241
530 PRINT"{2 DOWN}{2 RIGHT}THIS IS A SCREEN-"
                                           :rem 199
540 PRINT"{4 RIGHT}ORIENTED WORD"           :rem 116
545 PRINT"{2 RIGHT}PROCESSING PROGRAM"      :rem 190
550 PRINT"{2 RIGHT}USING THE VIC-20'S"      :rem 247
560 PRINT"{RIGHT}OWN BUILT-IN EDITING"      :rem 197
570 PRINT"{5 RIGHT}CAPABILITIES."           :rem 150
590 PRINT"{4 DOWN}{5 RIGHT}{RVS}HIT ANY KEY.{OFF}"
                                           :rem 204
600 POKE198,0                                 :rem 195
610 GETAŞ:IFAŞ=""THEN610                     :rem 81
620 PRINT"{CLR}{4 DOWN}{2 RIGHT}F1,F3,F5,F7,F2-TAB
    "                                         :rem 142
630 PRINT"IN INCREMENTS OF FIVE"           :rem 184
645 PRINT"{DOWN}{4 RIGHT}F4-SET SINGLE"     :rem 74

```

### 3: Applications

---

```
647 PRINT"{2 RIGHT}OR DOUBLE SPACING" :rem 13
650 PRINT"{DOWN}{2 RIGHT}F6-PRINT EXPANDED":rem 54
660 PRINT"{5 RIGHT}CHARACTERS" :rem 222
670 PRINT"{DOWN}{3 RIGHT}F8-PRINTS NORMAL" :rem 42
682 PRINT"{4 DOWN}{5 RIGHT}{RVS}HIT ANY KEY"
:rem 14
683 POKE198,0 :rem 206
685 GETA$:IFA$=""THEN685 :rem 105
690 PRINT"{CLR}{3 DOWN}{3 RIGHT}ALL EDITING KEYS"
:rem 166
695 PRINT"{4 RIGHT}WORK AS NORMAL." :rem 183
700 PRINT"{DOWN}{2 RIGHT}TEXT CANNOT SCROLL"
:rem 138
705 PRINT"{3 SPACES}PAST THE END OF" :rem 242
707 PRINT"{5 SPACES}THE SCREEN." :rem 62
710 PRINT"{DOWN}YOU MAY USE CURSOR-UP" :rem 251
715 PRINT"OR CURSOR-DOWN MODE."; :rem 224
720 PRINT"{DOWN}{4 RIGHT}THE PRINT ROUTINE"
:rem 131
730 PRINT"{2 RIGHT}WILL AUTOMATICALLY" :rem 182
740 PRINT"{2 RIGHT}SET THE PRINT MODE" :rem 37
745 PRINT"{6 RIGHT}CORRECTLY." :rem 4
750 PRINT"{2 DOWN} {RVS}HIT ANY KEY TO BEGIN{OFF}"
:rem 241
770 GETA$:IFA$=""THEN770 :rem 95
780 POKE198,0:RETURN :rem 230
800 PRINT"{CLR}{RVS}>{OFF}";:FORI=1TO5:PRINTSPC(79
)">";:NEXTI :rem 182
805 PRINT:PRINT:PRINT:PRINT"{18 SPACES}END ";
:rem 213
810 PRINT" [4 Y]{RVS}MEMO WRITER{OFF}[5 Y]{HOME}";
:rem 214
820 RETURN :rem 122
60000 REM :rem 218
60004 G1$=CHR$(145) :rem 191
60010 G1$=G1$+CHR$(TF) :rem 131
60020 OPEN4,4:WAIT207,1:POKE204,255 :rem 234
60030 FORG0=0TORW:G0$=G1$:G1=G1+CL :rem 223
60040 FORG2=G1TOG1+(CL-1):G3=PEEK(G2) :rem 115
60050 IFG3>128THENG3=G3-128:G4=1:G0$=G0$+CHR$(18)
:rem 187
60060 IF(G3>0)*(G3<32)THENG3=G3+64:GOTO60100
:rem 185
60070 IF(G3>31)*(G3<64)THEN60100 :rem 186
60080 IF(G3>63)*(G3<96)THENG3=G3+128:GOTO60100
:rem 47
60090 IF(G3>95)*(G3<128)THENG3=G3+64:GOTO60100
:rem 48
60100 G0$=G0$+CHR$(G3) :rem 97
```



```

60110 IFG4=1THENG0$=G0$+CHR$(146):G4=0      :rem 76
60120 NEXTG2:PRINT#4,G0$:IFSP=2THENPRINT#4:rem 132
60130 NEXTG0:CLOSE4:POKE204,0              :rem 239
60140 RETURN                                :rem 219

```

## Program 2. Memo Writer, 64 Version

You will need to abbreviate a keyword in order to make line 800 fit. For example, you can type ? instead of PRINT.

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

10 GOSUB500                                :rem 118
19 REM::::::::::INITIALIZATION           :rem 128
20 POKE204,0:POKE198,0:POKE56,155:CLR:SC=1024:BF=3
   9936:SN=1                                :rem 202
25 POKE648,156:GOSUB800:POKE648,4        :rem 189
30 GOSUB800                                :rem 123
39 REM::::::::::MAIN LOOP                 :rem 185
40 GETA$:IFA$=""THEN40                     :rem 235
45 IFA$=CHR$(16)THENGOSUB450:GOTO40      :rem 114
47 IFA$=CHR$(7)THENGOSUB400:GOTO40      :rem 63
50 IFA$="{CLR}"THEN30                      :rem 255
52 IFA$=CHR$(4)THENGOSUB200:GOTO40       :rem 54
55 IFA$=CHR$(24)THENGOSUB900:GOTO40     :rem 114
57 IFA$=CHR$(3)THENGOSUB850:GOTO40     :rem 69
60 IFA$=CHR$(25)THENGOSUB950:GOTO40     :rem 116
65 IFASC(A$)<141THENIFASC(A$)>132THENGOSUB100:IFA$
   =""THEN40                                :rem 250
70 POKE205,3:WAIT207,1:PRINTA$;          :rem 88
80 IFPEEK(214)=24THENA$="{UP}":GOTO70    :rem 39
90 POKE216,0:POKE212,0:POKE213,79:GOTO40 :rem 89
99 REM::::::::::FUNCTION KEYS             :rem 2
100 ON(ASC(A$)-132)GOTO110,120,130,140,150,160,170
   ,180:A$="":RETURN                          :rem 34
110 A$="{5 RIGHT}":RETURN                 :rem 35
120 SP=1:POKE2021,177:RETURN              :rem 183
130 POKE650,0:POKE2022,173:RETURN        :rem 202
140 TF=15:RW=11:CL=80:POKE2023,142:RETURN :rem 207
150 GOTO230                                :rem 100
160 SP=2:POKE2021,178:RETURN              :rem 189
170 POKE650,128:POKE2022,171:RETURN      :rem 55
180 TF=14:RW=23:CL=40:POKE2023,133:RETURN :rem 209
199 REM::::::::::CLEAR > FROM DISPLAY    :rem 70
200 FORZ=1024TO1904STEP80                 :rem 236
210 IF(PEEK(Z)AND127)=62THENPOKEZ,32     :rem 158
220 NEXTZ:RETURN                           :rem 71
229 REM::::::::::DUMP TO PRINTER         :rem 78
230 A$="":B$="{RVS} PRINTING ↑{LEFT}":GOSUB280
                                           :rem 104

```

### 3: Applications

---

```
240 GOSUB200:GOSUB60000 :rem 88
249 REM::::::::::RESTORE MESSAGE :rem 124
250 B$="{RVS}MEMO WRITER{HOME}{OFF}":GOSUB280 :rem 148
260 RETURN :rem 120
279 REM::::::::::PRINT MESSAGE :rem 232
280 POKE205,3:WAIT207,1:POKE214,23:PRINT:POKE211,1
4:PRINTB$;:RETURN :rem 140
299 REM::::::::::PRINT CENTERED :rem 47
300 PRINTTAB(20-LEN(B$)/2);B$:RETURN :rem 137
309 REM::::::::::"HIT KEY" ROUTINE :rem 136
310 POKE214,22:PRINT :rem 177
320 B$="{2 SPACES}{RVS} HIT ANY KEY TO CONTINUE
{OFF}":GOSUB300 :rem 43
330 POKE198,0 :rem 195
340 GETA$:IFA$=""THEN340 :rem 81
350 RETURN :rem 120
399 REM::::::::::INPUT DISK FILE :rem 52
400 GOSUB1000:OPEN1,8,8,FI$ :rem 25
405 B$="{RVS} FROM DISK↑{LEFT}":GOSUB280 :rem 64
410 FORZ=0TO959 :rem 141
420 GET#1,A$:IFA$=""THEN420 :rem 207
425 IFST>0THENIFST<>64THEN495 :rem 89
430 POKE5C+Z,ASC(A$):NEXTZ :rem 109
440 CLOSE1:B$="{RVS}MEMO WRITER{HOME}{OFF}":GOSUB2
80:RETURN :rem 144
449 REM::::::::::OUTPUT DISK FILE :rem 145
450 GOSUB1000:OPEN1,8,8,"@0:"+FI$+",P,W" :rem 165
455 B$="{RVS}{2 SPACES}TO DISK ↑{LEFT}":GOSUB280 :rem 180
460 FORZ=0TO959 :rem 146
470 PRINT#1,CHR$(PEEK(SC+Z)); :rem 198
475 IFST>0THEN495 :rem 21
480 NEXTZ :rem 53
490 CLOSE1:B$="{RVS}MEMO WRITER{HOME}{OFF}":GOSUB2
80:RETURN :rem 149
494 REM::::::::::DISK ERROR :rem 10
495 B$="{RVS}{RED}{3 SPACES}ERROR{3 SPACES}{WHT}":
GOSUB280 :rem 154
496 FORZ=1TO100:POKE54296,15:POKE54296,0:NEXTZ:GOT
O490 :rem 61
499 REM::::::::::INSTRUCTIONS :rem 15
500 POKE53281,0:POKE53280,2:PRINTCHR$(142):rem 156
505 PRINT"{CLR}{WHT}{2 DOWN}";:FORZ=1TO7:READB$:GO
SUB300:NEXTZ :rem 255
510 DATA "{3 SPACES}{RVS}{10 SPACES}MEMO WRITER
{9 SPACES}{DOWN}" :rem 34
520 DATA " BY MARK R. BROWN{DOWN}" :rem 212
530 DATA "THIS IS A SCREEN-ORIENTED" :rem 82
540 DATA "WORD PROCESSING PROGRAM" :rem 72
```

```
550 DATA "USING THE COMMODORE-64'S" :rem 21
560 DATA "OWN BUILT-IN EDITING" :rem 53
570 DATA "CAPABILITIES." :rem 146
590 GOSUB310 :rem 178
600 PRINT"{CLR}";:B$="{RVS} MEMO WRITER CONTROL KE
YS {OFF}":GOSUB300 :rem 175
605 PRINT"{DOWN}{2 RIGHT}{RVS}KEY{OFF} {RVS}SYMBOL
{OFF} {RVS}FUNCTION{OFF}" :rem 200
610 PRINT"{3 RIGHT}F1{8 SPACES}TAB FIVE SPACES"
:rem 246
615 PRINT"{3 RIGHT}F2{8 SPACES}DUMP TO PRINTER"
:rem 57
620 PRINT"{3 RIGHT}F3{3 SPACES}{RVS}1{OFF}
{4 SPACES}SINGLE SPACE TO PRINTER" :rem 3
630 PRINT"{3 RIGHT}F4{3 SPACES}{RVS}2{OFF}
{4 SPACES}DOUBLE SPACE TO PRINTER" :rem 255
640 PRINT"{3 RIGHT}F5{3 SPACES}{RVS}-{OFF}
{4 SPACES}KEY REPEAT OFF" :rem 147
650 PRINT"{3 RIGHT}F6{3 SPACES}{RVS}+{OFF}
{4 SPACES}KEY REPEAT ON" :rem 85
660 PRINT"{3 RIGHT}F7{3 SPACES}{RVS}N{OFF}
{4 SPACES}NORMAL PRINT TO PRINTER" :rem 80
670 PRINT"{3 RIGHT}F8{3 SPACES}{RVS}E{OFF}
{4 SPACES}EXPANDED PRINT TO PRINTER" :rem 201
673 PRINT"CTRL-D{7 SPACES}CLEAR > MARKS FROM DISPL
AY" :rem 132
675 PRINT"CTRL-P{7 SPACES}PUT SCREEN TO DISK"
:rem 172
680 PRINT"CTRL-G{7 SPACES}GET SCREEN FROM DISK"
:rem 23
685 PRINT"CTRL-C{7 SPACES}COPY SCREEN TO MEMORY"
:rem 144
690 PRINT"CTRL-X{7 SPACES}SWAP SCREEN WITH MEMORY"
:rem 58
695 PRINT"CTRL-Y{7 SPACES}YANK SCREEN FROM MEMORY"
:rem 48
697 PRINT"{3 RIGHT}{4 SPACES}{RED}{RVS}{3 SPACES}
{OFF}{WHT}{3 SPACES}SCREEN ONE DISPLAYED"
:rem 212
699 PRINT"{3 RIGHT}{4 SPACES}{BLU}{RVS}{3 SPACES}
{OFF}{WHT}{3 SPACES}SCREEN TWO DISPLAYED{DOWN}
" :rem 2
700 FORZ=1TO3:READB$:GOSUB300:NEXT :rem 158
710 DATA "TEXT CANNOT SCROLL PAST END OF SCREEN."
:rem 95
720 DATA "YOU MAY TYPE IN UPPERCASE OR" :rem 253
730 DATA "LOWERCASE MODE." :rem 240
740 GOSUB310 :rem 175
750 RETURN :rem 124
```

### 3: Applications

---

```
799 REM::::::::::CLEAR SCREEN           :rem 132
800 POKE205,3:WAIT207,1:POKE646,1:PRINTCHR$(14):PO
    KE53281,1:PRINT"{CLR}";:POKE53281,0     :rem 112
805 PRINT"{OFF}>";:FORX=1TO11:PRINT"{ 2 DOWN}{LEFT}
    >";:NEXTX:PRINT                          :rem 64
810 PRINT"[DOWN][14 Y][RVS]MEMO WRITER{OFF}[14 Y]
    {HOME}";                                :rem 124
820 SP=1:POKE2021,177:POKE650,0:POKE2022,173
                                           :rem 250
830 G1=944:TF=15:RW=11:CL=80:POKE2023,142  :rem 75
840 POKE56295,1:RETURN                      :rem 77
849 REM::::::::::COPY SCREEN TO MEMORY   :rem 208
850 B$="{RVS}{2 SPACES}COPYING ↑{LEFT}{OFF}":GOSUB
    280                                       :rem 144
860 FORZ=0TO959                             :rem 150
870 POKEBF+Z,PEEK(SC+Z)                     :rem 152
880 NEXT                                     :rem 223
890 GOTO995                                  :rem 129
899 REM::::::::::SWAP SCREEN AND MEMORY   :rem 5
900 B$="{RVS} SWAPPING ↑{LEFT}{OFF}":GOSUB280
                                           :rem 220
910 FORZ=0TO959                             :rem 146
920 Q=PEEK(BF+Z):POKEBF+Z,PEEK(SC+Z):POKESC+Z,Q
                                           :rem 224
930 NEXT                                     :rem 219
940 GOTO990                                  :rem 120
949 REM::::::::::YANK MEMORY TO SCREEN    :rem 201
950 B$="{RVS} YANK MEM ↑{OFF}{LEFT}":GOSUB280
                                           :rem 138
960 FORZ=0TO959                             :rem 151
970 POKESC+Z,PEEK(BF+Z)                     :rem 153
980 NEXT:GOTO995                             :rem 250
990 IFSN=2THENSN=1:POKE53280,2:GOTO995     :rem 98
993 SN=2:POKE53280,6                       :rem 130
995 B$="{RVS}MEMO WRITER{HOME}{OFF}":GOSUB280:RETU
    RN                                       :rem 190
999 REM::::::::::GET DISK FILENAME       :rem 171
1000 TP$="":FI$="":POKE205,3:WAIT207,1     :rem 60
1010 FORZ=1983TO2023                        :rem 127
1020 TP$=TP$+CHR$(PEEK(Z)):POKEZ,32        :rem 49
1030 NEXTZ                                  :rem 93
1040 POKE205,3:WAIT207,1:POKE214,23:PRINT:PRINT"
    {RVS} FILENAME: {OFF}";                :rem 12
1050 GETA$:IFA$=""THEN1050                  :rem 175
1060 IFA$=CHR$(13)THEN1090                  :rem 167
1070 IFASC(A$)>31THENIFASC(A$)<129THENPRINTA$;:FI$
    =FI$+A$                                :rem 114
1075 IFASC(A$)=20THENA$="" :Q=LEN(FI$):IFQTHENFI$=L
    EFT$(FI$,Q-1):PRINTCHR$(20);          :rem 101
1085 GOTO1050                               :rem 205
```

```
1090 FI$=LEFT$(FI$,15) :rem 159
1100 WAIT207,1:FORZ=1TO41 :rem 222
1110 POKE1982+Z,ASC(MID$(TP$,Z,1)) :rem 153
1120 NEXTZ :rem 93
1130 RETURN :rem 165
59999 REM:::::SCREEN DUMP TO PRINTER :rem 22
60000 Q=PEEK(53272):IFQ=21THENG1$=CHR$(145):GOTO600010 :rem 21
60005 G1$=CHR$(17) :rem 142
60010 G1$=G1$+CHR$(TF):IFTF=15THENG1=944:GOTO60020 :rem 121
60015 G1=984 :rem 86
60020 OPEN4,4 :rem 190
60030 FORG0=0TORW:G0$=G1$:G1=G1+CL :rem 223
60040 FORG2=G1TOG1+(CL-1):G3=PEEK(G2) :rem 115
60050 IFG3>128THENG3=G3-128:G4=1:G0$=G0$+CHR$(18) :rem 187
60060 IF(G3>0)*(G3<32)THENG3=G3+64:GOTO60100 :rem 185
60070 IF(G3>31)*(G3<64)THEN60100 :rem 186
60080 IF(G3>63)*(G3<96)THENG3=G3+128:GOTO60100 :rem 47
60090 IF(G3>95)*(G3<128)THENG3=G3+64:GOTO60100 :rem 48
60100 G0$=G0$+CHR$(G3) :rem 97
60110 IFG4=1THENG0$=G0$+CHR$(146):G4=0 :rem 76
60120 NEXTG2:PRINT#4,G0$:IFSP=2THENPRINT#4:rem 132
60130 NEXTG0:CLOSE4 :rem 148
60140 RETURN :rem 219
```

# Making Calendars

Paul C. Liu

*Put your printer to good use by making a full set of calendars. These four programs will give you a screen calendar, a wall calendar, and an appointment calendar, as well as one that shows the year at a glance. For the VIC-20 and Commodore 64. On the VIC, 8K expansion may be required.*

**O**ne practical use for a computer with a printer is making your own calendars. Here are four calendar-making programs written for the VIC or 64, three of which require the use of a printer. Since the programs are written entirely in BASIC without PEEKs or POKEs, they can be easily adapted for other computers or non-Commodore printers.

## What Day Was It?

In calendar making, it is essential to know the correct day of the week for any given date. If you let D1 be the day of the week (D1=1 for Sunday, D1=2 for Monday, and so on), and let M, D, and Y be the month, day, and year, then D1 can be calculated as follows:

$$\begin{aligned}D1 &= \text{INT}(2.6 - (M - 2) - 0.2) + D + Y - 1900 + \text{INT}((Y - 1900)/4) \\D1 &= D1 + \text{INT}(19/4) - 2 * 19 \\D1 &= D1 - \text{INT}(D1/7) * 7 + 1\end{aligned}$$

Two modifications have to be used with the above formulation. Whenever M is 1 or 2, you have to add 12 to M and decrement Y by 1. In other words, the months January and February are thought of as the thirteenth and fourteenth months of the previous year. In addition, when M is equal to 4 or 9, the calculated value of D1 has to be increased by 1.

## Good for More Than a Century

This algorithm performs flawlessly for the twentieth and twenty-first centuries, up to the year 2100. If you want to go beyond that, you can make further modifications by reducing D1 by 1 after March 2100 (and repeating that every 100 years). Why? Because century years like 2100 and 2200 (that

are not divisible by 400) are not leap years, even though the algorithm treats them as if they are.

The programs may be modified, using changes like the one described above, to make them accurate for the next five centuries—provided, of course, that the current calendar system is not reformed. The last calendar reform was in 1752.

## **A Monthly Calendar**

Once you know the day of the week for the given date, the rest of the calendar-making task is just a matter of setting up and getting the proper format and display.

Program 1 will display a monthly calendar on the screen. In this and the other programs, the computer will briefly explain what the program does and then ask you to input the month and year of the calendar you wish to see. The numbers should be separated by a comma, and the year should be the full four digits (1984, not 84). After you press RETURN, the appropriate monthly calendar will be displayed on the screen.

Program 2 will give you a copy of what you see on the screen in the first program by printing it on your printer in enlarged form. This is a long program (it requires 8K memory expansion on the VIC) because it contains a set of enlarged numbers and characters, together with a bank of subroutines required to use them. The result is a calendar you can hang on the wall.

Program 3 also gives you a printed monthly calendar, but in a different format. It tabulates the days of the month as a list. It can serve as an appointment calendar for your desk, with room for short notes each day. Along with the regular date, you are told what day of the year it is. This program runs on the VIC without memory expansion.

## **A Year on a Page**

Program 4 will give you all 12 months of the year printed on one sheet. The message "Happy New Year" is at the top of the calendar, but you can put a different message there by modifying the text in line 7. This program will run on the unexpanded VIC.

In Programs 2, 3, and 4, after you input the month and year as requested, the computer prompts you to turn on the printer. Before you do this, you should set the perforation of the printing paper over the starting position of the print head

### 3: Applications

---

so that the calendar will appear entirely on one sheet of paper. The programs are written for the Commodore 1515 and 1525 printers. Other printers may require modifications to the program.

#### Program 1. Monthly Screen Calendar

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
80 DIMM$(12):FORI=1TO12:READM$(I):NEXTI      :rem 104
90 SYS65517:A=PEEK(781):IFA=40THENPOKE53281,1
                                                :rem 167
100 PRINT"{CLR}{3 DOWN}{2 RIGHT}THIS IS A PROGRAM"
    :PRINT"{6 RIGHT}TO SHOW A"                :rem 109
105 PRINT"{3 RIGHT}{PUR}MONTHLY CALENDAR{BLU}":PRI
    NT"{4 RIGHT}ON THE SCREEN"                :rem 155
110 PRINT"{2 DOWN}{2 RIGHT}PLEASE TYPE IN THE":PRI
    NT"{3 RIGHT}{RED}MONTH{BLU} AND {RED}YEAR{BLU}
    "                                          :rem 149
111 PRINT"{RIGHT}THAT YOU WISH TO SEE":PRINT"
    {DOWN}{2 RIGHT}(EXAMPLE: {RED}12,1983{BLU})
    {PUR}{2 DOWN}"                            :rem 180
130 PRINT"{5 RIGHT}";:INPUTM0,Y:PRINT"{2 DOWN}
    {5 RIGHT}{PUR}THANK YOU!{BLU}{DOWN}":FORI=1TO8
    00:NEXT                                     :rem 167
292 IFM0=1ORM0=3ORM0=5ORM0=7ORM0=8ORM0=10ORM0=12TH
    EN1=31                                      :rem 26
293 IFM0=4ORM0=6ORM0=9ORM0=11THENEN1=30      :rem 66
294 IFM0=2ANDY/4<>INT(Y/4)THENEN1=28         :rem 103
295 IFM0=2ANDY/4=INT(Y/4)THENGOSUB1400      :rem 83
297 PRINT"{CLR}{DOWN}{RIGHT}{RED}";M$(M0);" ";Y;"
    {BLU}"                                       :rem 123
298 GOSUB1350:IFA=40THENPRINT                 :rem 83
300 PRINT"{2 RIGHT}{RED}S{BLU}{2 RIGHT}M{2 RIGHT}T
    {2 RIGHT}W{2 RIGHT}T{2 RIGHT}F{2 RIGHT}S"
                                                :rem 109
305 GOSUB1360                                  :rem 226
310 D=1:GOSUB1050                              :rem 198
320 IFD1=7THENFORI=1TO19:PRINT"{RIGHT}";:NEXT:PRIN
    TD:IFA=40THENPRINT                          :rem 53
321 IFD1=7THEN330                              :rem 211
322 IFD1=6THENFORI=1TO16:PRINT"{RIGHT}";:NEXT:PRIN
    TD;:GOTO330                                  :rem 16
323 IFD1=5THENFORI=1TO13:PRINT"{RIGHT}";:NEXT:PRIN
    TD;:GOTO330                                  :rem 13
324 IFD1=4THENFORI=1TO10:PRINT"{RIGHT}";:NEXT:PRIN
    TD;:GOTO330                                  :rem 10
325 IFD1=3THENFORI=1TO7:PRINT"{RIGHT}";:NEXT:PRINT
    D;:GOTO330                                    :rem 224
```



```

326 IFD1=2THENFORI=1TO4:PRINT "{RIGHT}";:NEXT:PRINT
D;:GOTO330                                     :rem 221
327 IFD1=1THENPRINT "{RIGHT}{RED}";D;"{BLU}";:GOTO3
30                                             :rem 168
330 FORD=2TOE1:GOSUB1050                       :rem 201
331 IFD1=1ANDD<=9THENPRINT "{RIGHT}{RED}";D;"{BLU}"
;:GOTO345                                       :rem 114
332 IFD1=1ANDD>9THENPRINT "{RED}";D;"{BLU}";:GOTO34
5                                             :rem 27
333 IFD1=7THEN340                             :rem 215
334 IFD<=9THENPRINTD;:GOTO345                 :rem 105
335 PRINT "{LEFT}";D;:GOTO345                 :rem 210
340 IFD>9THENPRINT "{LEFT}";D:GOTO345        :rem 12
341 PRINTD                                     :rem 105
345 IFA=40ANDD1=7THENPRINT                   :rem 133
346 NEXTD                                     :rem 32
1045 PRINT:PRINT:FL=1:GOSUB1350:IFA=22THENPRINT"
{3 UP}"                                         :rem 57
1046 PRINT "{3 DOWN}ANOTHER?(Y/N)"           :rem 70
1047 GET R$:IF R$="" THEN 1047                 :rem 221
1048 IF R$=CHR$(89) THEN RUN                   :rem 246
1049 END                                       :rem 165
1050 IFM0=1THENM0=13:Y=Y-1:GOTO1080          :rem 80
1060 IFM0=2THENM0=14:Y=Y-1                   :rem 23
1080 M=M0-2                                    :rem 47
1100 D1=INT(2.6*M-0.2)+D+Y-1900+INT((Y-1900)/4)
                                             :rem 207
1150 D1=D1+INT(19/4)-2*19                     :rem 21
1200 D1=D1-INT(D1/7)*7+1                     :rem 235
1210 IFM0=4ORM0=9THEND1=D1+1                 :rem 135
1230 IFM0=13THENM0=1:Y=Y+1:GOTO1245         :rem 81
1240 IFM0=14THENM0=2:Y=Y+1:D1=D1+1          :rem 210
1244 IFD1=8THEND1=1                           :rem 86
1245 IF(Y=2100ANDM0>=3)OR(Y>2100)THEND1=D1-1:IFD1=
0THEND1=7                                       :rem 198
1247 IF(Y=2200ANDM0>=3)OR(Y>2200)THEND1=D1-1:IFD1=
0THEND1=7                                       :rem 202
1249 IF(Y=2300ANDM0>=3)OR(Y>2300)THEND1=D1-1:IFD1=
0THEND1=7                                       :rem 206
1250 RETURN                                   :rem 168
1350 IFFL=0THENPRINT:FORI=1TO22:PRINT"*";:NEXT:PRI
NT:RETURN                                       :rem 188
1355 IFD1=7THENPRINT "{3 UP}":FORI=1TO22:PRINT"*";:
NEXT:PRINT "{UP}":RETURN                       :rem 119
1358 FORI=1TO22:PRINT"*";:NEXT:PRINT "{UP}":RETURN
                                             :rem 21
1360 PRINT "{2 SPACES}{T}{2 SPACES}{T}{2 SPACES}{T}
{2 SPACES}{T}{2 SPACES}{T}{2 SPACES}{T}
{2 SPACES}{T}":RETURN                          :rem 42

```

### 3: Applications

---

```
1400 IF(Y/100=INT(Y/100))AND(Y/400<>INT(Y/400))THE
      NE1=28:GOTO1410                               :rem 231
1405 E1=29                                           :rem 232
1410 RETURN                                          :rem 166
1420 DATA"{3 SPACES}JANUARY","{3 SPACES}FEBRUARY",
      "{4 SPACES}MARCH","{4 SPACES}APRIL"          :rem 210
1430 DATA"{5 SPACES}MAY","{5 SPACES}JUNE",
      "{5 SPACES}JULY","{4 SPACES}AUGUST"         :rem 172
1440 DATA"{2 SPACES}SEPTEMBER","{3 SPACES}OCTOBER"
      ,"{3 SPACES}NOVEMBER","{3 SPACES}DECEMBER"
                                                    :rem 193
```

### Program 2. Monthly Calendar Printer

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
1 GOTO10                                           :rem 203
5 E1=1:E2=1:E3=1:E4=1:E5=1:E6=1:E7=1           :rem 226
6 GOSUB1109:D8=D7-1:RETURN                        :rem 103
10 OPEN1,4:SYS65517:A=PEEK(781):IFA=40THENPOKE5328
    1,1                                             :rem 156
20 GOSUB4000:GOSUB3200:PRINT#1,""                :rem 176
30 ONM0GOSUB3010,3020,3030,3040,3050,3060,3070,308
    0,3090,3100,3110,3120                         :rem 56
40 PRINT#1,"":PRINT#1,"":GOSUB1610:GOSUB1650:GOSUB
    1660                                           :rem 207
80 OND9GOSUB1811,1821,1831,1841,1851,1861,1871
                                                    :rem 172
99 PRINT#1,"":PRINT#1,""                          :rem 78
100 G1=D8                                          :rem 194
105 G=G1:GOSUB1720:D1=D:E1=E                     :rem 120
110 G2=G+1:G=G2:GOSUB1720:D2=D:E2=E             :rem 10
115 G3=G+1:G=G3:GOSUB1720:D3=D:E3=E             :rem 19
120 G4=G+1:G=G4:GOSUB1720:D4=D:E4=E             :rem 19
125 G5=G+1:G=G5:GOSUB1720:D5=D:E5=E             :rem 28
130 G6=G+1:G=G6:GOSUB1720:D6=D:E6=E             :rem 28
135 G7=G+1:G=G7:GOSUB1720:D7=D:E7=E             :rem 37
140 G1=G7+1:GOSUB1109:PRINT#1,"":PRINT#1,"":IFG1<=
    E9THEN105                                       :rem 188
155 PRINT#1,""                                     :rem 236
1000 GOTO5000                                       :rem 191
1109 GOSUB2000:X=E1:X1=D1:GOSUB11000            :rem 115
1120 X=E2:X1=D2:GOSUB11000                       :rem 242
1130 X=E3:X1=D3:GOSUB11000                       :rem 245
1140 X=E4:X1=D4:GOSUB11000                       :rem 248
1150 X=E5:X1=D5:GOSUB11000                       :rem 251
1160 X=E6:X1=D6:GOSUB11000                       :rem 254
1170 X=E7:X1=D7:FL=1:GOSUB11000                 :rem 59
1209 GOSUB2000:X=E1:X1=D1:GOSUB12000           :rem 117
1220 X=E2:X1=D2:GOSUB12000                     :rem 244
1230 X=E3:X1=D3:GOSUB12000                     :rem 247
```

```

1240 X=E4:X1=D4:GOSUB12000           :rem 250
1250 X=E5:X1=D5:GOSUB12000           :rem 253
1260 X=E6:X1=D6:GOSUB12000           :rem 0
1270 X=E7:X1=D7:FL=1:GOSUB12000      :rem 61
1309 GOSUB2000:X=E1:X1=D1:GOSUB13000 :rem 119
1320 X=E2:X1=D2:GOSUB13000           :rem 246
1330 X=E3:X1=D3:GOSUB13000           :rem 249
1340 X=E4:X1=D4:GOSUB13000           :rem 252
1350 X=E5:X1=D5:GOSUB13000           :rem 255
1360 X=E6:X1=D6:GOSUB13000           :rem 2
1370 X=E7:X1=D7:FL=1:GOSUB13000      :rem 63
1409 GOSUB2000:X=E1:X1=D1:GOSUB14000 :rem 121
1420 X=E2:X1=D2:GOSUB14000           :rem 248
1430 X=E3:X1=D3:GOSUB14000           :rem 251
1440 X=E4:X1=D4:GOSUB14000           :rem 254
1450 X=E5:X1=D5:GOSUB14000           :rem 1
1460 X=E6:X1=D6:GOSUB14000           :rem 4
1470 X=E7:X1=D7:FL=1:GOSUB14000      :rem 65
1509 GOSUB2000:X=E1:X1=D1:GOSUB15000 :rem 123
1520 X=E2:X1=D2:GOSUB15000           :rem 250
1530 X=E3:X1=D3:GOSUB15000           :rem 253
1540 X=E4:X1=D4:GOSUB15000           :rem 0
1550 X=E5:X1=D5:GOSUB15000           :rem 3
1560 X=E6:X1=D6:GOSUB15000           :rem 6
1570 X=E7:X1=D7:FL=1:GOSUB15000      :rem 67
1600 RETURN                           :rem 167
1610 PRINT#1,"{5 SPACES}";:PRINT#1,CHR$(14)"SUN";:
      PRINT#1,CHR$(15){5 SPACES}";    :rem 69
1611 PRINT#1,CHR$(14)"MON";:PRINT#1,CHR$(15)"
      {5 SPACES}";                    :rem 116
1612 PRINT#1,CHR$(14)"TUE";:PRINT#1,CHR$(15)"
      {5 SPACES}";                    :rem 121
1613 PRINT#1,CHR$(14)"WED";:PRINT#1,CHR$(15)"
      {5 SPACES}";                    :rem 108
1614 PRINT#1,CHR$(14)"THU";:PRINT#1,CHR$(15)"
      {5 SPACES}";                    :rem 126
1615 PRINT#1,CHR$(14)"FRI";:PRINT#1,CHR$(15)"
      {5 SPACES}";                    :rem 111
1616 PRINT#1,CHR$(14)"SAT":PRINT#1,CHR$(15)" "
                                          :rem 1
1620 PRINT#1,"{5 SPACES}";:PRINT#1,CHR$(14)"---";:
      PRINT#1,CHR$(15){5 SPACES}";    :rem 215
1621 PRINT#1,CHR$(14)"---";:PRINT#1,CHR$(15)"
      {5 SPACES}";                    :rem 18
1622 PRINT#1,CHR$(14)"---";:PRINT#1,CHR$(15)"
      {5 SPACES}";                    :rem 19
1623 PRINT#1,CHR$(14)"---";:PRINT#1,CHR$(15)"
      {5 SPACES}";                    :rem 20
1624 PRINT#1,CHR$(14)"---";:PRINT#1,CHR$(15)"
      {5 SPACES}";                    :rem 21

```

### 3: Applications

---

```
1625 PRINT#1,CHR$(14)"---";:PRINT#1,CHR$(15)"
{5 SPACES}";:rem 22
1626 PRINT#1,CHR$(14)"---":PRINT#1,CHR$(15)" ":RET
URN:rem 187
1650 IFM0=1ORM0=3ORM0=5ORM0=7ORM0=8ORM0=10ORM0=12T
HENE9=31:rem 81
1652 IFM0=4ORM0=6ORM0=9ORM0=11THENE9=30:rem 122
1654 IFM0=2ANDY/4<>INT(Y/4)THENE9=28:rem 160
1656 IFM0=2ANDY/4=INT(Y/4)THENE9=29:rem 102
1658 RETURN:rem 180
1660 IFM0=1THENM0=13:Y=Y-1:GOTO1670:rem 92
1665 IFM0=2THENM0=14:Y=Y-1:rem 34
1670 M=M0-2:rem 52
1675 D9=INT(2.6*M-0.2)+D+Y-1900+INT((Y-1900)/4)
:rem 232
1680 D9=D9+INT(19/4)-2*19:rem 45
1685 D9=D9-INT(D9/7)*7+1:rem 20
1690 IFM0=4ORM0=9THEND9=D9+1:rem 163
1695 IFM0=13THENM0=1:Y=Y+1:GOTO1710:rem 93
1700 IFM0=14THENM0=2:Y=Y+1:D9=D9+1:rem 227
1705 IFD9=8THEND9=1:rem 104
1710 IF(Y=2100ANDM0>=3)OR(Y>2100)THEND9=D9-1:IFD9=
0THEND9=7:rem 227
1711 IF(Y=2200ANDM0>=3)OR(Y>2200)THEND9=D9-1:IFD9=
0THEND9=7:rem 230
1712 IF(Y=2300ANDM0>=3)OR(Y>2300)THEND9=D9-1:IFD9=
0THEND9=7:rem 233
1715 RETURN:rem 174
1720 IFG>E9THENGOTO1740:rem 144
1722 IFG<10THENGOTO1742:rem 117
1726 IFG>=10ANDG<20THENGOTO1746:rem 116
1728 IFG>=20ANDG<30THENGOTO1748:rem 122
1730 IFG>=30THENGOTO1750:rem 180
1740 D=1:E=1:GOTO1755:rem 176
1742 D=G+2:E=1:GOTO1755:rem 37
1746 D=G-10+2:E=2:GOTO1755:rem 184
1748 D=G-20+2:E=3:GOTO1755:rem 188
1750 D=G-30+2:E=4:rem 114
1755 RETURN:rem 178
1811 D1=1:D2=3:D3=4:D4=5:D5=6:D6=7:D7=8:GOSUB5:RET
URN:rem 149
1821 D1=1:D2=1:D3=3:D4=4:D5=5:D6=6:D7=7:GOSUB5:RET
URN:rem 143
1831 D1=1:D2=1:D3=1:D4=3:D5=4:D6=5:D7=6:GOSUB5:RET
URN:rem 138
1841 D1=1:D2=1:D3=1:D4=1:D5=3:D6=4:D7=5:GOSUB5:RET
URN:rem 134
1851 D1=1:D2=1:D3=1:D4=1:D5=1:D6=3:D7=4:GOSUB5:RET
URN:rem 131
```

```
1861 D1=1:D2=1:D3=1:D4=1:D5=1:D6=1:D7=3:GOSUB5:RET      :rem 129
    URN
1871 D1=3:D2=4:D3=5:D4=6:D5=7:D6=8:D7=9:GOSUB5:RET      :rem 163
    URN
2000 PRINT#1,"{4 SPACES}";:RETURN                       :rem 104
2001 PRINT#1," [2 +] ";:RETURN                          :rem 181
2002 PRINT#1,"[+] {2 SPACES} [+]";:RETURN              :rem 182
2003 PRINT#1,"[+] {2 SPACES} [+]";:RETURN              :rem 183
2004 PRINT#1,"[+] {2 SPACES} [+]";:RETURN              :rem 184
2005 PRINT#1," [2 +] ";:RETURN                          :rem 185
2011 PRINT#1," [+] {2 SPACES}";:RETURN                  :rem 16
2012 PRINT#1," [+] {2 SPACES}";:RETURN                  :rem 17
2013 PRINT#1," [+] {2 SPACES}";:RETURN                  :rem 18
2014 PRINT#1," [+] {2 SPACES}";:RETURN                  :rem 19
2015 PRINT#1," [+] {2 SPACES}";:RETURN                  :rem 20
2021 PRINT#1," [2 +] ";:RETURN                          :rem 183
2022 PRINT#1,"[+] {2 SPACES} [+]";:RETURN              :rem 184
2023 PRINT#1,"{2 SPACES} [+] ";:RETURN                  :rem 19
2024 PRINT#1," [+] {2 SPACES}";:RETURN                  :rem 20
2025 PRINT#1,"[4 +]";:RETURN                            :rem 7
2031 PRINT#1,"[3 +] ";:RETURN                            :rem 94
2032 PRINT#1,"{3 SPACES} [+]";:RETURN                   :rem 19
2033 PRINT#1," [2 +] ";:RETURN                          :rem 186
2034 PRINT#1,"{3 SPACES} [+]";:RETURN                   :rem 21
2035 PRINT#1,"[3 +] ";:RETURN                            :rem 98
2041 PRINT#1,"{2 SPACES} [+] ";:RETURN                  :rem 19
2042 PRINT#1," [2 +] ";:RETURN                          :rem 186
2043 PRINT#1,"[+] [+] ";:RETURN                         :rem 187
2044 PRINT#1,"[4 +]";:RETURN                            :rem 8
2045 PRINT#1,"{2 SPACES} [+] ";:RETURN                   :rem 23
2051 PRINT#1,"[4 +]";:RETURN                            :rem 6
2052 PRINT#1,"[+] {3 SPACES}";:RETURN                   :rem 21
2053 PRINT#1,"[3 +] ";:RETURN                            :rem 98
2054 PRINT#1,"{3 SPACES} [+]";:RETURN                   :rem 23
2055 PRINT#1,"[3 +] ";:RETURN                            :rem 100
2061 PRINT#1," [2 +] ";:RETURN                          :rem 187
2062 PRINT#1,"[+] {3 SPACES}";:RETURN                   :rem 22
2063 PRINT#1,"[3 +] ";:RETURN                            :rem 99
2064 PRINT#1,"[+] {2 SPACES} [+]";:RETURN              :rem 190
2065 PRINT#1," [2 +] ";:RETURN                          :rem 191
2071 PRINT#1,"[4 +]";:RETURN                            :rem 8
2072 PRINT#1,"{3 SPACES} [+]";:RETURN                   :rem 23
2073 PRINT#1,"{2 SPACES} [+] ";:RETURN                   :rem 24
2074 PRINT#1," [+] {2 SPACES}";:RETURN                   :rem 25
2075 PRINT#1," [+] {2 SPACES}";:RETURN                   :rem 26
2081 PRINT#1," [2 +] ";:RETURN                          :rem 189
2082 PRINT#1,"[+] {2 SPACES} [+]";:RETURN              :rem 190
2083 PRINT#1," [2 +] ";:RETURN                          :rem 191
2084 PRINT#1,"[+] {2 SPACES} [+]";:RETURN              :rem 192
```

### 3: Applications

---

```
2085 PRINT#1," [2 +] ";:RETURN :rem 193
2091 PRINT#1," [2 +] ";:RETURN :rem 190
2092 PRINT#1,"[+] {2 SPACES} [+] ";:RETURN :rem 191
2093 PRINT#1," [3 +] ";:RETURN :rem 102
2094 PRINT#1," {3 SPACES} [+] ";:RETURN :rem 27
2095 PRINT#1," [2 +] ";:RETURN :rem 194
2111 PRINT#1," {2 SPACES} [+] ";:RETURN :rem 17
2112 PRINT#1," {2 SPACES} [+] ";:RETURN :rem 18
2113 PRINT#1," {2 SPACES} [+] ";:RETURN :rem 19
2114 PRINT#1," {2 SPACES} [+] ";:RETURN :rem 20
2115 PRINT#1," {2 SPACES} [+] ";:RETURN :rem 21
3010 GOSUB2000:PRINT#1," [3 +] {3 SPACES} [3 +]
{2 SPACES} [+] {3 SPACES} [+] " :rem 193
3011 GOSUB2000:PRINT#1," {2 SPACES} [+] {3 SPACES} [+]
{3 SPACES} [+] [2 +] {2 SPACES} [+] " :rem 118
3012 GOSUB2000:PRINT#1," {2 SPACES} [+] {3 SPACES} [+]
{3 SPACES} [+] [3 +] [3 +] [3 +] " :rem 119
3013 GOSUB2000:PRINT#1," [3 +] [3 +] {3 SPACES} [5 +] [3 +]
{2 SPACES} [2 +] " :rem 16
3014 GOSUB2000:PRINT#1," [3 +] {3 SPACES} [3 +]
{3 SPACES} [3 +] [3 +] {3 SPACES} [3 +] " :rem 31
3015 RETURN :rem 169
3020 GOSUB2000:PRINT#1," [5 +] [5 +] [4 +] "
:rem 166
3021 GOSUB2000:PRINT#1," [3 +] {5 SPACES} [3 +] {5 SPACES}
[3 +] {3 SPACES} [3 +] " :rem 43
3022 GOSUB2000:PRINT#1," [3 +] {3 SPACES} [4 +]
{2 SPACES} [4 +] " :rem 182
3023 GOSUB2000:PRINT#1," [3 +] {5 SPACES} [3 +] {5 SPACES}
[3 +] {3 SPACES} [3 +] " :rem 45
3024 GOSUB2000:PRINT#1," [3 +] {5 SPACES} [5 +] [4 +] "
:rem 18
3025 RETURN :rem 170
3030 GOSUB2000:PRINT#1," [3 +] {3 SPACES} [3 +] {2 SPACES}
[3 +] {2 SPACES} [4 +] " :rem 105
3031 GOSUB2000:PRINT#1," [2 +] [2 +] [3 +] {3 SPACES}
[3 +] [3 +] {3 SPACES} [3 +] " :rem 196
3032 GOSUB2000:PRINT#1," [3 +] [3 +] [3 +] [3 +] {3 SPACES}
[3 +] [4 +] " :rem 107
3033 GOSUB2000:PRINT#1," [3 +] [3 +] [3 +] [5 +] [3 +]
{2 SPACES} [3 +] " :rem 18
3034 GOSUB2000:PRINT#1," [3 +] {3 SPACES} [3 +] [3 +]
{3 SPACES} [3 +] [3 +] {3 SPACES} [3 +] " :rem 123
3035 RETURN :rem 171
3040 GOSUB2000:PRINT#1," [3 +] {2 SPACES} [4 +]
{2 SPACES} [4 +] " :rem 182
3041 GOSUB2000:PRINT#1," [3 +] {3 SPACES} [3 +] [3 +]
{3 SPACES} [3 +] [3 +] {3 SPACES} [3 +] " :rem 121
3042 GOSUB2000:PRINT#1," [3 +] {3 SPACES} [3 +] [4 +]
{2 SPACES} [4 +] " :rem 18
```

```
3043 GOSUB2000:PRINT#1,"[5 +] [5 SPACES][+]  
      {2 SPACES}[+]" :rem 199  
3044 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [5  
      {5 SPACES}[+]{3 SPACES}[+]" :rem 214  
3045 RETURN :rem 172  
3050 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+]{2 SPACES}  
      [3 +]{2 SPACES}[+]{3 SPACES}[+]" :rem 31  
3051 GOSUB2000:PRINT#1,"[2 +] [2 +] [3 SPACES]  
      [+][+]{3 SPACES}[+]" :rem 198  
3052 GOSUB2000:PRINT#1,"[+] [3 SPACES] [3 SPACES]  
      [+]{2 SPACES}[+]" :rem 33  
3053 GOSUB2000:PRINT#1,"[+] [3 SPACES] [5 +]  
      {3 SPACES}[+]{2 SPACES}" :rem 110  
3054 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [3  
      {3 SPACES}[+]{3 SPACES}[+]{2 SPACES}" :rem 215  
3055 RETURN :rem 173  
3060 GOSUB2000:PRINT#1," [3 +]{2 SPACES}[+]  
      {3 SPACES}[+] [3 SPACES][+]" :rem 32  
3061 GOSUB2000:PRINT#1,"{2 SPACES}[+]{3 SPACES}[+]  
      {3 SPACES}[+] [2 +]{2 SPACES}[+]" :rem 123  
3062 GOSUB2000:PRINT#1,"{2 SPACES}[+]{3 SPACES}[+]  
      {3 SPACES}[+] [3 SPACES][+]" :rem 124  
3063 GOSUB2000:PRINT#1,"[+] [3 SPACES][2 +]  
      {3 SPACES}[+] [2 SPACES][2 +]" :rem 35  
3064 GOSUB2000:PRINT#1,"[3 +]{4 SPACES}[3 +]  
      {2 SPACES}[+]{3 SPACES}[+]" :rem 202  
3065 RETURN :rem 174  
3070 GOSUB2000:PRINT#1," [3 +]{2 SPACES}[+]  
      {3 SPACES}[+] [4 SPACES]" :rem 123  
3071 GOSUB2000:PRINT#1,"{2 SPACES}[+]{3 SPACES}[+]  
      {3 SPACES}[+] [4 SPACES]" :rem 48  
3072 GOSUB2000:PRINT#1,"{2 SPACES}[+]{3 SPACES}[+]  
      {3 SPACES}[+] [4 SPACES]" :rem 49  
3073 GOSUB2000:PRINT#1,"[+] [3 SPACES][4 SPACES]  
      {3 SPACES}[+] [4 SPACES]" :rem 216  
3074 GOSUB2000:PRINT#1,"[3 +]{4 SPACES}[3 +]  
      {2 SPACES}[5 +]" :rem 189  
3075 RETURN :rem 175  
3080 GOSUB2000:PRINT#1," [3 +]{2 SPACES}[+]  
      {3 SPACES}[+]{2 SPACES}[3 +]" :rem 200  
3081 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [4  
      {3 SPACES}[+] [4 SPACES]" :rem 215  
3082 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [2  
      {3 SPACES}[+] [2 SPACES][2 +]" :rem 36  
3083 GOSUB2000:PRINT#1,"[5 +] [3 SPACES][+]  
      {3 SPACES}[+]" :rem 113  
3084 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+]{2 SPACES}  
      [3 +]{3 SPACES}[3 +]" :rem 204  
3085 RETURN :rem 176  
3090 GOSUB2000:PRINT#1," [4 +] [5 +] [4 +]" :rem 7
```

### 3: Applications

---

```
3091 GOSUB2000:PRINT#1,"[+]{5 SPACES}[+]{5 SPACES}
[+]{3 SPACES}[+]" :rem 50
3092 GOSUB2000:PRINT#1," [3 +]{2 SPACES}[4 +]
{2 SPACES}[4 +] " :rem 189
3093 GOSUB2000:PRINT#1,"{4 SPACES}[+] [+]
{5 SPACES}[+]{4 SPACES}" :rem 142
3094 GOSUB2000:PRINT#1,"[4 +]{2 SPACES}[5 +] [+]
{4 SPACES}" :rem 25
3095 RETURN :rem 177
3100 GOSUB2000:PRINT#1," [3 +]{3 SPACES}[3 +]
{2 SPACES}[5 +]" :rem 179
3101 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [+]
{3 SPACES}[+]{3 SPACES}[+]{2 SPACES}" :rem 208
3102 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [+]
{7 SPACES}[+]{2 SPACES}" :rem 43
3103 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [+]
{3 SPACES}[+]{3 SPACES}[+]{2 SPACES}" :rem 210
3104 GOSUB2000:PRINT#1," [3 +]{3 SPACES}[3 +]
{4 SPACES}[+]{2 SPACES}" :rem 31
3105 RETURN :rem 169
3110 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+]{2 SPACES}
[3 +]{2 SPACES}[+]{3 SPACES}[+]" :rem 28
3111 GOSUB2000:PRINT#1,"[2 +]{2 SPACES}[+] [+]
{3 SPACES}[+] [+] {3 SPACES}[+]" :rem 29
3112 GOSUB2000:PRINT#1,"[+] [+] [+] [+] {3 SPACES}
[+] [+] {3 SPACES}[+]" :rem 30
3113 GOSUB2000:PRINT#1,"[+]{2 SPACES}[2 +] [+]
{3 SPACES}[+]{2 SPACES}[+] [+] " :rem 31
3114 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+]{2 SPACES}
[3 +]{4 SPACES}[+]{2 SPACES}" :rem 122
3115 RETURN :rem 170
3120 GOSUB2000:PRINT#1,"[4 +]{2 SPACES}[5 +]
{2 SPACES}[3 +] " :rem 91
3121 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [+]
{5 SPACES}[+]{3 SPACES}[+]" :rem 210
3122 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [4 +]
{2 SPACES}[+]{4 SPACES}" :rem 31
3123 GOSUB2000:PRINT#1,"[+]{3 SPACES}[+] [+]
{5 SPACES}[+]{3 SPACES}[+]" :rem 212
3124 GOSUB2000:PRINT#1,"[4 +]{2 SPACES}[5 +]
{2 SPACES}[3 +] " :rem 95
3125 RETURN :rem 171
3200 I1=INT(Y/1000):J1=Y-I1*1000:I2=INT(J1/100):J2
=J1-I2*100:I3=INT(J2/10) :rem 83
3210 I4=J2-I3*10 :rem 48
3211 IFI2=0THENI2=10 :rem 134
3212 IFI3=0THENI3=10 :rem 137
3213 IFI4=0THENI4=10 :rem 140
3214 GOSUB2000:X=I1:GOSUB6000:GOSUB2000:X=I2:GOSUB
6000:GOSUB2000:X=I3:GOSUB6000 :rem 98
```



```

3215 GOSUB2000:X=I4:FL=1:GOSUB6000           :rem 19
3314 GOSUB2000:X=I1:GOSUB7000:GOSUB2000:X=I2:GOSUB
7000:GOSUB2000:X=I3:GOSUB7000           :rem 102
3315 GOSUB2000:X=I4:FL=1:GOSUB7000           :rem 21
3414 GOSUB2000:X=I1:GOSUB8000:GOSUB2000:X=I2:GOSUB
8000:GOSUB2000:X=I3:GOSUB8000           :rem 106
3415 GOSUB2000:X=I4:FL=1:GOSUB8000           :rem 23
3514 GOSUB2000:X=I1:GOSUB9000:GOSUB2000:X=I2:GOSUB
9000:GOSUB2000:X=I3:GOSUB9000           :rem 110
3515 GOSUB2000:X=I4:FL=1:GOSUB9000           :rem 25
3614 GOSUB2000:X=I1:GOSUB10000:GOSUB2000:X=I2:GOSU
B10000:GOSUB2000:X=I3                       :rem 60
3615 GOSUB10000:GOSUB2000:X=I4:FL=1:GOSUB10000:RET
URN                                           :rem 7
4000 PRINT"{CLR}{DOWN}{2 SPACES}THIS IS A PROGRAM"
:PRINT"{5 RIGHT}TO PRINT A"                 :rem 115
4020 PRINT"{2 SPACES}{PUR}MONTHLY CALENDAR{BLU}":P
RINT"{3 RIGHT}ON THE PRINTER"               :rem 187
4030 PRINT"{DOWN}{2 RIGHT}PLEASE TYPE IN THE":PRIN
T"{3 RIGHT}{RED}MONTH{BLU} AND {RED}YEAR{BLU}
"                                             :rem 185
4035 PRINT" THAT YOU WISH TO SEE":PRINT"{2 SPACES}
(EXAMPLE: {RED}12,1983{BLU}){PUR}{DOWN}":PRIN
TTAB(5);                                     :rem 211
4060 INPUTM0,Y                               :rem 92
4080 PRINT"{2 DOWN}{2 SPACES}{BLU}THANK YOU! NOW--
":PRINT" PLEASE {PUR}TURN ON{BLU} THE" :rem 7
4085 PRINT"PRINTER AND THEN TYPE":PRINTTAB(8)"
{PUR}OK{DOWN}":INPUTR$                       :rem 252
4110 IFR$<>"OK"THEN4080                     :rem 30
4130 PRINT"{BLU}PRINTING{DOWN}":FORI=1TO800:NEXT:R
ETURN                                         :rem 218
4999 PRINT#1,CHR$(15)" "                    :rem 232
5000 GOSUB1620                               :rem 14
5001 CLOSE1:END                             :rem 126
6000 ONXGOSUB2011,2021,2031,2041,2051,2061,2071,20
81,2091,2001                                 :rem 146
6010 IFFL<>1THENPRINT#1," ";:RETURN          :rem 104
6020 PRINT#1,"":FL=0:RETURN                  :rem 108
7000 ONXGOSUB2012,2022,2032,2042,2052,2062,2072,20
82,2092,2002                                 :rem 157
7010 IFFL<>1THENPRINT#1," ";:RETURN          :rem 105
7020 PRINT#1,"":FL=0:RETURN                  :rem 109
8000 ONXGOSUB2013,2023,2033,2043,2053,2063,2073,20
83,2093,2003                                 :rem 168
8010 IFFL<>1THENPRINT#1," ";:RETURN          :rem 106
8020 PRINT#1,"":FL=0:RETURN                  :rem 110
9000 ONXGOSUB2014,2024,2034,2044,2054,2064,2074,20
84,2094,2004                                 :rem 179
9010 IFFL<>1THENPRINT#1," ";:RETURN          :rem 107

```

### 3: Applications

---

```
9020 PRINT#1, "":FL=0:RETURN :rem 111
10000 ONXGOSUB2015,2025,2035,2045,2055,2065,2075,2
      085,2095,2005 :rem 229
10010 IFFL<>1THENPRINT#1, " ":RETURN :rem 147
10020 PRINT#1, "":FL=0:RETURN :rem 151
11000 ONXGOSUB2000,2111,2021,2031:PRINT#1, " ";
      :rem 195
11010 ONX1GOSUB2000,2001,2011,2021,2031,2041,2051,
      2061,2071,2081,2091 :rem 222
11020 IFFL<>1THENPRINT#1,"{2 SPACES}":RETURN
      :rem 149
11030 FL=0:PRINT#1, "":RETURN :rem 153
12000 ONXGOSUB2000,2112,2022,2032:PRINT#1, " ";
      :rem 199
12010 ONX1GOSUB2000,2002,2012,2022,2032,2042,2052,
      2062,2072,2082,2092 :rem 233
12020 IFFL<>1THENPRINT#1,"{2 SPACES}":RETURN
      :rem 150
12030 FL=0:PRINT#1, "":RETURN :rem 154
13000 ONXGOSUB2000,2113,2023,2033:PRINT#1, " ";
      :rem 203
13010 ONX1GOSUB2000,2003,2013,2023,2033,2043,2053,
      2063,2073,2083,2093 :rem 244
13020 IFFL<>1THENPRINT#1,"{2 SPACES}":RETURN
      :rem 151
13030 FL=0:PRINT#1, "":RETURN :rem 155
14000 ONXGOSUB2000,2114,2024,2034:PRINT#1, " ";
      :rem 207
14010 ONX1GOSUB2000,2004,2014,2024,2034,2044,2054,
      2064,2074,2084,2094 :rem 255
14020 IFFL<>1THENPRINT#1,"{2 SPACES}":RETURN
      :rem 152
14030 FL=0:PRINT#1, "":RETURN :rem 156
15000 ONXGOSUB2000,2115,2025,2035:PRINT#1, " ";
      :rem 211
15010 ONX1GOSUB2000,2005,2015,2025,2035,2045,2055,
      2065,2075,2085,2095 :rem 10
15020 IFFL<>1THENPRINT#1,"{2 SPACES}":RETURN
      :rem 153
15030 FL=0:PRINT#1, "":RETURN :rem 157
```

### Program 3. Monthly Appointment Calendar Printer

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
80 DIMM$(12),W$(7):FORI=1TO12:READM$(I):NEXTI:FORI
  =1TO7:READW$(I):NEXTI :rem 118
90 SYS65517:A=PEEK(781):IFA=40THENPOKE53281,1
      :rem 167
100 PRINT"{CLR}{DOWN}{2 SPACES}THIS IS A PROGRAM":
  PRINT"{6 RIGHT}TO SHOW A" :rem 17
```

```

105 PRINT"{2 RIGHT}{PUR}MONTHLY CALENDAR{BLU}":PRI
    NT"{3 RIGHT}ON THE PRINTER{DOWN}"           :rem 214
110 PRINT"{RIGHT}PLEASE TYPE IN THE":PRINT"
    {3 RIGHT}{RED}MONTH{BLU} AND {RED}YEAR{BLU}"
                                                :rem 86
111 PRINT"THAT YOU WISH TO SEE":PRINT"{RIGHT}(EXAM
    PLE: {RED}12,1983{BLU}){PUR}{2 DOWN}"       :rem 105
120 PRINTTAB(5);:INPUTM0,Y                       :rem 132
130 PRINT"{2 DOWN}{2 SPACES}{BLU}THANK YOU! NOW--"
    :PRINT" PLEASE {PUR}TURN ON{BLU} THE"       :rem 207
131 PRINT"PRINTER AND THEN TYPE":PRINTTAB(9)"{PUR}
    OK{DOWN}":INPUTR$                             :rem 193
151 IFR$<>"OK"THEN130                             :rem 183
154 PRINT"{BLU}PRINTING{DOWN}":FORI=1TO800:NEXT:GO
    SUB1292:OPEN1,4                               :rem 23
202 PRINT#1,CHR$(14)"{3 SPACES}";M$(M0);" ";Y:GOSU
    B1600:GOSUB1700:FORD=1TOEL:J1=J1+1          :rem 225
210 GOSUB1050:IFD<10THENG$=" "                  :rem 158
213 IFD>=10THENG$=" "                            :rem 96
214 IFD1=1THENPRINT#1,CHR$(15)"{3 SPACES}"W$(D1);C
    HR$(14)G$;"{RVS}"D"{OFF}";CHR$(15)"(";J1;")"
                                                :rem 71
215 IFD1=1THENGOSUB1600                          :rem 128
217 IFD1=1THENGOTO220                             :rem 8
219 PRINT#1,CHR$(15)"{3 SPACES}"W$(D1);CHR$(14)G$;
    D;CHR$(15)"(";J1;")":GOSUB1600             :rem 0
220 NEXTD                                         :rem 23
1000 CLOSE1:END                                  :rem 121
1050 IFM0=1THENM0=13:Y=Y-1:GOTO1080             :rem 80
1060 IFM0=2THENM0=14:Y=Y-1                     :rem 23
1080 M=M0-2                                       :rem 47
1100 D1=INT(2.6*M-0.2)+D+Y-1900+INT((Y-1900)/4)
                                                :rem 207
1150 D1=D1+INT(19/4)-2*19                         :rem 21
1200 D1=D1-INT(D1/7)*7+1                         :rem 235
1210 IFM0=4ORM0=9THEND1=D1+1                    :rem 135
1230 IFM0=13THENM0=1:Y=Y+1:GOTO1245            :rem 81
1240 IFM0=14THENM0=2:Y=Y+1:D1=D1+1             :rem 210
1244 IFD1=8THEND1=1                              :rem 86
1245 IF(Y=2100ANDM0>=3)OR(Y>2100)THEND1=D1-1:IFD1=
    0THEND1=7                                     :rem 198
1247 IF(Y=2200ANDM0>=3)OR(Y>2200)THEND1=D1-1:IFD1=
    0THEND1=7                                     :rem 202
1249 IF(Y=2300ANDM0>=3)OR(Y>2300)THEND1=D1-1:IFD1=
    0THEND1=7                                     :rem 206
1250 RETURN                                       :rem 168
1292 IFM0=1ORM0=3ORM0=5ORM0=7ORM0=8ORM0=10ORM0=12T
    HENE1=31                                       :rem 75
1293 IFM0=4ORM0=6ORM0=9ORM0=11THENE1=30        :rem 115
1294 IFM0=2ANDY/4<>INT(Y/4)THENE1=28           :rem 152

```

### 3: Applications

---

```
1295 IFMØ=2ANDY/4=INT(Y/4)THENGOSUB14ØØ :rem 132
1296 RETURN :rem 178
14ØØ IF(Y/1ØØ=INT(Y/1ØØ))AND(Y/4ØØ<>INT(Y/4ØØ))THE
    NE1=28:GOTO141Ø :rem 231
14Ø5 E1=29 :rem 232
141Ø RETURN :rem 166
16ØØ FORI=1TO2Ø:PRINT#1,CHR$(15)" ";:NEXTI:rem 17Ø
16Ø5 FORK=1TO18:PRINT#1,"."; " "; " ";:NEXTK:PRINT#1
    ,".": :rem 231
161Ø RETURN :rem 168
17ØØ IFMØ=1THENJ1=Ø :rem 89
17Ø2 IFMØ=2THENJ1=31 :rem 144
17Ø4 IFMØ=3THENJ1=59 :rem 157
17Ø6 IFMØ=4THENJ1=9Ø :rem 155
17Ø7 IFMØ=5THENJ1=12Ø :rem 199
17Ø9 IFMØ=6THENJ1=151 :rem 2Ø6
1711 IFMØ=7THENJ1=181 :rem 2Ø3
1713 IFMØ=8THENJ1=212 :rem 2Ø1
1715 IFMØ=9THENJ1=243 :rem 2Ø8
1717 IFMØ=1ØTHENJ1=273 :rem 253
1719 IFMØ=11THENJ1=3Ø4 :rem 251
1721 IFMØ=12THENJ1=334 :rem 248
1723 IFY/4<>INT(Y/4)THENGOTO173Ø :rem 189
1725 IF(Y/1ØØ=INT(Y/1ØØ))AND(Y/4ØØ<>INT(Y/4ØØ))THE
    NGOTO173Ø :rem 159
1727 IF(Y/4=INT(Y/4))AND(MØ>=3)THENJ1=J1+1:rem 175
173Ø RETURN :rem 171
2ØØØ DATA "{2 SPACES}JANUARY"," FEBRUARY",
    "{4 SPACES}MARCH","{4 SPACES}APRIL","
    {6 SPACES}MAY" :rem 36
2Ø1Ø DATA "{5 SPACES}JUNE","{5 SPACES}JULY",
    "{3 SPACES}AUGUST","SEPTEMBER","{2 SPACES}OCTO
    BER" :rem 229
2Ø2Ø DATA " NOVEMBER"," DECEMBER" :rem 39
2Ø3Ø DATA "{4 SPACES}{RVS}SUNDAY{OFF}","{4 SPACES}
    MONDAY","{3 SPACES}TUESDAY"," WEDNESDAY",
    "{2 SPACES}THURSDAY" :rem 9Ø
2Ø4Ø DATA "{4 SPACES}FRIDAY","{2 SPACES}SATURDAY"
    :rem 192
```

#### Program 4. Yearly Calendar Printer

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
3 SYS65517:A=PEEK(781):IFA=4ØTHENPOKE53281,1 :rem 113
5 OPEN1,4:DIMW4(3):GOSUB151Ø:I=1:J=2 :rem 128
7 PRINT#1,CHR$(14)SPC(13)"HAPPY NEW YEAR ";Y:PRINT
    #1 :rem 38
1Ø PRINT#1,CHR$(14)SPC(8)"JANUARY"SPC(13)"FEBRUARY
    " :rem 49
```

```

12 GOSUB1009:GOSUB1000:GOSUB1012:C0=6:GOSUB1019:GO
    SUB1000:GOSUB1022                                :rem 69
15 M0=I:M8=1:GOSUB292:GOSUB20:GOTO35                :rem 228
20 D=1:GOSUB1050:W2=8-D1:W4(M8)=W2+1:GOSUB321
                                                    :rem 123
22 IFD1=7THENGOTO30                                  :rem 167
25 FORD=2TOW2:GOSUB1050:GOSUB331:NEXTD              :rem 187
30 RETURN                                             :rem 67
35 GOSUB990:M0=J:M8=2:GOSUB292:GOSUB20            :rem 105
44 W3=1                                               :rem 96
45 M0=I:M8=1:GOSUB292:GOSUB200                    :rem 60
46 IFW4(2)=9THENPRINT#1,CHR$(15)SPC(1);            :rem 20
50 GOSUB991:M0=J:M8=2:GOSUB292:GOSUB200          :rem 151
56 IFW3=1ANDW4(1)>9THENPRINT#1,CHR$(15)SPC(0);
                                                    :rem 223
57 IFW3=1ANDW4(1)<10THENPRINT#1,CHR$(15)SPC(1);
                                                    :rem 7
58 IFW3=4ANDW4(2)>30THENPRINT#1,CHR$(15)SPC(0);
                                                    :rem 15
65 W3=W3+1                                           :rem 24
70 IFW3<C0THENGOTO45                                 :rem 0
71 PRINT#1," "                                       :rem 185
72 IFI=1THENGOTO86                                   :rem 133
73 IFI=3THENGOTO96                                   :rem 137
74 IFI=5THENGOTO106                                  :rem 180
75 IFI=7THENGOTO116                                  :rem 184
76 IFI=9THENGOTO126                                  :rem 188
77 IFI=11THENGOTO199                                 :rem 240
86 PRINT#1,CHR$(14)SPC(9)"MARCH"SPC(16)"APRIL"
                                                    :rem 171
88 I=3:J=4:GOTO12                                    :rem 244
96 PRINT#1,CHR$(14)SPC(10)"MAY"SPC(17)"JUNE"
                                                    :rem 11
98 I=5:J=6:GOTO12                                    :rem 249
106 PRINT#1,CHR$(14)SPC(9)"JULY"SPC(16)"AUGUST"
                                                    :rem 14
108 I=7:J=8:GOTO12                                   :rem 37
116 PRINT#1,CHR$(14)SPC(7)"SEPTEMBER"SPC(13)"OCTOB
    ER"                                               :rem 162
118 I=9:J=10:GOTO12                                  :rem 81
126 PRINT#1,CHR$(14)SPC(7)"NOVEMBER"SPC(13)"DECEMB
    ER"                                               :rem 131
128 I=11:J=12:GOTO12                                 :rem 125
199 PRINT#1,CHR$(15)SPC(1):CLOSE1:END              :rem 194

```

# Therapy

---

Steven Rubio

*It'll never replace Freud, but "Therapy" may just cure your blues. For the Commodore 64.*

**T**here is something fascinating about carrying on a seemingly reasonable conversation with a machine. I still remember the thrill when I first learned my computer could ask me a question (WHAT IS YOUR NAME?) and remember the answer. That thrill is what prompted me to write "Therapy."

## A Smarter Therapist

Therapy is a program that illustrates some of the basics of artificial intelligence. *Eliza*, the computer psychotherapist, is probably the most famous of all artificial intelligence programs. Written in LISP by Joseph Weizenbaum in 1966, *Eliza* has been run on computers of all sizes and types (including home computers programmed in BASIC) ever since.

Why another version of *Eliza*? When written in BASIC, *Eliza* is extremely slow, taking as much as ten seconds to respond to your comments. It seemed to me that *Eliza* was a bit stand-offish for a therapist; sometimes, it seemed rather dumb too.

The problem is that *Eliza* tries for too much. BASIC searches for 50 keywords and 100 responses slow *Eliza* down—and in its attempt to give meaningful responses to *all* of the user's statements, it consumes a lot of time for only occasional (if spectacular) success.

That is all right, since Weizenbaum never intended the program to substitute for actual therapy. But when showing off your computer to friends at your next get-together, it might be fun to have a program to demonstrate your machine's "intelligence" beyond a shadow of a doubt. Therapy is such a program. Load it up and run it the next time someone asks you what your computer can *really* do.

## Therapy

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

100 PRINTCHR$(142)CHR$(8)CHR$(30):POKE53281,0:POKE
    53280,0:GOSUB1230:POKE198,0           :rem 188
105 Q=0:QD=0                             :rem 144
110 PRINTCHR$(147);"HELLO. I'M DR. ROM. WHAT'S YOU
    R NAME?"                               :rem 40
115 GOSUB1160:A$=P1$:PRINT                :rem 39
120 PRINT"IN ONE WORD, ";A$;"":PRINT"WHAT IS YOUR
    PROBLEM?":GOSUB1160:B$=P1$           :rem 14
130 PRINT:PRINTB$;"...?":PRINT:PRINT"CAN YOU TELL
    {SPACE}ME MORE?"                     :rem 108
140 GOSUB1160:GOSUB900                   :rem 48
150 PRINT:PRINT"I UNDERSTAND ";B$;" IS DIFFICULT":
    PRINT"FOR YOU."                       :rem 226
160 GOSUB1160:IFP1$="NO"THENPRINT"MAYBE I'M NOT QU
    ITE UNDERSTANDING..."               :rem 111
170 PRINT:PRINT"CAN YOU BE MORE SPECIFIC? HOW IS":
    PRINTB$;" A PROBLEM FOR YOU?"        :rem 233
180 GOSUB1160:GOSUB900                   :rem 52
190 PRINT:PRINT"HOW DOES THIS MAKE YOU FEEL, ";A$;
    "?":GOSUB1160:C$=P1$                 :rem 73
200 PRINTCHR$(147)                       :rem 13
205 PRINT"SO WHAT YOU'RE SAYING, ";A$;"":PRINT"IS
    THAT YOUR PROBLEM WITH ";B$         :rem 80
210 PRINT"IS MAKING YOU FEEL ";C$;".":GOSUB1160
                                           :rem 161
220 PRINT:PRINT"CAN YOU ELABORATE ON YOUR FEELINGS
    ?":GOSUB1160:GOSUB900               :rem 215
230 PRINT:PRINT"HAS THIS BEEN A PROBLEM FOR YOU BE
    FORE? (YES OR NO)":GOSUB1160       :rem 133
240 IFP1$<>"NO"THEN260                   :rem 236
250 PRINT"I SEE. THEN THIS NEW SITUATION MUST BE
    {2 SPACES}DIFFICULT FOR YOU.":GOTO320 :rem 81
260 PRINT:PRINT"DID YOU ALSO FEEL ";C$;" THEN?"
                                           :rem 216
270 GOSUB1160:PRINT"TELL ME MORE."      :rem 16
280 GOSUB1160:GOSUB900                   :rem 53
290 PRINTCHR$(147)"I THINK WE HAVE SOMETHING HERE.
    DO YOU{2 SPACES}SEE A PATTERN?"     :rem 236
300 GOSUB1160:PRINT:PRINT"GO ON..."   :rem 106
310 GOSUB1160:PRINT:PRINT"THIS SOUNDS DIFFICULT FO
    R YOU.":GOSUB1160                   :rem 240
320 PRINT:PRINT"DO YOU HAVE A PLAN TO DEAL WITH TH
    IS{4 SPACES}CURRENT SITUATION?"     :rem 156
330 PRINT"YES OR NO.":GOSUB1160        :rem 70
340 IFP1$<>"YES"THEN350                   :rem 65
343 PRINT"DO YOU THINK THIS PLAN WILL BE":PRINT"SU
    CCESSFUL?":GOTO360                   :rem 241

```

### 3: Applications

---

```
350 PRINT:PRINT"WHY DON'T YOU MAKE A LIST OF POSSI
BLE{3 SPACES}SOLUTIONS, THEN." :rem 107
360 GOSUB1160:GOSUB900 :rem 52
370 FORT=1TO500:NEXTT:PRINTCHR$(147) :rem 253
380 PRINT"OKAY, WHAT SINGLE WORD BEST DESCRIBES"
:rem 192
385 PRINT"HOW YOU ARE FEELING RIGHT NOW?" :rem 223
390 GOSUB1160:D$=P1$:PRINT:PRINTD$;"...?" :rem 224
400 GOSUB1160:GOSUB900:PRINT :rem 246
410 PRINT"I'M THINKING OF DOING SOMETHING HERE.
{3 SPACES}LET'S TRY SOME WORD"; :rem 142
430 PRINT" ASSOCIATION":PRINT"AND SEE WHERE IT LEA
DS US." :rem 183
440 PRINT"WHAT DO YOU THINK(YES OR NO)?":GOSUB1160
:rem 236
450 IFP1$="YES"THEN490 :rem 11
460 PRINT:PRINT"YOU SEEM TO BE HAVING SOME PROBLEM
S WITHTHIS." :rem 122
470 PRINT"CAN YOU TELL ME ABOUT IT?":GOSUB1160:IFP
1$="NO"THEN840 :rem 46
480 PRINT:PRINT"I REALLY THINK A WORD ASSOCIATION
{SPACE}WOULD BE USEFUL RIGHT NOW." :rem 4
490 PRINT:PRINT"LET'S DO IT." :rem 242
500 PRINT"I'LL SAY A WORD. YOU SAY THE FIRST WORD
{SPACE}THAT COMES TO YOUR MIND." :rem 133
510 REM ***WORD ASSOCIATION*** :rem 239
520 FORT=1TO5000:NEXTT:PRINTCHR$(147);"DOG":PRINT:
GOSUB1160 :rem 204
530 PRINT:PRINT"DRINK":PRINT:GOSUB1160 :rem 241
540 PRINT:PRINT"HOME":PRINT:GOSUB1160:E$=P1$
:rem 40
550 PRINT:PRINTB$:PRINT:GOSUB1160:F$=P1$ :rem 35
560 PRINT:PRINT"FEELINGS":PRINT:GOSUB1160 :rem 201
570 PRINT:PRINT"FUN":PRINT:GOSUB1160:G$=P1$
:rem 237
580 PRINT:PRINT"MOM":PRINT:GOSUB1160:I$=P1$
:rem 240
590 PRINT:PRINTC$:PRINT:GOSUB1160:J$=P1$ :rem 44
600 FORT=1TO1000:NEXTT:PRINTCHR$(147) :rem 37
610 PRINT"I NOTICED WHEN I SAID HOME":PRINT"THAT Y
OU SAID ";E$;". " :rem 39
620 PRINT"DOES THIS SOMEHOW REFLECT HOW YOU FEEL
{2 SPACES}ABOUT YOURSELF?" :rem 45
630 PRINT"YES OR NO":GOSUB1160:IFP1$<>"YES"THEN650
:rem 2
640 PRINT:PRINT"IN WHAT WAY?":GOSUB1160:GOSUB900
:rem 2
650 PRINT:PRINT"HOW DOES THIS RELATE TO YOUR PROBL
EM":PRINT"WITH ";B$ :rem 44
```



```

660 GOSUB1160:GOSUB900:PRINT:PRINT"WHEN I SAID ";B
    $;" YOU SAID ";F$           :rem 136
670 PRINT"WHAT DO YOU THINK THIS MEANS?":GOSUB1160
    :GOSUB900                   :rem 112
680 PRINT:PRINT"ARE YOU DISTRESSED? DO YOU WANT A
    {7 SPACES}TISSUE?":GOSUB1160 :rem 237
690 IFP1$<>"YES"THEN710         :rem 73
700 PRINT"HERE.":FORT=1TOL000:NEXTT :rem 206
710 PRINT:PRINT"IT'S INTERESTING THAT WHEN I SAID
    {SPACE}FUN,{2 SPACES}YOU SAID ";G$ :rem 57
720 GOSUB1160:GOSUB900:PRINTCHR$(147);"HMMM..."
    :rem 110
730 PRINT:PRINT"IT SEEMS TO ME, ";A$;" " :rem 248
735 PRINT"THAT THIS ALL TIES IN TO YOUR PROBLEM"
    :rem 129
740 PRINT"WITH ";B$           :rem 73
750 GOSUB1160:GOTO770         :rem 245
760 REM ***DREAMS***         :rem 57
770 PRINT:PRINT"LET'S TRY A DIFFERENT":PRINT"APPRO
    ACH, ";A$                 :rem 145
780 PRINT"TELL ME ABOUT ONE OF YOUR DREAMS.":GOSUB
    1160:GOSUB1040:IFQD=1THEN840 :rem 246
790 PRINT:PRINT"HOW WOULD YOU DESCRIBE YOUR FEELIN
    GS{4 SPACES}IN THE DREAM?" :rem 171
795 GOSUB 1160               :rem 237
800 PRINT:PRINT"DID THE DREAM HAVE ANYTHING TO DO
    {SPACE}WITH{2 SPACES}";I$   :rem 235
810 GOSUB1160:FORT=1TOL000:NEXTT :rem 245
820 REM ***ALL DONE***       :rem 121
830 PRINT:PRINT"I THINK WE'RE MOVING IN A
    {15 SPACES}GOOD DIRECTION.":PRINT :rem 187
840 PRINT"WE'VE DISCUSSED YOUR PROBLEM WITH":PRINT
    B$;" AND HOW THIS MAKES YOU :rem 255
850 PRINT"FEEL ";C$;" "      :rem 230
860 PRINT"AND DISCUSSED SOME POSSIBLE SOLUTIONS."
    :rem 124
870 PRINT:PRINT"I SEE YOUR TIME IS UP.{18 SPACES}S
    EE YOU NEXT WEEK."         :rem 189
880 END                       :rem 119
890 REM ***KEYWORDS***       :rem 249
900 IFQ>0THENRETURN          :rem 246
910 FORJ=1TOLEN(P1$)-5       :rem 19
920 IFMID$(P1$,J,5)<>" FUN "THEN930 :rem 103
925 PRINT:PRINT"WHAT ARE YOUR FEELINGS ABOUT FUN?"
    :GOTO950                   :rem 148
930 NEXTJ                    :rem 37
940 RETURN                   :rem 125
950 GOSUB1160:Q=1:PRINT:PRINT"THESE FEELINGS SEEM
    {SPACE}IMPORTANT."         :rem 141
960 GOSUB1160:RETURN         :rem 1

```

### 3: Applications

---

```
1040 REM ***DREAM KEYWORD SEARCH***           :rem 233
1050 FORJ=1TOLEN(P1$)-7                       :rem 65
1060 IFMID$(P1$,J,7)=" DON'T "THEN1120       :rem 243
1070 NEXTJ                                     :rem 81
1080 FORJ=1TOLEN(P1$)-6                       :rem 67
1090 IFMID$(P1$,J,6)=" DONT "THEN1120       :rem 206
1100 NEXTJ                                     :rem 75
1110 RETURN                                   :rem 163
1120 PRINTCHR$(147)"WHY DO YOU SUPPOSE THAT IS?":G
      OSUB1160:GOSUB900                       :rem 27
1130 PRINT"THIS MAY BE SOMETHING THAT WE'LL WANT"
                                             :rem 176
1140 PRINT"TO DISCUSS LATER. WE MAY FIND THAT IT"
                                             :rem 112
1150 PRINT"RELATES TO YOUR PROBLEM WITH ";B$:QD=1:
      RETURN                                   :rem 223
1160 REM ***COMMODORE PUNCTUATION INPUT***   :rem 55
1170 P1$=""                                   :rem 239
1180 GETP2$:IFP2$=""THEN1180                 :rem 57
1190 PRINTP2$;                               :rem 57
1200 IFP2$=CHR$(13)THENRETURN                :rem 250
1210 P1$=P1$+P2$                             :rem 28
1220 GOTO1180                                 :rem 200
1230 REM ***INTRODUCTION***                 :rem 72
1240 PRINTCHR$(147);TAB(15)"THERAPY"         :rem 108
1250 PRINT:PRINT"WOULD YOU LIKE AN INTRODUCTION (Y
      /N)"                                     :rem 101
1260 GETQ$:IFQ$<>"Y"ANDQ$<>"N"THEN1260       :rem 191
1270 IFQ$="N"THENRETURN                       :rem 172
1280 PRINTCHR$(147);"WELCOME TO YOUR THERAPY SESSI
      ON. DR. ROM";                           :rem 31
1285 PRINT"WILL BE WITH YOU IN A ";          :rem 172
1290 PRINT"MOMENT. WHILE YOU ARE WAITING, HERE ARE
      SOME HELPFUL"                           :rem 104
1300 PRINT"SUGGESTIONS ON HOW TO GET THE MOST OUT
      {2 SPACES}OF YOUR THERAPY SESSION."     :rem 109
1305 PRINT:PRINT                             :rem 29
1310 PRINT"AS WITH MOST THINGS IN LIFE, WITH
      {7 SPACES}THERAPY, THE MORE YOU ";      :rem 42
1320 PRINT"PUT IN, THE MORE{2 SPACES}YOU GET OUT.
      {SPACE}YOU MAY FIND IT FUN ";          :rem 48
1322 PRINT "TO TRY AND TRIP ";               :rem 136
1330 PRINT"UP THE DOCTOR; MAKE FUN OF HIS GRAMMAR,
      OR INSULT HIM MERCILESSLY."           :rem 175
1340 PRINT"{DOWN}HOWEVER, EVEN THOUGH THIS IS A PA
      RLOR{3 SPACES}GAME, YOU MAY STILL FIND ";
                                             :rem 230
1350 PRINT"YOURSELF HAVINGINTERESTING, AND EVEN IM
      PORTANT,"                               :rem 51
```

```
1360 PRINT"INSIGHTS. THIS WILL ONLY HAPPEN IF YOU
      {2 SPACES}TRY YOUR BEST TO UTILIZE ";:rem 172
1370 PRINT"THIS SESSION ASAN ENJOYABLE WAY TO MULL
      OVER THE" :rem 159
1380 PRINT"PROBLEMS AND PEEVES OF LIFE." :rem 127
1390 PRINT:PRINT:PRINTCHR$(18)"HIT ANY KEY TO CONT
      INUE" :rem 165
1400 POKE198,0:WAIT198,1 :rem 96
1410 PRINTCHR$(147):PRINT:PRINT"I SEE THE DOCTOR I
      S IN NOW." :rem 58
1420 PRINT:PRINT:PRINT"TO TALK TO DR. ROM, JUST TY
      PE IN YOUR" :rem 228
1430 PRINT"RESPONSE; AND HIT ";CHR$(18);"RETURN";C
      HR$(146);" WHEN YOU ARE" :rem 254
1440 PRINT"FINISHED.":PRINT:PRINT:PRINT"ENJOY YOUR
      THERAPY SESSION." :rem 238
1450 PRINTSPC(240);CHR$(18);"HIT ANY KEY TO BEGIN"
      :rem 87
1460 POKE198,0:WAIT198,1:RETURN :rem 128
```

# The Indexer

Dan Carmichael

*Designed to provide an indexing system for articles in COMPUTE!'s Gazette, this program can be used for a variety of purposes. It runs on any VIC-20 and the Commodore 64.*

If you're like many computer hobbyists, you keep your back issues of *COMPUTE!* and *COMPUTE!'s Gazette*. There's a wealth of reference material in each issue. The only problem is remembering just which issue contains that article you so desperately need.

"The Indexer" is a small data base program that allows you to keep an index of any articles or books that are of interest to you. It stores such information as the magazine (or book) name, subject matter, article title, month and year of issue, page number, and type of computer the article applies to. It can also search for that article by subject, article name, magazine name, and type of computer.

## Storing Data in the Program

The Indexer is *machine-independent*. In other words, it does not rely on a peripheral device such as a tape cassette or disk drive. Information is read into the program from DATA statements and is stored within the program in an array. If you study it carefully, you'll see some useful array and table look-up techniques.

Each DATA statement you enter must include the following six elements, in order, and each entry should be separated from the others by a comma.

DATA <sup>last name</sup> magazine <sup>1st name</sup> name, <sup>phone#</sup> article title, <sup>address</sup> subject, <sup>city+state</sup> month, year, page number, <sup>zip code</sup> type of computer

Be careful when entering the DATA statements. A misplaced or forgotten comma will cause errors when the program is run. Be sure not to use commas or colons when typing in the article titles.

## How to Use The Indexer

Type in the program (be careful with all cursor control characters) and save it to tape or disk before running. The five DATA statements at the end of the program are optional, in-

cluded only as examples of the DATA statement format. If you wish to begin your own data base, you can replace the DATA statements from line 901 on.

Each time you add or delete DATA statements from the program, change the value of the variable N in line 900. This variable represents the exact number of DATA statements included. If you number consecutively, beginning at line 901, it will be easy to figure out how many DATA statements there are. Anytime you update your program, you should save a copy to tape or disk.

Once the program is running, you'll be prompted to select the target of your search. You can search for article subject, article name, name of magazine, or type of computer. To start the search, press the indicated function key. You'll then be asked for the target of your search. Just enter the search keyword, press RETURN, and the program will search the table for you.

If you're using the program with an unexpanded VIC, memory may become a problem as you add DATA statements. String arrays—the kind used in this program to store data—use a lot of memory. In addition, each of the DATA statements takes up six bytes, plus one byte per character. If you accumulate a lot of data, an expander cartridge will come in handy.

### **Tips for Data Entry**

*Subject:* Your searches will usually be done by article subject, so keep this category as broad as possible. For example, let's say you want to index various articles about game paddles. Enter all of them with the subject "paddles," even if some are about drawing with paddles and others about using them in games. That way, when you enter "paddles" as the target of your search, the index of *all* articles on this subject will be displayed.

*Spelling:* Watch your spelling, and be consistent with your subject category names. For example, don't enter one subject as "paddle" and another as "paddles." The computer will see these as two completely different categories.

*Memory:* As stated before, The Indexer can use a lot of memory, so you might want to abbreviate article titles. For example, this article could be entered as "Indexer" or as "Ind."

Although this program was written as an article index, it

### 3: Applications

can be adapted for other uses. The data base has six elements and can search by any of four variables. It could be easily adapted for other uses, such as a birthday reminder or an electronic phone book. The applications are up to you.

#### The Indexer for VIC and 64

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
1 REM REMEMBER TO CHNG{2 SPACES}"N", LINE #900 WHE
N{3 SPACES}ADDING/DELETING DATA{2 SPACES}STATEME
NTS :rem 63
10 PRINT"{CLR}{BLU}{DOWN} DO YOU WISH TO SEE THE M
ENU?":PRINT"{DOWN} (PRESS Y OR N)" :rem 215
20 POKE53280,3:POKE53281,1:POKE646,6 :rem 38
21 GETY$:IFY$=""THEN21 :rem 25
25 IFY$="Y"THEN800 :rem 20
29 PRINT"{3 DOWN}{5 SPACES}LOADING DATABASE...PLEA
SE WAIT.":GOSUB900 :rem 156
30 S=0:PRINT"{CLR}{DOWN} ENTER SEARCH"TAB(33)"
{GRN}PRESS":PRINT"{BLU} ARGUMENT"TAB(33)"{GRN}F
-KEY" :rem 92
35 PRINT"{BLU}{2 DOWN} telephone number ARTICLE SUBJECT"TAB(35)"
{GRN}1" :rem 177
36 PRINT"{BLU}{DOWN} first name ARTICLE NAME"TAB(35)"{GRN}3"
:rem 180
37 PRINT"{BLU}{DOWN} last name MAGAZINE NAME"TAB(35)"{GRN}5"
:rem 255
38 PRINT"{BLU}{DOWN} zip code TYPE OF COMPUTER"TAB(35)"
{GRN}7" :rem 219
39 PRINT"{BLU}{DOWN} END PROGRAM"TAB(35)"{GRN}8"
:rem 134
40 GETX$ :rem 192
41 IFX$="{F1}"THENS=3 :rem 104
42 IFX$="{F3}"THENS=2 :rem 105
43 IFX$="{F5}"THENS=1 :rem 106
44 IFX$="{F7}"THENS=6 :rem 113
45 IFX$="{F8}"THENPRINT"{CLR}{DOWN} END PROGRAM":C
LR:END :rem 64
46 IFS=0THEN40 :rem 76
60 PRINT"{CLR}{DOWN} ENTER SUBJECT OF SEARCH:
{2 DOWN}":INPUTS$ :rem 81
65 FORZ=1TON:IFA$(Z,S)=S$THENGOSUB300 :rem 89
70 NEXTZ :rem 0
75 PRINT"{CLR}{DOWN} END OF DATA OR{DOWN}":PRINT"
{SPACE}SUBJECT NOT FOUND{DOWN}" :rem 178
76 PRINT" (CHECK SPELLING){2 DOWN}" :rem 109
77 GOSUB600:GOTO30 :rem 90
300 PRINT"{CLR}{GRN} SUBJECT FOUND:{2 DOWN}":PRINT
"{YEL}MAGAZINE:{BLU}":PRINTA$(Z,1) :rem 127
```

```

305 PRINT "{DOWN}{YEL}{ARTICLE}{BLU}":PRINTA$(Z,2)
                                     1st name
                                     1st name
                                     :rem 170
310 PRINT "{DOWN}{YEL}{SUBJECT}{BLU}":PRINTA$(Z,3):P
    RINT "{DOWN}{YEL}{DATE}{BLU}":PRINTA$(Z,4)
                                     1st name
                                     1st name
                                     :rem 27
315 PRINT "{DOWN}{YEL}{PAGE NO.}{BLU}":PRINTA$(Z,5)
                                     city state
                                     :rem 146
320 PRINT "{DOWN}{YEL}{COMPUTER}{BLU}":PRINTA$(Z,6)
                                     zip code
                                     :rem 22
330 GOSUB600:RETURN                  :rem 198
600 PRINT "{GRN}{DOWN} (PRESS RETURN){BLU}":rem 115
601 GETY$:IFY$=" THEN601            :rem 129
602 RETURN                          :rem 120
800 PRINT "{CLR}{GRN}{DOWN} RECORD FORMAT":PRINT "
    {DOWN}{BLU}1) MAGAZINE NAME" last name :rem 239
805 PRINT "{DOWN}2) NAME OF ARTICLE 1st name :rem 114
810 PRINT "{DOWN}3) SUBJECT OF ARTICLE":PRINT " 1st name
    {DOWN}4) MONTH.YEAR" address :rem 222
815 PRINT "{DOWN}5) PAGE NO.":PRINT "{DOWN}6) TYPE O
    F COMPUTER" zip code :rem 135
820 PRINT "{YEL}{2 DOWN} SEPARATE EACH ENTRY BY A C
    OMMA{BLU}" :rem 139
830 GOSUB600                        :rem 177
840 PRINT "{CLR}{DOWN}WHEN PROMPTED TO ENTER SEARCH
    ARGUMENT" :rem 232
850 PRINT "{DOWN}PRESS F KEY FOR DESIRED FUNCTION"
                                     :rem 136
860 GOSUB600                        :rem 180
870 PRINT "{CLR}{DOWN}WHEN PROMPTED TO ENTER SUBJEC
    T OF" :rem 119
880 PRINT "{DOWN}SEARCH, ENTER NAME, THEN PRESS RET
    URN." :rem 249
885 GOSUB600                        :rem 187
890 GOTOL0                          :rem 59
900 N=5:DIMA$(N,6):FORR=1TON:FORC=1TO6:READA$(R,C)
    :NEXTC:NEXTR:RETURN              :rem 190
901 DATAGAZETTE,DO YOU NEED A CASSETTE RECORDER,CA
    SSETTE,7.83,28,ALL               :rem 178
902 DATAGAZETTE,COMMODORE64 VIDEO UPDATE,VIDEO,7.8
    3,40,64                          :rem 3
903 DATAGAZETTE,INSIDE VIEW JIMMY HUEY,INTERVIEW,7
    .83,49,ALL                       :rem 49
904 DATAGAZETTE,SKYDIVER,GAME,7.83,52,ALL :rem 66
905 DATAGAZETTE,COMPUTING FOR KIDS ADVENTURES,EDUC
    ATION,7.83,34,ALL                :rem 29

```

□ □ □ □ □

□ □ □ □ □



# Chapter 4

---

# Graphics and Sound

□ □ □ □ □

□ □ □ □ □

# VIC Hi-Res Sketchpad

Anthony T. Beville

"Hi-Res Sketchpad" lets you create interesting drawings and pictures, using your joystick to draw on a 128 X -128 bit screen. For the unexpanded VIC.

**W**ith this program, you will be able to create simple pictures using your VIC and a joystick.

When the program is run, you are asked to set screen, border, and line colors. Then a drawing window and a blinking pixel will appear. The joystick controls the pixel's movement; pressing the fire button will leave behind a line.

If you want to clear the screen, press C. When you want to quit, press Q and the screen and memory will be set back to normal.

I got the idea for a high-resolution sketch program from Paul Schatz's article "High-Resolution Plotting" in *COMPUTE!'s First Book of VIC*. Basically, the program works by filling the screen with programmable characters and then redefining them based on the joystick movement.

While this program won't let you create any complex masterpieces, it will let you have fun just doodling. If you have a printer capable of printing redefined characters, you might want to add C. D. Lane's "Printing the Screen" routine from *COMPUTE!'s First Book of VIC* to save your creation.

## Hi-Res Sketchpad for the VIC

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```
10 POKE52,22:POKE56,22:CLR:PRINTCHR$(147) :rem 187
20 INPUT"LINE COLOR(1-8)";LC:IFLC>8ORLC<1THEN20
                                     :rem 138
30 INPUT"SCREEN COLOR(1-8)";SC:IFSC>8ORSC<1THEN30
                                     :rem 57
40 INPUT"BORDER COLOR(1-8)";BC:IFBC>8ORBC<1THEN40
                                     :rem 6
50 LC=LC-1:BC=BC-1:POKE36879,SC*16-8+BC :rem 223
60 POKE36869,208:POKE648,22:FORJ=217TO228:POKEJ,15
  0:NEXT:FORJ=229TO250:POKEJ,151:NEXT :rem 227
70 POKE36864,11:POKE36865,34:POKE36866,144:POKE368
  67,32:POKE36869,222 :rem 181
80 FORI=0TO255:POKE5632+I,I:POKE38400+I,LC:NEXT
                                     :rem 118
85 X=64:Y=64 :rem 166
```

## 4: Graphics and Sound

---

```
90 FORI=6144TO8191:POKEI,0:NEXT :rem 162
100 GOSUB500:DX=0:DY=0:IFJ0THENDX=1 :rem 169
110 IFJ1THENDY=1 :rem 214
120 IFJ2THENDX=-1 :rem 4
130 IFJ3THENDY=-1 :rem 7
140 X=DX+X:Y=DY+Y :rem 58
150 A=INT(Y/8)*16+INT(X/8) :rem 199
160 B=(Y/8-INT(Y/8))*8:C=6144+8*A+B :rem 118
170 D=7-(X-INT(X/8)*8):POKEC,PEEK(C)OR(2↑D) :rem 7
180 IFFB=0THENPOKEC,PEEK(C)-(2↑D) :rem 245
190 A$="":GETA$ :rem 255
200 IFA$="C"THEN90 :rem 226
210 IFA$="Q"THEN220 :rem 28
215 GOTO100 :rem 98
220 POKE36864,5:POKE36865,25:POKE36866,150:POKE368
67,46:POKE36869,208:POKE36879,27 :rem 202
230 PRINTCHR$(147):POKE52,30:POKE56,30:END:rem 227
500 POKE37154,127:P=PEEK(37152)AND128:J0=-(P=0)
:rem 103
510 POKE37154,255:P=PEEK(37151):J1=-(PAND8)=0)
:rem 88
520 J2=-(PAND16)=0):J3=-(PAND4)=0) :rem 157
530 FB=-(PAND32)=0):RETURN :rem 59
```

# SDA: A Sprite Design Aid for the Commodore 64

---

Karl Dittman

*“Sprite Design Aid” is a useful graphics utility that allows you to create sprites by drawing on the screen with your joystick. The resulting sprites can then be saved, and later modified, if desired.*

“**S**prite Design Aid” lets Commodore 64 users create complex sprites by drawing directly on the screen. The program then reads the screen and calculates the values that define the sprite. The sprite appears on the screen, and the programmer has the option of saving the data for later use or modification.

## Using Sprite Design Aid

After you load and run the program and review the instruction screen, press any key to enter the main program. A red box with a cursor will appear, and you are ready to design your sprite.

Draw your sprite using a joystick plugged into port 2. To erase any part of your sprite, press the fire button as you move the cursor.

When you're finished, press any key. The sprite DATA statements will be calculated and displayed and can then be saved on disk under any filename you assign.

The screen will be saved in a sequential file using the name you assign followed by “SC”. The program uses the “SC” file to recreate the sprite whenever you wish to make changes; it's transparent to the user. One word of warning: When using a second disk to save your sprites, be *sure* that it has a different disk ID than your main program disk.

If you feel that you will not want to change your sprite, then the “SC” file can be deleted. Once deleted, however, your sprite cannot be re-created by this program. If in doubt, leave it intact.

### Using the DATA

To use sprite DATA saved on disk, include the following statements in your program (preferably at the beginning):

```
10 DIM S1(62) : REM ARRAY USED TO STORE THE
    SPRITE DATA STATEMENTS
20 OPEN 2,8,2,"SPRITE": REM (YOUR ASSIGNED FILE
    NAME)
30 FOR S=0 TO 62 : INPUT #2,S1(S) : NEXT S
40 CLOSE 2
```

If more than one sprite is used, then DIMension arrays S2, S3, and so on as required. You will also have to change the filename in line 20 and the array name in line 30 for every array used to contain sprite DATA.

Additional statements can be added between lines 30 and 40 for each sprite used. After the array has been loaded, it is used instead of the DATA statements and corresponding READ.

### How the Program Works

#### Line(s)

- 15-115 Read screen and assign sprite DATA values.
- 120-128 Capture screen in array SS for storage on disk.
- 200-275 Display sprite as it will appear in program.
- 280-299 Disk file SAVE routine.
- 500-730 Joystick movement and sprite design.
- 800-1000 Subroutine to print instructions.
- 2000-2075 Subroutine to load array SS to recall previously designed sprite for modification.
- 3000-3110 Subroutine to bring previously saved sprite back to the screen.

### SDA, A Sprite Design Aid for the Commodore 64

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
1 POKE53280,1:POKE53281,1:GOSUB800 :rem 218
2 DIMS(64),SS(504):PRINT"{CLR}":GOSUB2000 :rem 240
4 PRINT"{CLR}":BD=102:FORI=1028TO1868STEP40:POKEI,
    BD:POKEI+54272,10:NEXT :rem 255
5 FORI=1908TO1933:POKEI,BD:POKEI+54272,10:NEXT
    :rem 50
6 FORI=1933TO1052STEP-40:POKEI,BD:POKEI+54272,10:N
    EXT :rem 246
9 FORI=1052TO1028STEP-1:POKEI,BD:POKEI+54272,10:NE
    XT :rem 193
```

```

10 PRINTCHR$(144);"{HOME}": FORI=1TO21:PRINTI:NEXT
                                                                    :rem 210
12 PRINT:PRINT"PRESS ANY KEY WHEN SPRITE IS COMPLE
TE."
                                                                    :rem 133
13 IF LEFT$(A$,1)="Y" THEN GOSUB3000
                                                                    :rem 161
14 GOSUB500:DIMY3(64)
                                                                    :rem 213
15 X=1076
                                                                    :rem 201
20 FORI=XTOX-7STEP-1
                                                                    :rem 32
30 IFPEEK(I)<>32THENY=2↑(X-I)
                                                                    :rem 4
40 IFPEEK(I)<>32THEN Y2=Y2+Y
                                                                    :rem 151
50 NEXTI
                                                                    :rem 237
60 Y3(T2)=Y2:Y2=0
                                                                    :rem 195
70 T2=T2+1
                                                                    :rem 12
80 C=C+1
                                                                    :rem 135
90 IFC<>3THENX=X+8:GOTO 115
                                                                    :rem 113
100 X=X+24
                                                                    :rem 15
105 LN=LN+1
                                                                    :rem 99
108 PRINT"{HOME}"
                                                                    :rem 125
109 FOR I=1TO22:PRINT"{DOWN}";:NEXT:PRINT"I'M WORK
ING-----PLEASE WAIT-----"
                                                                    :rem 228
110 C=0
                                                                    :rem 66
115 IF X<1932THEN20
                                                                    :rem 26
120 T2=0:X=1069
                                                                    :rem 37
122 FORI=XTOX+23
                                                                    :rem 229
123 T2=T2+1
                                                                    :rem 59
124 IFPEEK(I)<>32THENSS(T2)=1
                                                                    :rem 222
125 IFPEEK(I)=32THENSS(T2)=0
                                                                    :rem 161
127 NEXT I
                                                                    :rem 34
128 X=X+40 : IFT2<504THEN122
                                                                    :rem 255
129 PRINT"{CLR}"
                                                                    :rem 0
130 PRINT "DATA ";:FOR I=0TO10:PRINT Y3(I);:NEXT:P
RINT
                                                                    :rem 253
135 PRINT "DATA ";:FOR I=11TO20:PRINT Y3(I);:NEXT:
PRINT
                                                                    :rem 53
140 PRINT "DATA ";:FOR I=21TO30:PRINT Y3(I);:NEXT:
PRINT
                                                                    :rem 51
145 PRINT "DATA ";:FOR I=31TO40:PRINT Y3(I);:NEXT:
PRINT
                                                                    :rem 58
150 PRINT "DATA ";:FOR I=41TO50:PRINT Y3(I);:NEXT:
PRINT
                                                                    :rem 56
155 PRINT "DATA ";:FOR I=51TO62:PRINT Y3(I);:NEXT:
PRINT
                                                                    :rem 65
200 PRINT
                                                                    :rem 31
206 PRINT"THE SPRITE YOU HAVE JUST DESIGNED WILL
{2 SPACES}LOOK LIKE THIS."
                                                                    :rem 195
210 V=53248
                                                                    :rem 44
220 POKEV+21,0
                                                                    :rem 3
230 T=13:M=T*64
                                                                    :rem 54
240 POKE V,160:POKEV+1,200
                                                                    :rem 183

```

## 4: Graphics and Sound

---

```
250 POKEV+23,1:POKEV+29,1           :rem 187
260 POKEV+39,2                       :rem 18
265 POKE 2040,T                       :rem 18
270 POKEV+21,1                       :rem 9
275 FORI=0TO62:POKE832+I,Y3(I):NEXT  :rem 139
280 PRINT:PRINT"DO YOU WISH TO SAVE THE SPRITE ON
   {SPACE}DISK{2 SPACES}(Y OR N)": INPUT K$
                                           :rem 33
281 IF LEFT$(K$,1)="N" THEN GOTO 299  :rem 113
282 PRINT"{CLR}"B$:PRINT"ENTER FILE NAME OF SAVED
   {SPACE}SPRITE ": INPUT N$:N2$=N$     :rem 216
283 PRINT "DO YOU WISH TO REPLACE AN EXISTING FILE
   ":INPUT A$                           :rem 33
284 IF LEFT$(K$,1)="Y"THENGOSUB3400    :rem 233
285 OPEN15,8,15                       :rem 45
290 IFLEFT$(A$,1)="Y"THENOPEN 2,8,2,"@0:"+N$+"",S,W
   "                                     :rem 227
292 INPUT#15,A,B$                     :rem 181
294 IFLEFT$(A$,1)="N"THENOPEN 2,8,2,"0:"+N$+"",S,W"
   "                                     :rem 156
295 INPUT#15,A,B$                     :rem 184
296 IFA=63THENCLOSE2:CLOSE15:GOTO282  :rem 19
297 FOR I=0TO62: PRINT#2,Y3(I):NEXT    :rem 49
298 CLOSE2:CLOSE15                   :rem 97
299 FOR I=0TO62:POKE832+I,0:NEXT      :rem 155
300 GOTO 3500                         :rem 148
500 X=1524:J=56320:B=160             :rem 7
510 POKE X,81                        :rem 178
520 X2=X                              :rem 182
525 LETB=0                            :rem 48
526 S2=PEEK(56320)AND15               :rem 14
527 A=-((PEEK(56320)AND16)=0)        :rem 8
530 IFS2=6ANDPEEK(X-39)<>BDTHENX=(X-39):GOTO610
   "                                   :rem 105
540 IFS2=5ANDPEEK(X+41)<>BDTHENX=(X+41):GOTO610
   "                                   :rem 87
550 IFS2=10ANDPEEK(X-41)<>BDTHENX=(X-41):GOTO610
   "                                   :rem 136
560 IFS2=9ANDPEEK(X+39)<>BDTHENX=(X+39):GOTO610
   "                                   :rem 107
570 IFS2=7ANDPEEK(X+1)<>BDTHENX=X+1:GOTO610
   "                                   :rem 163
580 IF S2=14 THEN X=X-40:GOTO610      :rem 10
590 IF S2=13 THEN X=X+40:GOTO610      :rem 8
600 IFS2=11ANDPEEK(X-1)<>BDTHENX=X-1   :rem 194
610 IFX>1909 AND X<=1932 THEN X=X-40  :rem 175
620 IF X>1029 AND X<1052 THEN X=X+40  :rem 99
630 IF X<1069 THEN X=1069            :rem 32
640 IF X>1892 THEN X=1892            :rem 43
650 POKE X,160                       :rem 229
```



```
660 IFA<>1 THENPOKEX2+54272,A           :rem 155
670 POKEX+54272,14                       :rem 228
680 IF A=1 THENPOKEX,32                   :rem 35
690 GETA$:IFA$<>"THEN730                 :rem 153
720 GOTO 520                              :rem 105
730 RETURN                                :rem 122
800 PRINT"{CLR}{BLK}";"{12 SPACES}SPRITE DESIGN AI
D"                                         :rem 106
806 PRINT"THIS PROGRAM ALLOWS YOU TO USE A "
                                           :rem 95
808 PRINT"JOYSTICK TO DESIGN SPRITES.":PRINT
                                           :rem 93
810 PRINT"MOVE YOUR JOYSTICK IN ANY DIRECTION TO"
                                           :rem 35
811 PRINT"DRAW SPRITE DESIRED. IF YOU NEED TO "
                                           :rem 233
820 PRINT"ERASE ANY PART OF THE PICTURE THEN HOLD"
                                           :rem 226
830 PRINT"THE FIRE BUTTON DOWN AS YOU DRAW.":PRINT
                                           :rem 59
850 PRINT"PRESS RETURN WHEN YOUR SPRITE IS"
                                           :rem 207
851 PRINT"FINISHED.{2 SPACES}THE COMMODORE WILL TH
EN"                                       :rem 212
860 PRINT"GENERATE THE DATA STATEMENTS FOR THE":PR
INT"SPRITE YOU DESIGNED."               :rem 213
870 PRINT:PRINT"YOU CAN SAVE THE DATA STATEMENTS O
N"                                       :rem 213
875 PRINT"DISK BY ANSWERING 'Y' WHEN ASKED IF YOU"
                                           :rem 182
880 PRINT"WOULD LIKE TO SAVE YOUR SPRITE ON DISK."
:PRINT                                     :rem 214
890 PRINT"YOU CAN RECALL THE SPRITE AT ANOTHER
{4 SPACES}TIME WHENEVER YOU NEED TO ";:rem 220
900 PRINT"USE IT.":PRINT                 :rem 233
980 PRINTCHR$(31)"PRESS ANY KEY TO BEGIN.":rem 188
990 GETA$:IFA$="THEN990                 :rem 103
1000 RETURN                              :rem 161
2000 PRINT"WOULD YOU LIKE TO CHANGE A SPRITE NOW"
                                           :rem 149
2010 PRINT"STORED ON DISK (NO IS ASSUMED)":A$="N":
INPUTA$                                     :rem 102
2020 IF LEFT$(A$,1)="N" THEN RETURN      :rem 147
2030 PRINT"ENTER THE NAME OF THE SPRITE DISK FILE"
:INPUT N$:N2$=N$:N$=N$+"SC"             :rem 215
2040 OPEN15,8,15                          :rem 84
2050 OPEN2,8,2,"0:"+N$+",S,R"           :rem 208
2055 INPUT#15,A,B$:IFA<>62THEN2060      :rem 199
```

## 4: Graphics and Sound

---

```
2056 IFA=62THENPRINT"{CLR}{4 DOWN}FILE NOT FOUND "  
      :PRINT"{2 SPACES}PRESS ANY KEY TO START OVER  
                                          :rem 90  
2057 GETA$:IFA$=""THEN2057                :rem 191  
2058 CLR:RUN2                              :rem 17  
2060 FORI=1TO504:INPUT#2,SS(I):NEXT       :rem 166  
2070 CLOSE2:CLOSE15                       :rem 135  
2075 RETURN                                :rem 174  
3000 X=1069                                :rem 40  
3010 FORI=XTOX+23                          :rem 20  
3011 T2=T2+1                              :rem 106  
3012 IFSS(T2)=1THEN POKEI,160             :rem 170  
3015 IFSS(T2)=1THEN POKEI+54272,0        :rem 117  
3020 IFSS(T2)=0THEN POKEI,32             :rem 118  
3030 NEXT I                                :rem 78  
3040 X=X+40                                :rem 67  
3050 IFT2<504THEN3010                    :rem 165  
3100 T2=0:C=0                              :rem 161  
3110 RETURN                                :rem 165  
3400 OPEN15,8,15                          :rem 85  
3410 N$=N$+"SC"                            :rem 238  
3420 IFLEFT$(A$,1)="Y"THENOPEN 2,8,2,"@0:"+N$+",S,  
      W"                                     :rem 17  
3430 INPUT#15,A,B$                        :rem 226  
3440 IFLEFT$(A$,1)="N"THENOPEN 2,8,2,"0:"+N$+",S,W  
      "                                     :rem 200  
3450 INPUT#15,A,B$                        :rem 228  
3460 IFA=63THENCLOSE2:CLOSE15:GOTO282    :rem 63  
3470 FORI=1TO504:PRINT#2,SS(I):NEXT       :rem 169  
3480 CLOSE2:CLOSE15                       :rem 141  
3485 N$=N2$                               :rem 39  
3490 RETURN                                :rem 176  
3500 INPUT "WOULD YOU LIKE TO DRAW ANOTHER SPRITE"  
      ;A$                                   :rem 162  
3510 IF LEFT$(A$,1)="Y"THENCLR :RUN2      :rem 5  
3520 END                                   :rem 161
```

# Multichar

---

John S. Graves

*How would you like a multicolor character editor that offers high-resolution characters, joystick control, and many other options—all on your unexpanded VIC? Look no further. Here it is.*

**D**esigning multicolor or high-resolution characters on your VIC is easy if you let the computer do the dirty work. "Multichar" is a menu-driven BASIC program for the unexpanded VIC which helps you design  $16 \times 16$ ,  $16 \times 8$ , and  $8 \times 8$  custom characters. You design the character in a large plotting area on the screen, using the joystick. The character is simultaneously displayed in actual size below the plotting area, while the values you need to generate it are displayed to the side.

## Entering the Program

To fit Multichar into the unexpanded VIC, the program had to be "crunched" by using BASIC statement abbreviations and eliminating spaces and remarks. Some lines are more than 88 characters long and will not execute unless abbreviations are used. Refer to your owner's manual for a list of the abbreviations.

Integer arrays are used instead of floating-point arrays to minimize variable storage requirements. IF statements are located, whenever possible, at the end of a line to avoid sacrificing a whole line for one logical statement, and colons are extensively used to combine statements on one line. Only 32 free bytes remain after the program has allocated storage for variables and character generation. Unfortunately, this isn't enough for an additional line to allow an eloquent exit from the program; you must press the RUN/STOP key to stop execution.

Make sure you use upper- and lowercase letters when you type in the color selection menu (lines 37-42). Otherwise, the menu will be unreadable when the program runs in multicolor mode.

### Multicolor Mode

Each character in multicolor mode may contain four colors (screen, border, auxiliary, and character color). Since  $16 \times 16$  characters are actually made up of four  $8 \times 8$  characters, you can choose a different character color for each quadrant, resulting in a  $16 \times 16$  character that contains seven colors.

Screen, border, and auxiliary colors are set by the color selection menu, which appears after you press M for multicolor mode. Character colors should be set for each quadrant when the large display area appears. You may choose the same character color for different quadrants.

A function key is referenced at each corner of the large display. Press the key for the quadrant you want and then press a number (1–8) to set the character color.

Once you've set all the colors, you're ready to be creative. The joystick moves the cursor, which represents a pixel. In multicolor mode the cursor is double-wide, since it takes two bits to define the color of a pixel. Press B, C, or A to turn on the pixel in the border, character, or auxiliary color. Press S to turn off the pixel by changing its color to the screen color. As you turn on pixels they will appear at  $8\times$  magnification on the large display, and at actual size on the small display below. The values you will need to generate the custom character in your own program or game are calculated automatically and appear as you turn on a pixel.

The first eight numbers on the left define the upper left-hand  $8 \times 8$  quadrant. The next eight numbers on the left define the lower left-hand quadrant. Numbers for the two right-hand quadrants appear on the right.

### High-Resolution Mode

The VIC's high-resolution mode allows only two colors per  $8 \times 8$  character (screen and character). You lose two colors, but the horizontal resolution is twice that of multicolor mode. As before, you may set the character color for each quadrant to a different value. That will give you a  $16 \times 16$  five-color (counting screen color) character.

Instead of using the keyboard to turn on a pixel, you simply press the fire button on the joystick. If you want to turn off a pixel, position the cursor over it and press the fire button again.

## How the Program Works

The following program outline should help you understand the program's logic and structure.

### Line(s)

- 1-2 Protect upper 512 bytes of BASIC area for RAM character memory. Initialize variables.
- 3-8 Joystick subroutine.
- 10-21 Subroutine to calculate character memory values for custom characters and generate small, actual-size display.
- 24-31 Set character colors subroutine. Press f1, f3, f5, or f7 followed by 1 to 8 to set the character color for each  $8 \times 8$  quadrant.
- 32-35 Main menu to select multicolor or high-resolution mode.
- 37-49 Color selection menu for multicolor mode. Uses joystick subroutine.
- 50-56 Set screen, border, and auxiliary colors based on menu.
- 57-58 Transfer characters A-? from ROM to RAM. Assign @ to cursor. Transfer character-color bar symbols, CHR\$(239) and CHR\$(247), to CHR\$(27) and CHR\$(29) locations in RAM. Change character memory starting address to RAM location 7168.
- 59-61 Draw the large  $8 \times$  magnified display. Draw auxiliary color bar at top of screen if in multicolor mode.
- 62-75 Multicolor design mode. Calls joystick subroutine. Press B, C, or A to turn on a pixel in border, character, or auxiliary color. Press S to turn off the pixel. Calls calculation subroutine.
- 80-86 High-resolution mode using joystick and calculation subroutines. Press Q to return to the mode selection menu.

## Multichar

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

1 POKE52,28:POKE56,28:CLR:DD=37154:PA=37137:PB=371
  52:C=30720:POKE37139,0 :rem 127
2 POKE646,6:POKE36879,29:POKE36869,240:SC=1:BD=6:G
  OTO32 :rem 248
3 POKEDD,127:S3=-((PEEK(PB)AND128)=0):POKEDD,255:P
  =PEEK(PA):S1=-((PAND8)=0):S2=((PAND16)=0):rem 13
4 S0=((PAND4)=0):FR=-((PAND32)=0):X=X+(S2+S3)*DX:Y
  =Y+S0+S1:IFX<0THENX=0 :rem 211
5 IFX>XTHENX=XM :rem 104
6 IFY<0THENY=0 :rem 127
7 IFY>YTHENY=YM :rem 110
8 L=X+Y*22+LI:RETURN :rem 5
10 N=0:IFX>7THENLI=7758 :rem 142

```

## 4: Graphics and Sound

```
11 IFX>7ANDS=-1THENLI=7757 :rem 90
12 FORI=7TO0STEPS:Z=ABS(I-7)+Y*22+LI:PC=PEEK(Z+C)A
   ND15:IFS=-1THEN15 :rem 173
13 IFPC=AUTHEENN=N+2↑(I-1)+2↑I:GOTO16 :rem 242
14 IFPC=BDTHENENN=N+2↑(I-1):GOTO16 :rem 223
15 IFPC=C%(1)ORPC=C%(2)ORPC=C%(3)ORPC=C%(4)THENN=N
   +2↑I :rem 210
16 NEXT:IFX<8THENPOKE7488+Y,N:P=7748 :rem 19
17 IFX>7THENPOKE7504+Y,N:P=7767 :rem 146
18 FORI=0TO2:POKEP+Y*22-I,32:NEXT:N$=STR$(N):LN=LE
   N(N$) :rem 126
19 FORI=0TOLN-2:POKEP+Y*22-I,VAL(MID$(N$,LN-I,1))+
   48:POKEP+Y*22-I+C,BD:NEXT:POKE8130,40 :rem 212
20 POKE8152,41:POKE8131,42:POKE8153,43:POKE38850,C
   %(1)+E:POKE38872,C%(3)+E:POKE38851,C%(2)+E
   :rem 116
21 POKE38873,C%(4)+E:RETURN :rem 66
23 IFG$="C"THEN28 :rem 188
24 IFG1=67THEN28 :rem 176
25 GETG$:IFGG$=""THEN25 :rem 139
26 C%(G1-132)=ASC(GG$)-49:FORI=38447TO38453:POKEI,
   C%(1):POKEI+9,C%(2):POKEI+374,C%(3) :rem 240
27 POKEI+383,C%(4):NEXT:GOTO31 :rem 19
28 CK=C%(1):IFX>7ANDY<8THENCK=C%(2) :rem 58
29 IFX<8ANDY>7THENCK=C%(3) :rem 77
30 IFX>7ANDY>7THENCK=C%(4) :rem 71
31 RETURN :rem 68
32 PRINT"{CLR}":PRINT:PRINT:PRINT :rem 30
33 PRINT" SELECT MODE":PRINT:PRINT:PRINT:PRINT" M
   {SPACE}- MULTICOLOR":PRINT:PRINT:PRINT:PRINT" H
   - HIGH RESOLUTION" :rem 9
34 GETG$:IFG$=""THEN34 :rem 253
35 IFG$="H"THENPRINT"{CLR}":GOTO57 :rem 157
37 PRINT"{CLR}"CHR$(14):PRINT:PRINTSPC(17)"S
   {SHIFT-SPACE}B{SHIFT-SPACE}A"SPC(17)"C
   {SHIFT-SPACE}D{SHIFT-SPACE}U"SPC(17)"R
   {SHIFT-SPACE}R{SHIFT-SPACE}XSELECT"SPC(6)"{RVS}
   {BLK}BLK {OFF}{BLU} . . ."; :rem 12
38 PRINT"SCREEN,"SPC(5)"WHT{2 SPACES}. . .BORDER,&
   "SPC(4)"{RVS}{RED}RED {OFF}{BLU} . . .AUXILIARY
   "SPC(3)"{RVS}{CYN}CYN {OFF}{BLU} . . .";:rem 79
39 PRINT"COLORS"SPC(6)"{RVS}{PUR}{OFF}{BLU} .
   {SPACE}. . "SPC(12)"{RVS}{GRN}GRN {OFF}{BLU} . .
   .MOVE (@)"SPC(4)"{RVS}BLU {OFF} . . .";:rem 13
40 PRINT"WITH"SPC(8)"{RVS}{YEL}YEL {OFF}{BLU} . .
   {SPACE}.JOYSTICK"SPC(4)"ORN{2 SPACES}.
   {3 SPACES}."SPC(12)"LORN.{3 SPACES}."; :rem 19
41 PRINT"PICK COLORS PINK .{3 SPACES}.WITH BUTTON
   {SPACE}LCYN .{3 SPACES}."SPC(12)"LPUR .
   {3 SPACES}.PRESS C"SPC(5)"LGRN"; :rem 82
```

```

42 PRINT" .{3 SPACES}.TO CONTINUE LBLU .{3 SPACES}
   ."SPC(12)"LYEL .{3 SPACES}.";           :rem 209
45 DX=1:LI=7807:XM=4:YM=15                 :rem 40
46 GOSUB3:CUC=PEEK(L):CC=PEEK(L+C):POKEL,122:POKEL
   +C,0:FORI=1TO50:NEXT:POKEL,CUC:POKEL+C,CC:GETGG
   $                                           :rem 9
47 IFGG$="C"THEN50                          :rem 4
48 IFFR=0THEN46                             :rem 153
49 POKEL,122:GOTO46                         :rem 134
50 FORX=0TOXMSTEP2:FORY=0TOYM:L=X+Y*22+LI:IFPEEK(L
   )<>122THENNEXTY                          :rem 69
51 IFX=0THENS=C=Y                          :rem 21
52 IFX=2THENBD=Y                            :rem 8
53 IFX=4THENAU=Y                            :rem 27
56 NEXTX:POKE36879,SC*16+BD+8:POKE36878,AU*16:PRIN
   T"{CLR}":FORI=38469TO38814:POKEI,SC:NEXT
                                           :rem 154
57 PRINT"{HOME}"CHR$(142):FORI=7176TO7679:POKEI,PE
   EK(I+25600):NEXT:FORI=7488TO7519:POKEI,0:NEXT
                                           :rem 245
58 FORI=7168TO7175:POKEI,255:POKEI+216,PEEK(I+2751
   2):POKEI+232,PEEK(I+27576):NEXT:POKE646,BD:POKE
   36869,255                                 :rem 56
59 PRINT:PRINT" F1[[[[[[[[{2 SPACES}[[[[[[[[F3":FORI
   =0TO15:PRINT:II=I*22+7748                :rem 124
60 POKEII,48:POKEII+C,BD:POKEII+19,48:POKEII+19+C,
   BD:NEXT:PRINT" F5]]]]]]]]{2 SPACES}]]]]]]]F7":IF
   G$="H"THEN80                              :rem 216
61 FORI=7680TO7701:POKEI,0:POKEI+C,9:NEXT  :rem 39
62 DX=2:LI=7750:XM=14:YM=15:GOSUB3:CUC=PEEK(L):CC=
   PEEK(L+C):POKEL+C,BD:POKEL-1+C,BD:POKEL,0:POKEL
   -1,0                                       :rem 12
63 FORI=1TO60:NEXT:POKEL-1,CUC:POKEL,CUC:POKEL-1+C
   ,CC:POKEL+C,CC:GETG$:IFG$=""THEN62      :rem 56
64 POKEL,0:POKEL-1,0:POKEL+C,BD:POKEL-1+C,BD:CK=SC
   :IFG$="B"THENCK=BD                       :rem 202
65 IFG$="A"THENCK=AU                       :rem 183
66 G1=ASC(G$):IFG1>132ANDG1<137ORG1=67THENGOSUB24
                                           :rem 195
73 IFG$="Q"THEN1                            :rem 150
74 POKEL+C,CK:POKEL-1+C,CK:S=-2:IFCK=SCTHENPOKEL,3
   2:POKEL-1,32                             :rem 123
75 E=8:GOSUB10:GOTO62                      :rem 28
80 DX=1:LI=7749:XM=15:YM=15:GOSUB3:CUC=PEEK(L):CC=
   PEEK(L+C):POKEL+C,0:POKEL,0           :rem 28
81 FORI=1TO40:NEXT:POKEL,CUC:POKEL+C,CC:GETG$:IFG$
   =""THENG$="0"                             :rem 157
82 IFG$="Q"THEN1                            :rem 150
83 G1=ASC(G$):IFG1>132ANDG1<137THENGOSUB24:rem 255
84 IFFR=0THEN80                             :rem 151

```

## 4: Graphics and Sound

---

```
85 G1=67:GOSUB24:POKEL,0:POKEL+C,CK:S=-1:IFCUC=0TH  
    ENPOKEL,32:POKEL+C,1           :rem 145  
86 E=0:GOSUB10:GOTO80           :rem 22
```



# The Magic Pointer

---

C. D. Lane

*"The Magic Pointer" is a machine language (ML) program that lets VIC users place a movable pointer onto the screen. The pointer can be moved about with the joystick. For any VIC.*

**T**his program creates a movable pointer that can be manipulated without significantly affecting BASIC. The position of the pointer can be polled by pressing the fire button, so that BASIC can PEEK it. Since the pointer itself is invisible to BASIC, PEEKing the screen will not find it and you do not have to worry about overwriting it (or it overwriting your screen).

## Grabbing IRQ

This invisibility is achieved by making the VIC execute "The Magic Pointer" ML routine along with its own IRQ routine. About 60 times a second, the 6502 microprocessor interrupts what it's doing (such as running your BASIC program) to perform the IRQ routine that scans the keyboard, increments the system clock, and does other necessary background tasks. Another kind of interrupt, termed BRK, occurs when you hit RUN/STOP and RESTORE.

When either of these interrupts occurs, the 6502 first pushes the return address and status register onto the stack. Then execution is transferred to a ROM routine whose address is contained in memory locations \$FFFE/FFFF. This routine saves the contents of the A, X, and Y registers by pushing them onto the stack. Next, it copies the saved status register from the stack to the X register in order to test the BRK bit and determine if it was an IRQ or BRK interrupt. Based on that test, the routine then jumps to one of two routines whose addresses are contained in RAM memory locations \$0314/0315 (IRQ) or \$0316/0317 (BRK). It is there that you are able to grab the beginning of the interrupt.

Each of these address-holding locations (vectors) normally points to another routine in ROM. For BRK, it is a routine that resets the VIC and returns to the READY state. For IRQ, it is a routine that performs the background tasks.

In order to add your own routine to the IRQ task list, you

must first disable all interrupts with SEI, and then change the RAM vector at \$0314/0315 to point to your routine. When your routine is finished, it must jump to the IRQ routine in ROM, in order for the VIC to perform its necessary functions. If all goes well, your routine will execute automatically, 60 times a second, along with every IRQ interrupt.

Intercepting the beginning of the interrupt is fairly straightforward, since the path to the routine goes through RAM. But if you wish to add a routine following the system functions, you must grab the end (tail) of the interrupt—and that is another trick entirely.

After the ROM routine is finished, the A, X, and Y registers are restored and an RTI is done. All this is done in ROM, so how do you take control once more at the end of the interrupt? If you recall how the interrupt began, the return address and the status register were pushed onto the stack (which is in RAM). So if you change the return address on the stack when you have the head of the interrupt, you can make it return to your own routine when finished.

To do this, you don't have to change what is on the stack. Instead, you simply push a new return address onto the stack, on top of the return address which the interrupt already placed there. The new address is the beginning of your tail-end routine. Stack items are processed in last-in, first-out (LIFO) order. Thus, when RTI is performed at the end of the ROM routine, the VIC will pull your new address from the stack, and execute your routine before doing anything else. When your routine ends with a second RTI, it will pull the old address from the stack, and return you to the original environment.

First, push the new return address, high byte first. Next push your current status register, even though you may not need it on return. When the interrupt starts, further interrupts are disabled. They will be enabled again when RTI restores the original status register. Since two RTIs will now be performed, you don't want the first one turning on interrupts before you're done. Thus, you have it restore the current status register (which has interrupts off).

Finally, you push three single bytes onto the stack. These are just dummy bytes, needed because the ROM routine will end by pulling three bytes from the stack to restore the A, X, and Y registers. You don't care what those registers contain

when your new routine starts, but the dummy bytes must be put on the stack to keep everything straight. That is all you have to do in order to make the ROM interrupt routine return to your tail-end routine.

When your routine ends, it must restore the original A, X, and Y registers from the stack and do an RTI. That will cause the VIC to pull the original address from the stack, returning you to where you were before all this began. Magic Pointer uses both of the techniques described above, to wedge one ML routine in at the head of the IRQ interrupt, and a second at the tail.

### Running Magic Pointer

The Magic Pointer BASIC loader program will load the routine into the highest page of memory and protect that page from BASIC. The loader also calculates the location of the screen and POKEs the appropriate bytes in the Magic Pointer routine. Therefore, the program will work in a VIC with any amount of memory.

Since the joystick is polled 60 times a second, the pointer is quite responsive. In fact, it can be thought of as a software light pen, since you can point to arbitrary positions on the screen and let BASIC know when you pick a position.

When you run the BASIC loader program, you should see the following:

```
READY.  
RUN  
INIT = SYS( 7440 )  
SOFTWARE REGS = 7424  
DONE
```

READY.

The numbers displayed depend on the amount of memory in your computer; the ones shown are for an unexpanded VIC. If the program detects an error in the DATA statements, it will print an error message and the suspect line number.

To start the pointer, do a SYS to the address of INIT. To kill the pointer hold STOP and RESTORE or reset the IRQ interrupt vector as described above.

If all is in order, once you have done a SYS to INIT, you will be back at the READY prompt and there will be an arrow (↑) in the middle of the screen. You should be able to move

the arrow around with the joystick. Try LISTing the program and moving the arrow about the screen as the listing scrolls by. You should be able to do so without affecting the listing at all, except for a slight decrease in speed. That is due to the fact that there is a slight delay when the pointer is on the screen—necessary in order to keep the arrow from blinking out completely. You will notice that it gets a bit fuzzy every few seconds, since it is constantly being turned on and off.

The first six locations of the routine are software registers. The first holds the character that will be placed on the screen; the next holds the color of the pointer. If this location is POKEd with a value of 128 or more, the color of the pointer will not be set and the pointer will take on the colors of whatever is on the screen.

The third and fourth locations are the pointer's row and column locations when the fire button was last pressed. The next two locations are the actual row and column of the pointer. The position of the pointer can be adjusted by POKeing values into these locations. Since the pointer routine and BASIC run in two different time frames, you needn't worry about confusing the routine by POKeing these registers. The location of the software registers is printed out when the BASIC loader is run.

### A Demonstration Program

The second listing is a BASIC program that runs at the same time as the Magic Pointer. This program partitions the VIC's screen into three parts: a drawing area (the canvas), a band of color (the color palette), and (at the bottom) a display of the graphic character set (the character palette).

To use the demonstration program, Program 2, type it in and save it. Load and run Program 1, the Magic Pointer program, but do *not* do a SYS to initiate it. Instead, load and run Program 2.

You will have the symbol @ in the middle of the canvas. You can go down to the character palette and use the fire button to select any character you wish, changing the pointer to that character. If you then go up to the canvas and select a location (by pressing the fire button again), then that character will be deposited where you indicate.

To select a color, move the cursor to the chosen color and press the fire button. You can also select a character from the

canvas, and the cursor will become that character and color while the character will be erased. Note that the demonstration program reads the position of the pointer but does not have anything to do with moving it.

The idea behind putting the Magic Pointer on the IRQ interrupt is to make the pointing to and picking up of objects appear to be a *hardware* feature for use from BASIC. A more ambitious challenge would be to implement *software* sprites (independently moving characters), with features like automatic collision detection, that would be invisible to BASIC's manipulations of the screen.

### Program 1. Magic Pointer for the VIC

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

10 M=PEEK(52)-1:POKE52,M:POKE56,M:POKE51,0:POKE55,
   0:CLR:M=PEEK(52):B=M*256:PC=0 :rem 217
20 H$="10":GOSUB90:PRINT"INIT = SYS("D+B)", "SOFTW
   ARE REGS =";B :rem 90
30 V=36866:S=(PEEK(V)/64AND2)+(PEEK(V+3)/4AND28):C
   =148+(PEEK(V)/64AND2) :rem 11
40 Q=0:FORK=1TO16:READH$:IFH$="END"THENPRINT"DONE"
   :POKEB+13,S:POKEB+15,C:END :rem 240
50 IFH$="**"THEND=M:GOTO70 :rem 12
60 GOSUB90:Q=Q+D :rem 215
70 POKEPC+B,D:PC=PC+1:NEXT:READD:IFD=QGOTO40
   :rem 74
80 PRINT"ERROR IN LINE";(PC/1.6)+90:STOP :rem 121
90 D=0:FORI=1TOLEN(H$):J=ASC(MID$(H$,I,1))-48:D=16
   *D+J+(J>9)*7:NEXT:RETURN :rem 118
100 DATA 1E,00,0B,0B,0B,0B,00,00,00,00,00,00,1E
   ,00,96,254 :rem 137
110 DATA 78,AD,15,03,8D,DA,**,A9,**,8D,15,03,AD,14
   ,03,8D,1347 :rem 41
120 DATA D9,**,A9,29,8D,14,03,58,60,A2,05,B5,22,9D
   ,06,**,1320 :rem 234
130 DATA BD,0C,**,95,22,CA,10,F3,AC,04,**,C0,0C,30
   ,04,E6,1507 :rem 28
140 DATA 23,E6,25,B9,FD,ED,18,6D,05,**,90,04,E6,23
   ,E6,25,1795 :rem 69
150 DATA 85,22,85,24,A0,00,B1,22,85,26,AD,00,**,91
   ,22,B1,1407 :rem 215
160 DATA 24,85,27,AD,01,**,30,02,91,24,A9,00,8D,13
   ,91,AE,1261 :rem 240
170 DATA 05,**,F0,0A,A9,10,2C,11,91,D0,03,CE,05,**
   ,E0,15,1313 :rem 233
180 DATA 10,0D,A9,7F,8D,22,91,2C,20,91,30,03,EE,05
   ,**,A9,1329 :rem 28

```

## 4: Graphics and Sound

---

```
190 DATA FF,8D,22,91,AE,04,**,F0,0A,A9,04,2C,11,91
    ,D0,03,1593 :rem 55
200 DATA CE,04,**,E0,16,10,0A,A9,08,2C,11,91,D0,03
    ,EE,04,1318 :rem 15
210 DATA **,A9,20,2C,11,91,D0,0C,AD,04,**,8D,02,**
    ,AD,05,1125 :rem 231
220 DATA **,8D,03,**,A0,02,CC,04,90,D0,FB,88,D0,F8
    ,A9,**,1878 :rem 31
230 DATA 48,A9,DB,48,08,48,48,48,4C,BF,EA,A0,00,2C
    ,01,**,1462 :rem 64
240 DATA 30,04,A5,27,91,24,A5,26,91,22,A2,05,BD,06
    ,**,95,1330 :rem 223
250 DATA 22,CA,10,F8,68,A8,68,AA,68,40,END:rem 176
```

### Program 2. Magic Pointer Demo

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
10 V=36866:S=4*(PEEK(V)AND128):C=37888+S:S=S+64*(P
    EEK(V+3)AND112) :rem 188
20 P=PEEK(52)*256:Y=PEEK(P+2):X=PEEK(P+3):L=330:R=
    L+44 :rem 122
30 S$=CHR$(18)+" "+CHR$(146):PRINTCHR$(147);:U=371
    37:Q=22 :rem 70
40 FORI=0TO501:IFI<LTHENPRINT" ";:GOTO70 :rem 233
50 IFI>=RTHENPRINT" ";:POKES+I,I-R:GOTO70 :rem 67
60 PRINTS$;:POKEC+I,((I-L)-2*(I>(L+Q)))AND7
    :rem 217
70 NEXT:POKEP,0:SYS(P+16) :rem 39
80 OY=Y:OX=X:WAITU,32,32:WAITU,32:Y=PEEK(P+4):X=PE
    EK(P+5):IFOY=YANDOX=XGOTO30 :rem 248
90 Z=Y*Q+X:J=S+Z:K=C+Z:I=PEEK(J):IFZ>=LGOTO130
    :rem 177
100 IFI=32THENM=A:GOTO120 :rem 11
110 A=I:B=PEEK(K):M=32:POKEP,A:POKEP+1,BAND7
    :rem 175
120 POKEJ,M:POKEK,B:GOTO80 :rem 130
130 IFZ>=RTHENA=I:POKEP,A:GOTO80 :rem 65
140 B=PEEK(K):POKEP+1,BAND7:GOTO80 :rem 61
```

# Sound Shaper

---

Steven Kaye

*“Sound Shaper” manipulates volume and frequency to give your Commodore a smoother, more musical sound. For the unexpanded VIC or the 64.*

One of the main differences between the sound capabilities of the Commodore 64 and the VIC is the shape of the sound's waveform. The VIC produces only square waves. One microsecond the sound is off, the next it's on. This abrupt onset of sound produces comparatively nonmusical tones that sound unlike any acoustic instrument.

The Commodore 64, on the other hand, can simulate musical instruments by controlling the waveshape of the sound produced. Instead of turning the sound on and off abruptly, it can increase and decrease the amplitude (volume) more gradually under control of the programmer. It is important to bear in mind that the rise-fall time is still on the order of fractions of milliseconds, but it is not instantaneous, as is the case with the VIC. It is this programmable rise-fall time that allows the Commodore 64 to sound more like a traditional acoustic instrument.

You cannot control the actual waveshape of sounds on the VIC, but you can simulate waveshaping by modulating the volume. The first part of Program 1 demonstrates a simple application of this technique. It plays the entire frequency range for one of the VIC's four voices.

First, the program asks for two inputs, the rise time and the fall time. Values between .5 and 10 seem to work best. Then the frequency value is POKed into the appropriate register (line 140). Two separate FOR-NEXT loops (lines 150 and 180) control the rise and fall times. As the volume varies between 0 and 15, the input variables control the rate of volume change. Experiment with different rise-fall time values to see what sounds you can produce.

Frequency manipulation can also be used to produce unique effects. The second part of Program 1 shows how to produce an echo effect by rapidly alternating a frequency with its complementary frequency.

Again, you move through the frequency scale. Line 270 applies the amplitude modulation technique described above.

Lines 280 and 300 POKE the frequency (and then the frequency subtracted from 383) into the appropriate voice register.

The first time through the loop, voice 2 (36875) is POKED with 128 and then rapidly alternated with 255 ( $255=383-128$ ) while the sound fades as variable DB decreases. The timing loops in 290 and 310, as well as the step value in line 270, can be manipulated to increase or decrease the reverberation effect. Voice 2 was chosen for the example, but any of the four voices can be used to produce interesting sounds.

### 64 Sound Shaper

Since the Commodore 64 already has a programmable sound envelope, Program 2 is included to make the SID chip more accessible. By changing values entered for attack, decay, sustain, and release, you can control the shape of the sounds produced by the program. The second part of the program produces an echo effect very similar to the effect produced in the VIC version. The parameters set in the first part are also used for the sounds produced in the second part.

#### Program 1. VIC Sound Shaper

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
40 PRINT"{CLR}{9 DOWN}"TAB(2)"{RVS}SHAPING{OFF}
   {RVS}VIC{OFF} {RVS}SOUNDS{OFF}"           :rem 179
45 FOR T=1 TO 1500:NEXT                       :rem 244
50 PRINT"{CLR}{7 DOWN}{6 RIGHT}SHAPED (1)" :rem 37
55 PRINTTAB(9);"{DOWN}OR":PRINTTAB(7)"{DOWN}ECHO (
   2)"                                         :rem 166
60 PRINT"{4 DOWN}{9 RIGHT}";:INPUT I$:IFVAL(I$)<10
   R VAL(I$)>2THEN50                           :rem 15
70 ONVAL(I$)GOTO100,240                       :rem 49
100 REM*** THIS PART PRODUCES "SHAPED" MUSICAL NOTES***
    ES***                                     :rem 213
110 PRINT "{3 DOWN}{2 RIGHT}RISE AND FALL TIME"
                                           :rem 36
115 PRINT"VALUES MUST EXCEED 0"             :rem 95
116 INPUT R,D:IF (R=0)OR(D=0) THEN 116      :rem 45
120 V=36878:S=36875                          :rem 13
130 FOR F=128 TO 255 STEP3                   :rem 71
140 POKE S,F                                 :rem 137
150 FOR DB=0 TO 15 STEP 5/R                  :rem 107
160 POKE V,DB                               :rem 206
170 NEXT                                     :rem 215
180 FOR DB=15 TO 0 STEP -5/D                 :rem 141
```



```

190 POKE V,DB :rem 209
200 NEXT :rem 209
210 FOR T=1 TO 50:NEXT :rem 189
220 NEXT :rem 211
230 POKE V,0:END :rem 135
240 REM*** THIS PART CREATES AN ECHO EFFECT***
:rem 71
250 V=36878:S=36875 :rem 17
260 FOR P=128 TO 255 STEP 3 :rem 85
270 FOR DB=15 TO 1 STEP -.5 :rem 73
280 POKE V,DB:POKE S,P :rem 9
290 FOR T=1 TO 10:NEXT :rem 193
300 POKE S,383-P :rem 92
310 FOR J=1 TO 10:NEXT :rem 176
320 NEXT:NEXT :rem 77
330 POKE V,0 :rem 119

```

## Program 2. 64 Sound Shaper

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

15 PRINT"{CLR}SET PARAMETERS FOR SOUND AND ECHO"
:rem 12
20 CHIP = 54272 :rem 199
22 FOR T=CHIP TO CHIP + 24 : POKET,0:NEXT :rem 234
30 INPUT "ATTACK RATE (0-15)";AT$:AT=VAL(AT$):IF A
T<0 OR AT>15 THEN 30 :rem 82
40 INPUT "DECAY RATE (0-15)";DE$:DE=VAL(DE$):IF DE
<0 OR DE>15 THEN 40 :rem 198
50 INPUT "SUSTAIN VOLUME (0-15)";SU$:SU=VAL(SU$):I
F SUS<0OR SU>15 THEN 50 :rem 35
60 INPUT "RELEASE RATE(0-15)";RE$:RE=VAL(RE$):IF R
E<0ORRE>15 THEN 60 :rem 171
80 POKECHIP+24,15:POKECHIP+5,16*AT+DE :rem 209
90 POKECHIP+6,16*SU+RE :rem 68
100 FOR T= 20{2 SPACES}TO 80 STEP 5:POKECHIP+4,17
:rem 103
110 POKECHIP,50:POKECHIP+1,T :rem 223
115 FORJ= 1 TO 500+1.7↑AT+1.7↑DE:NEXTJ :rem 141
120 POKECHIP+4,16:FORH=1TO2↑RE:NEXT:NEXT :rem 107
200 FOR T= 20 TO 80 STEP 5 :rem 232
210 FOR DB = 15 TO 1STEP -.5 :rem 67
215 PRINT"{HOME}{5 DOWN}*ECHO*{6 LEFT}{7 SPACES}"
:rem 242
220 POKECHIP+4,17:POKECHIP+24,DB:POKECHIP+1,T:FORP
=1TO10:NEXT :rem 111
230 POKECHIP+1,100-T:FORJ=1TO10:NEXT:NEXT:NEXT
:rem 202
240 POKECHIP+4,16 :rem 219

```

# VIC Piano

Brad Bascom

*Turn your computer keyboard into a piano keyboard with this easy-to-use program for the unexpanded VIC.*

**T**he VIC has three musical voices. To make them play, just calculate the number to POKE, set up the durations, and turn the sound on and off. It works beautifully.

But it's programming, not *playing*. What if you want to sit down at the computer and pick out melodies, the way you can with a piano or organ? Typing something like POKE 36876,207 for each separate note isn't exactly recreational music.

## Easy Melodies

"VIC Piano" lets you use the top two rows of your keyboard, as if they were the keys on the piano. Just type in the program, save it to disk or tape, and then type in RUN.

You'll see almost two octaves of a piano keyboard, from G to E, with white and black keys. Below the piano keys are the VIC keys to press to play that note. Even more helpful is the white dot that appears directly under the picture of the key that was last pressed. It follows along as fast as you can play, allowing you to pick out melodies without looking at the VIC keyboard at all.

When you play a note, it will continue to sound until you play the next note. If you want a musical rest (silence) press any key that does *not* represent a note. The dot will jump to the lower-left corner of the screen and the sound will stop until you press another note.

Sometimes, if you play *very* quickly, you'll get ahead of the program. The keyboard buffer will come to your aid—the VIC can keep track of up to ten notes at a time. However, you'll find it's pretty hard to play fast enough to use up that buffer.

You may notice that some of the pitches aren't exact. That can't be helped, since the numbers the VIC understands do not correspond to the regular music scale. The VIC understands numbers that represent sound frequencies, and the numbering system does not always have an exact equivalent on the musical scale. So don't tune your piano to your VIC!

## How the Program Works

For each key you press, VIC Piano must decide several things:

1. Does the key represent a valid note?
2. Where on the screen should the dot be placed, showing which note is being played?
3. What frequency number should be POKEd into the sound register at 36876?

That can be quite complicated, and if the program had to test each time for every possible note, it would run very slowly.

Fortunately, with careful design, the program can run very quickly, even in BASIC. How? The placement of the dot is easiest. The piano keys are displayed on the screen so that each of the 22 notes can be clearly represented by a character on the VIC's 22-character line. All you need to do is determine the starting address of the row just under the piano keys. In the unexpanded VIC, that address is 7900. Each keypress will cause the dot to be displayed at 7900 plus the left-to-right order of that note. G, the lowest note, is 0, so that the dot character (screen code 81) will be POKEd into  $7900 + 0$ . The highest note, high E, is in the twenty-first column, so that when high E is played, the dot character is POKEd into  $7900 + 21$ .

Slightly more difficult is the calculation of the frequency to be played. For instance, the notes G, G#, A, and A# have POKE values of 175, 179, 183, and 187. So far, all the notes are four steps apart. But high C#, D, D#, and E have values of 227, 228, 229, and 231. There's no regular mathematical relationship between the notes' order and their POKE values.

The answer is to use arrays for both values. The screen offsets from 0 to 21 are in the array  $J(n)$ . The sound POKE values from 175 to 231 are in the array  $N(n)$ . Both occur in exactly the same order, so that when the note  $N(x)$  is played, the dot will be displayed at  $7900 + J(x)$ .

What will be the index into the arrays? That is provided by the ASCII value of the key the user presses. That way you won't have to use IF statements to set the sound and screen POKE values. You can just use the arrays  $J(n)$  and  $N(n)$ , with the keypress value as the index  $n$ . In BASIC, it couldn't be any faster.

Get the ASCII character of the key pressed with the statement GET A\$. Each ASCII character has a numeric value,

which is found using the function `ASC(A$)`. If the key pressed was Q, for instance, the value of `ASC(A$)` would be 81; if W is pressed, the value of `ASC(A$)` would be 87.

For the 22 VIC keys used as musical keys, the lowest value of A\$ that would play a note is 42, and the highest is 94. Since values lower than 42 and higher than 94 can never play a note, simply leave them out of the array. DIMENSION both arrays with `DIM J(55),N(55)`. Then, when you GET A\$, you'll say `X=ASC(A$)-42`. That means that if the \* (asterisk) key (42) is pressed, X will equal 0, and if the up-arrow key (94) is pressed, X will equal 52.

That's just what the program does. In line 160, the program DIMs `N(55),J(55)`. Then in line 180, it READs the values of the arrays. Each pair of numbers in the DATA statements starting at 800 represents the ASCII value of a key and the sound register POKE value for the corresponding note. The locations in which the dot appears are numbered in the same order, from 0 to 21, so the loop `FOR I=0 TO 21` yields the right values for the screen POKEs. In one pass through the loop, then, you have given every valid note an ASCII value (the subscript or index number), a sound POKE value `N(n)`, and a screen POKE value `J(n)`.

What about the leftover values of `N(n)` and `J(n)`? Line 170 puts 0 in every element of `N(n)` and 264 in every element of `J(n)`. By default, every possible key value will have the effect of the space bar (a rest). Then, when the note values are initialized in line 180, all the elements that are not valid notes will be rests.

Because of all this setup, initialization takes a few seconds. However, the extra time spent in setting up makes the program itself run very quickly. The main loop is from 400 to 480, only eight short lines. Line 400 GETs the value of A\$. If no key is pressed (`A$=""`), the line keeps looping back on itself until a key is pressed.

Lines 10–190 set up the screen and initialize the arrays and variables. Lines 800–830 are the DATA statements. Each pair of numbers is an ASCII value and its corresponding sound POKE value. The true ASCII values (instead of the ASCII values minus 42) are in the DATA statements, so it will be easier to see which character is paired with each sound POKE value.

## VIC Piano

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

10 POKE36879,106:PRINT"{CLR}{BLK}";           :rem 205
12 PRINT"{RVS}{YEL}{6 SPACES}VIC{2 SPACES}PIANO
   {28 SPACES}";                               :rem 120
20 PRINT"{RVS}{WHT}G{BLK} {WHT}A{BLK} {WHT}BC{BLK}
   {WHT}D{BLK} {WHT}EF{BLK} {WHT}G{BLK} {WHT}A
   {BLK} {WHT}BC{BLK} {WHT}D{BLK} {WHT}E"; :rem 52
25 FORT=1TO4                                     :rem 231
30 PRINT"{RVS}{WHT} {BLK} {WHT} {BLK} {WHT} [G]
   {BLK} {WHT} {BLK} {WHT} [G]{BLK} {WHT} {BLK}
   {WHT} {BLK} {WHT} [G]{BLK} {WHT} {BLK} {WHT} ";
                                               :rem 178
40 NEXTT                                         :rem 247
50 FORT=1TO3                                     :rem 228
60 PRINT"{RVS}{WHT} == [G]== [G]== [G]==
   {OFF}";                                       :rem 207
70 NEXT                                          :rem 166
75 PRINT"{DOWN}{WHT} 2 3{2 SPACES}5 6{2 SPACES}8 9
   Ø{2 SPACES}- £ ";                           :rem 136
80 PRINT"Q W ER T YU I O P@ * ↑";             :rem 101
100 PRINT"{5 DOWN}";                          :rem 242
110 PRINT"PLAY EACH NOTE BY THE CHARACTERS ABOVE."
;                                               :rem 18
120 PRINT"PRESS{2 SPACES}SPACE BAR TO REST."
                                               :rem 65
160 DIM N(55),J(55)                           :rem 171
170 FOR I=Ø TO 55:J(I)=264:N(I)=Ø:NEXT I:XX=55
                                               :rem 27
180 FOR I=Ø TO 21:READ K,M:J(K-42)=I:N(K-42)=M:NEX
   T I                                         :rem 146
190 POKE36878,15                               :rem 107
400 GETA$:IFA$=""THEN400                     :rem 75
420 X=ASC(A$)-42:IF X<Ø OR X>55 THEN X=55 :rem 167
430 POKE7900+J(XX),32                         :rem 157
440 POKE36876,Ø                               :rem 49
450 POKE36876,N(X)                           :rem 249
460 XX=X                                       :rem 223
470 POKE7900+J(X),81                          :rem 77
480 GOTO400                                    :rem 105
800 DATA 81,175,50,179,87,183,51,187        :rem 3
810 DATA 69,191,82,195,53,198,84,201,54,204 :rem 90
820 DATA 89,207,85,210,56,212,73,215,57,217,79,219
   ,48,221                                     :rem 15
830 DATA 80,223,64,225,45,227,42,228,92,229,94,231
                                               :rem 172

```

# The Mozart Machine

---

Original Program by Donald J. Eddington  
VIC and 64 Translations by Gregg Peele

*Using the techniques described here, your computer can compose music with this special technique. The compositions are unmistakably Mozartian in style. For the unexpanded VIC or the 64.*

**I**f you've ever gone through the steps required to make your VIC or 64 play a particular piece of music, you realize that it can be a major programming task. But to make your computer actually *write* music is an even more difficult challenge.

How can you turn your Commodore into a composer? First, you've got to find a way to work with POKES and DATA statements to actually create the measures of music. In addition, you need to be able to READ the values in different orders so that the songs will be different each time the program is run.

Unfortunately, the commonly used string manipulation methods won't work very well for such applications. In fact, they quickly yield a tangled mess. You could, of course, write each measure as a series of POKES (for note, duration, and next note) for every possible note—but you'll probably find it too long and repetitive.

## Array Referencing

One of the best ways around that problem is to use a technique called array referencing. First, to create measures of music, set up an array of all variables and reference them by subscript into a POKÉ loop. This program requires 14 variations on nine variables to make the music. A random number generator is used to make the song different every time the program is run. A Mozartian flavor results from a deliberate shortening of the low notes and varying the length of the high notes.

To keep the music from becoming totally random, DATA statements select the measures according to their underlying tonality—tonic, subdominant, dominant, or supertonic. In addition, the program provides for cadence measures every four measures and for a final ending chord for each tune.

## Entertainment and Education

This program does not copy any of Mozart's music; instead, it imitates Mozart's style. You might want to introduce some alternative composition rules and stylistic ideas and come up with a mechanical composer of your own. How about a Pink Floyd machine or a Bartok machine? Using this program as a starting point, you and your computer will soon be composing in a variety of styles.

### Program 1. Mozart Maker, VIC Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

10 DIMX(14,9) :rem 174
25 POKE36879,8 :rem 11
30 PRINT"{CLR}{2 DOWN}{RIGHT}{WHT}WELCOME! I AM VI
  CLANG AMAZIUS MOZART." :rem 56
35 PRINT"{DOWN}I PLAY SONGS LIKE THE CHILD PRODIGY
  ,WOLFGANG AMADEUS MOZART MIGHT HAVE DONE."
  :rem 42
40 PRINT"{2 DOWN}MOZART LIVED FROM 1756TO 1791 AND
  WROTE OVER626 WORKS IN 31 YEARS." :rem 165
45 FOR TD=1TO3000:NEXTTD :rem 205
50 PRINT"{DOWN}{RED}THE 5 PIECES YOU HEAR ARE BEIN
  G WRITTEN BY{2 SPACES}THE COMPUTER AS YOU
  {3 SPACES}LISTEN!" :rem 37
90 FORT=1TO14:FORTT=1TO9:READX:X(T,TT)=X+212:NEXTT
  T:NEXTT :rem 41
91 DATA3,11,11,11,11,16,16,11,7,3,11,16,11,16,13,1
  6,11,16,3,11,13,11,16,13,16,11,16 :rem 205
92 DATA3,13,16,13,19,22,19,23,13,3,13,19,13,19,16,
  19,13,13,3,19,13,13,3,19,19,13,3 :rem 203
93 DATA7,13,16,13,22,16,16,13,7,0,7,16,7,13,11,16,
  7,13,7,19,22,13,16,13,16,11,7 :rem 50
94 DATA7,13,11,13,7,11,19,13,7,7,13,11,13,7,11,19,
  13,13,3,11,13,11,16,16,16,16,16 :rem 136
95 DATA0,16,13,7,7,7,16,7,7,3,19,16,13,13,13,19,13
  ,13 :rem 75
100 REM SET VOICE NUMBERS,AND{2 SPACES}SPEED VALUE
  :rem 224
120 K=36875:L=36876:P=175 :rem 93
130 POKE36878,12 :rem 98
160 REM SET SELECTED{2 SPACES}MEASURE BY DATA NUMB
  ER :rem 64
170 DATA1,3,6,2,1,4,6,2,3,4,1,5,1,4,6,7,1,4,6,2,1,
  3,6,9 :rem 126
172 DATA 1,1,4,5,1,4,6,2,3,4,1,5,1,4,1,5,1,4,6,9
  :rem 252

```

## 4: Graphics and Sound

---

```
174 DATA 1,4,6,2,3,6,1,5,1,4,6,7,3,4,6,2,1,4,3,7,1
      ,4,6,9 :rem 138
176 DATA 1,4,3,7,1,6,4,5,6,3,6,2,4,6,1,5,1,4,6,9
      :rem 16
178 DATA 1,4,3,7,6,3,6,2,4,6,1,5,1,3,6,7,3,6,1,5,1
      ,4,6,9,8 :rem 246
180 READRR :rem 89
190 IFRR=1THEN300 :rem 253
200 IFRR=2THENY=12:GOTO1010 :rem 145
210 IFRR=3THEN310 :rem 249
220 IFRR=4THEN320 :rem 252
230 IFRR=5THENY=14:GOTO1010 :rem 153
240 IFRR=6THEN330 :rem 1
250 IFRR=7THENY=13:GOTO1010 :rem 156
260 IFRR=8THEN500 :rem 4
270 IFRR=9THEN1500 :rem 55
300 Y=1:X=RND(1):IFX<.35THENY=3 :rem 122
301 IFX>.75THENY=2 :rem 74
302 GOTO1010 :rem 144
310 Y=10:IFRND(1)<.4THENY=11:GOTO1010 :rem 180
320 Y=4:X=RND(1):IFX<.35THENY=5 :rem 129
321 IFX>.75THENY=6 :rem 80
322 GOTO1010 :rem 146
330 Y=7:X=RND(1):IFX<.35THENY=8 :rem 136
331 IFX>.75THENY=9 :rem 84
332 GOTO1010 :rem 147
500 PRINT"{CLR}{DOWN}{YEL}WELL, THAT'S ALL--HOPE
      {4 SPACES}YOU LIKED IT!!" :rem 87
510 PRINT"{DOWN}RUN IT AGAIN--AND HEAR FIVE MORE S
      ONGS!!":END :rem 241
900 REM FOLLOWING ARE THE MUSIC MEASURES THAT VICL
      ANG USES TO MAKETHE WHOLE TUNE :rem 159
1010 POKEK,X(Y,1):POKEL,X(Y,2):FORT=1TOP:NEXT:POKE
      K,0:POKEL,X(Y,3):FORT=1TOP:NEXT :rem 248
1020 POKEK,X(Y,4):POKEL,X(Y,5):FORT=1TOP:NEXT:POKE
      K,0:POKEL,X(Y,6):FORT=1TOP:NEXT :rem 2
1030 POKEK,X(Y,7):POKEL,X(Y,8):FORT=1TOP:NEXT:POKE
      K,0:POKEL,X(Y,9):FORT=1TOP :rem 147
1035 NEXT:GOTO 160 :rem 18
1500 POKE36876,235:POKE36875,239:POKE36874,235:FOR
      T=1TO1200:NEXT :rem 6
1510 POKE36874,0:POKE36875,0:POKE36876,0:FORT=1TO2
      000:NEXT:GOTO160 :rem 206
```



## Program 2. Mozart Maker, 64 Version

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

100 DIMH(14,9),L(14,9) :rem 97
101 FORT=54272TO54272+24:POKET,0:NEXT :rem 216
102 POKE54296,15 :rem 94
103 FORT=54272+5TO54272+24STEP7:POKET,17:POKET+1,2
    44:NEXT :rem 196
110 POKE53281,7:POKE53280,5 :rem 243
120 PRINT"{CLR}{2 DOWN}{RIGHT}{WHT}WELCOME! I AM 6
    4CLANG AMAZIUS MOZART." :rem 51
130 PRINT"{DOWN}{2 SPACES}I PLAY SONGS LIKE THE CH
    ILD PRODIGY," :rem 51
135 PRINT"WOLFGANG AMADEUS MOZART MIGHT HAVE DONE"
    :rem 95
140 PRINT"{2 DOWN}MOZART LIVED FROM 1756 TO 1791 A
    ND WROTE"; :rem 244
145 PRINT"{6 SPACES}OVER 626 WORKS IN 31 YEARS"
    :rem 90
150 PRINT"{DOWN}{BLK}{4 SPACES}THE 5 PIECES YOU HE
    AR ARE BEING" :rem 49
155 PRINT" COMPOSED BY THE COMPUTER AS YOU LISTEN"
    :rem 17
160 FORT=1TO2000:NEXT :rem 30
170 POKE53281,5:POKE53280,7 :rem 249
180 FORT=1TO14:FORTT=1TO9:READH,L:H(T,TT)=H:L(T,TT
    )=L:NEXTTT:NEXTT :rem 105
190 DATA12,143,31,165,31,165,15,210,31,165,37,162,
    18,209,31,165,14,24 :rem 102
200 DATA12,143,31,165,37,162,15,210,37,162,33,135,
    18,209,31,165,18,209 :rem 153
210 DATA12,143,31,165,33,135,15,210,37,162,33,135,
    18,209,31,165,18,209 :rem 150
220 DATA12,143,33,135,37,162,16,195,42,62,50,60,21
    ,31,50,60,16,195 :rem 206
230 DATA12,143,33,135,42,62,16,195,42,62,37,162,21
    ,31,33,135,16,195 :rem 6
240 DATA12,143,42,62,33,135,16,195,25,30,84,125,21
    ,31,33,135,12,143 :rem 249
250 DATA14,24,33,135,37,162,16,195,50,60,37,162,18
    ,209,33,135,12,143 :rem 60
260 DATA14,24,28,49,37,162,14,24,33,135,31,165,18,
    209,28,49,16,195 :rem 238
270 DATA14,24,42,62,50,60,16,195,37,162,33,135,18,
    209,31,165,14,24 :rem 218
280 DATA14,24,33,135,31,165,16,195,28,49,31,165,21
    ,31,33,135,14,24 :rem 216
290 DATA14,24,33,135,31,165,16,195,28,49,31,165,21
    ,31,33,135,16,195 :rem 20

```

## 4: Graphics and Sound

---

```
300 DATA12,143,31,165,33,135,15,210,37,162,37,162,
    18,209,37,162,18,209 :rem 157
310 DATA 12,143,37,162,33,135,14,24,28,49,28,49,18
    ,209,28,49,14,24 :rem 182
320 DATA12,143,42,62,37,162,16,195,33,135,33,135,2
    1,31,33,135,16,195 :rem 55
330 REM SET VOICE NUMBERS,AND{2 SPACES}SPEED VALUE
    :rem 229
340 K=54272:P=175:W=K+4 :rem 255
350 POKE54296,15 :rem 99
360 REM SET SELECTED{2 SPACES}MEASURE BY DATA NUMB
    ER :rem 66
370 DATA1,3,6,2,1,4,6,2,3,4,1,5,1,4,6,7,1,4,6,2,1,
    3,6,9 :rem 128
380 DATA 1,1,4,5,1,4,6,2,3,4,1,5,1,4,1,5,1,4,6,9
    :rem 253
390 DATA 1,4,6,2,3,6,1,5,1,4,6,7,3,4,6,2,1,4,3,7,1
    ,4,6,9 :rem 138
400 DATA 1,4,3,7,1,6,4,5,6,3,6,2,4,6,1,5,1,4,6,9
    :rem 6
410 DATA 1,4,3,7,6,3,6,2,4,6,1,5,1,3,6,7,3,6,1,5,1
    ,4,6,9,8 :rem 235
420 READRR :rem 86
425 ON RR GOTO520,426,550,560,427,590,428,620,1000
    :rem 15
426 Y=12:GOTO650 :rem 163
427 Y=14:GOTO650 :rem 166
428 Y=13:GOTO650 :rem 166
520 Y=1:X=RND(1):IFX<.35THENY=3 :rem 126
530 IFX>.75THENY=2 :rem 78
540 GOTO650 :rem 109
550 Y=10:IFRND(0)<.4THENY=11:GOTO650 :rem 146
560 Y=4:X=RND(1):IFX<.35THENY=5 :rem 135
570 IFX>.75THENY=6 :rem 86
580 GOTO650 :rem 113
590 Y=7:X=RND(1):IFX<.35THENY=8 :rem 144
600 IFX>.75THENY=9 :rem 83
610 GOTO650 :rem 107
620 PRINT"{CLR}{DOWN}{BLU}{2 SPACES}WELL,THAT'S AL
    L--HOPE YOU LIKED IT!!" :rem 219
625 POKE53281,1 :rem 44
630 PRINT"{DOWN}RUN IT AGAIN--AND HEAR FIVE MORE S
    ONGS!!":END :rem 244
640 REM FOLLOWING ARE THE MEASURES THAT 64CLANG US
    ES TO MAKE THE WHOLE TUNE :rem 234
650 POKEW,17:POKEK,L(Y,1):POKEK+1,H(Y,1):POKEK+7,L
    (Y,2):POKEK+8,H(Y,2) :rem 156
655 POKEW+7,17:FORQ=1TOP:NEXT:POKEW,16 :rem 187
```

```
660 POKEK,L(Y,3):POKEK+1,H(Y,3):FORT=1TOP:NEXT
                                     :rem 115
670 POKEW,17:POKEK,L(Y,4):POKEK+1,H(Y,4):POKEK+7,L
    (Y,5):POKEK+8,H(Y,5)
                                     :rem 170
675 POKEW+7,17:FORQ=1TOP:NEXT:POKEW,16
                                     :rem 189
680 POKEK,L(Y,6):POKEK+1,H(Y,6):FORT=1TOP:NEXT
                                     :rem 123
690 POKEW,17:POKEK,L(Y,7):POKEK+1,H(Y,7):POKEK+7,L
    (Y,8):POKEK+8,H(Y,8)
                                     :rem 184
695 POKEW+7,17:FORQ=1TOP:NEXT:POKEW,16
                                     :rem 191
700 POKEK,L(Y,9):POKEK+1,H(Y,9):FORT=1TOP:NEXT:GOT
    O370
                                     :rem 135
1000 POKEK,143:POKEK+1,12:POKEK+7,165:POKEK+8,31:P
    OKEK+14,30:POKEK+15,25
                                     :rem 206
1010 POKEW,17:POKEW+7,17:POKEW+14,17:FORT=1TO2000:
    NEXT:POKEW,16:POKEW+7,16
                                     :rem 63
1020 POKEW+14,16:GOTO370
                                     :rem 121
```

□ □ □ □ □

□ □ □ □ □

# Chapter 5

---

## Utilities and Programming Aids

□ □ □ □ □

□ □ □ □ □

# Color Chart

---

Sheldon Leemon

*Check out all the possible combinations of character colors and background colors with "Color Chart." For the unexpanded VIC and Commodore 64.*

One of the nicest things about color graphics on the VIC and 64 is that you can choose the color of each character that you print. This allows you to place many different, colored text statements on the same screen at one time. When you begin to design a screen with more than one text color, however, you may run into a problem. Many text colors do not show up well against certain background colors.

Most programmers use trial and error to discover which text color goes well with which background color. But wouldn't it be nice if you could see all of the combinations on the screen at one time? Then you could see which combinations would work best.

The two programs accompanying this article, one for the VIC and one for the 64, do just that. The VIC version has 16 rows of eight characters each. The top row has a black background (color 0), and each subsequent row has a different background color with a higher color value. The column at the extreme left has a black text character, and each subsequent column has a different color text character with a higher color value.

The 64 version is the same, except that there are eight additional text colors, 16 columns, and a total of 256 color combinations.

## Using the Computer's Speed

How is it possible to show more than one background color on the screen at one time? After all, the background color is determined by the value in a memory location called the color register (the 64 uses location 53281, while the VIC uses location 36879). Since those registers can hold only one number at a time, the only way to have more than one background color at a time is to change the value of this register in the middle of the display.

To understand how this is done, you have to know something about how a picture is displayed on your TV. An electron beam called a raster starts at the top-left corner of the screen and moves in a horizontal line from left to right. As this beam moves, it lights up appropriate parts of the screen line. When it gets to the end of a line, it goes back to the left side, drops down slightly, and starts all over again.

It takes about two hundred of these lines to complete your computer display, and the raster scans all of these lines 60 times every second. If you tell it the exact instant to change the background color, it can do so after part of the screen has already been drawn.

### Interrupting the Raster Scan

Both the VIC and 64 have a raster register. This is a memory location which holds the number of the line which is currently being scanned. The short machine language program in each of the examples simply waits for a particular line at the top to be scanned. When that happens, it changes the background color and waits for a few more lines to be scanned before changing the background color again. When all of the changes are done, it goes back to the beginning.

Type the program in carefully, and save it before you run it. The program will loop continuously, displaying all of the color combinations available to you. See which combination you think will be the best for your particular program, make a note of it, and then press RUN/STOP-RESTORE to break out of the program.

### Program 1. Color Chart, VIC Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
10 FOR ADRES=828TO 874:READ DATTA:POKE ADRES,DATTA
   :NEXT ADRES                                     :rem 250
20 PRINT CHR$(147):A=PEEK(648)*256:FOR I=A TO A+51
   2:POKE I,160:NEXT I                             :rem 58
30 PRINT:FOR I=0 TO 15:PRINT:PRINT TAB(7);:FOR J=0
   TO 7                                             :rem 170
40 POKE 646,J:PRINTCHR$(J+48);:NEXT J,I:PRINT:PRIN
   T                                               :rem 164
50 POKE 646,1:PRINTCHR$(18);"THIS CHART SHOWS ALL
   {2 SPACES}";                                   :rem 228
60 PRINT"COMBINATIONS OF LETTER";                 :rem 93
70 PRINT"AND BACKGROUND COLORS";                 :rem 248
80 SYS828                                         :rem 9
```



```

828 DATA 169, 41, 133, 251, 169, 9           :rem 165
834 DATA 141, 15, 144, 162, 15, 120         :rem 188
840 DATA 173, 4, 144, 197, 251, 208         :rem 205
846 DATA 249, 173, 15, 144, 24, 105        :rem 205
852 DATA 16, 234, 234, 234, 234, 234       :rem 249
858 DATA 234, 234, 141, 15, 144, 165      :rem 254
864 DATA 251, 24, 105, 4, 133, 251        :rem 143
870 DATA 202, 16, 223, 48, 209             :rem 2

```

## Program 2. Color Chart, 64 Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

40 FOR I=49152 TO 49188: READ A: POKE I,A: NEXT:PO
   KE 53280,11                               :rem 175
50 PRINT CHR$(147):FOR I=1024 TO I+1000: POKE I,16
   0: POKE I+54272,11:NEXTI                   :rem 204
60 FOR I=0 TO 15: FOR J=0 TO 15               :rem 237
70 P=1196+(40*I)+J: POKE P,J+1: POKE P+54272,J: NE
   XT J,I                                       :rem 174
80 PRINT TAB(15)CHR$(5)"COLOR CHART":FOR I=1 TO 19
   :PRINT:NEXT                                 :rem 100
85 PRINT"THIS CHART SHOWS ALL COMBINATIONS OF
   {3 SPACES}"                                 :rem 112
86 PRINT "FOREGROUND AND BACKGROUND COLORS.
   {6 SPACES}"                                 :rem 237
87 PRINT "FOREGROUND INCREASES FROM LEFT TO RIGHT"
   :rem 88
88 PRINT "BACKGROUND INCREASES FROM TOP TO BOTTOM"
   ;                                           :rem 152
90 SYS 12*4096                                 :rem 200
100 DATA 169,90,133,251,169,0,141,33,208,162,15,12
   0                                           :rem 191
105 DATA 173,17,208,48,251,173,18,208      :rem 35
110 DATA 197,251,208,249,238,33,208,24,105,8,133,2
   51,202,16,233,48,219                       :rem 121

```

# Cursor GET for the VIC and 64

David Mills

*This practical subroutine lets you create a cursor for use during GET routines. For any VIC or 64.*

**I**n many cases, it makes more sense to use GET instead of INPUT when asking a user for information. The GET command is more flexible and gives you greater control over the characters that are entered.

But there is a drawback. INPUT gives you a blinking cursor, which you don't have with GET. However, because the cursor is often convenient (and sometimes essential), this subroutine was developed to provide a cursor while using the GET statement for input.

Look at this short program to see why a cursor can be important.

```
10 FOR K=1 TO 30 :rem 7
20 GET A$: IF A$="" GOTO 20 :rem 241
30 PRINT A$;: NEXT K :rem 84
```

When you run this program, the computer will GET and print 30 keystrokes. Notice that the cursor has vanished. But the vanishing cursor can create problems if you include keystrokes such as cursor movement, RETURNS, DELETES, and so on in your input. In those cases, it is very easy to forget where the cursor is. The only way to find out is to start typing and see where the letters appear on the screen.

## Creating a Cursor

The short subroutines given below will provide VIC and 64 users with a blinking cursor during GET routines. The subroutine is invisible to the host program, even when embedded within it, because its first statement sends the host program around the subroutine. Also, to minimize the chance of interference with any other program, the subroutine uses variables starting with X. That makes it even safer to use as long as you avoid such variables in your main program.

The subroutine is called by using GOSUB 1102, and on return A\$ will hold the character from the GET statement.

In lines 1102-1104, XL% is set to the memory address of the screen cursor. In line 1104, XC is set to the color memory address, and the program automatically compensates for different memory sizes in the VIC. In line 1105 the current color (PEEK(646)) is put in the color memory, XO% is set to the character at the cursor, and XT% and XQ% are set up for the blinking process. In line 1106, XT% is reversed and POKEd into the cursor position on the screen. Then XQ% is reset for the next blink, and a FOR-NEXT loop that actually gets the character is started.

If something has been typed in, line 1107 resets the screen and returns. Otherwise, the FOR-NEXT loop continues in line 1108. When the loop is complete, the screen character is reversed again and the process repeats.

You can remove the REMs except in line 1109. Line 1101 directs the host program to line 1109, and if you remove that REM then line 1109 vanishes and you will get an execution error.

### Program 1. Cursor GET, VIC Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

10 GOSUB1102:PRINTA$;:GOTO10:REM THIS IS THE "HOST
   " PROGRAM                               :rem 77
1101 GOTO1109:REM GET WITH CURSOR BLINK    :rem 79
1102 XL%=PEEK(211)                          :rem 212
1103 IFXL%>21THENXL%=XL%-22:GOTO1103       :rem 133
1104 XL%=XL%+PEEK(214)*22+4096:XC=33792+XL%:IFPEEK
   (210)>20THENXC=XC+512:XL%=XL%+3584      :rem 70
1105 POKEXC,PEEK(646):XO%=PEEK(XL%):XT%=XO%:XQ%=12
   8:IFXO%>127THENXQ%=-XQ%                :rem 235
1106 XT%=XT%+XQ%:POKEXL%,XT%:XQ%=-XQ%:FORXR=1TO60:
   REM CHANGING 60 CHANGES BLINK SPEED   :rem 238
1107 GETA$:IFA$<>""THENPOKEXL%,XO%:RETURN :rem 51
1108 NEXT XR:GOTO1106                      :rem 238
1109 REM                                    :rem 175

```

### Program 2. Cursor GET, 64 Version

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

10 GOSUB1102:PRINTA$;:GOTO10:REM THIS IS THE "HOST
   " PROGRAM                               :rem 77
1101 GOTO1109:REM GET WITH CURSOR BLINK    :rem 79
1102 XL%=PEEK(211)                          :rem 212
1103 IFXL%>39THENXL%=XL%-40:GOTO1103       :rem 142
1104 XL%=XL%+PEEK(214)*40+1024:XC=54272+XL%:rem 87

```

## 5: Utilities and Programming Aids

---

```
1105 POKEXC, PEEK(646):XQ%=PEEK(XL%):XT%=XQ%:XQ%=12
      8:IFXQ%>127THENXQ%=-XQ% :rem 235
1106 XT%=XT%+XQ%:POKEXL%,XT%:XQ%=-XQ%:FORXR=1TO60:
      REM CHANGES SPEED :rem 217
1107 GETA$:IFA$<>" "THENPOKEXL%,XQ%:RETURN :rem 51
1108 NEXT XR:GOTO1106 :rem 238
1109 REM :rem 175
```

# File Copier

---

Martin Engert

"File Copier" is a BASIC utility that lets you transfer files from one disk to another, using a single drive, without worrying about starting addresses or machine language. For any VIC-20 or Commodore 64.

"File Copier" can help those who want to copy sequential or program files from one disk to another but have only a single disk drive and no machine language monitor. Since the program is written in BASIC, it's a bit slow. But one advantage of this program is that you don't have to know the initial address or length of the program to be transferred.

File Copier works on both the VIC-20 and Commodore 64. The program first resets the top-of-BASIC pointers to reserve 1K of memory for itself. The remaining memory is used to store your file temporarily. VIC users should make sure enough memory is available for this purpose before running the program; any amount of expansion memory can be added if necessary.

Each byte of the file is read from disk using the GET# command and POKEd into free memory. Then you insert the new disk and the program writes these bytes onto it using PRINT#. After the file is copied, the top-of-BASIC pointers are restored to normal.

Screen instructions are provided within the program for easier use.

## File Copy for VIC and 64

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
10 POKE251,PEEK(52) :rem 49
20 POKE52,PEEK(44)+4:POKE56,PEEK(52):CLR :rem 89
30 PRINT"{CLR}RUN THIS PROGRAM TO" :rem 175
40 PRINT"COPY A PROGRAM OR" :rem 106
50 PRINT"SEQUENTIAL FILE FROM" :rem 133
60 PRINT"ONE DISK (THE SOURCE" :rem 30
70 PRINT"DISK) TO ANOTHER (THE" :rem 73
80 PRINT"DESTINATION DISK).":rem 253
90 PRINT"INSERT SOURCE DISK.":rem 57
100 M=256*PEEK(52) :rem 191
110 OPEN15,8,15 :rem 32
```

## 5: Utilities and Programming Aids

---

```
120 PRINT"WHAT IS THE NAME OF" :rem 203
130 PRINT"THE FILE OR PROGRAM":INPUTF$ :rem 83
140 T$="P":PRINT"WHAT IS THE FILE TYPE" :rem 252
150 PRINT"(P FOR PROGRAM, S FOR" :rem 68
160 PRINT"FILE)" :rem 177
170 INPUTT$ :rem 160
180 OPEN2,8,2,F$+",""+T$+",R" :rem 128
190 INPUT#15,E,E$,X,X:IFE<>0THENPRINTE$:CLOSE2:GOT
O120 :rem 134
200 GET#2,A$:IFA$=""THENA$=CHR$(0) :rem 90
210 POKEM+J,ASC(A$):J=J+1:IFST=0THEN200 :rem 66
220 CLOSE2 :rem 60
230 PRINT"INSERT DESTINATION" :rem 125
240 PRINT"DISK AND PRESS {RVS}RETURN" :rem 228
250 PRINT"TO COPY." :rem 116
260 GETC$:IFC$<>CHR$(13)THEN260 :rem 6
270 PRINT"PRESS {RVS}RETURN{OFF} IF YOU" :rem 7
280 PRINT"WANT TO KEEP THE NAME" :rem 111
290 PRINTF$ :rem 146
300 INPUT"FILE NAME ";F$ :rem 77
310 OPEN2,8,2,F$+",""+T$+",W" :rem 128
320 INPUT#15,E,E$,X,X:IFE<>0THENPRINTE$:CLOSE2:GOT
O300 :rem 129
330 FORK=0TOJ-1:PRINT#2,CHR$(PEEK(M+K));:NEXT
:rem 7
340 CLOSE2:CLOSE15 :rem 85
350 POKE52,PEEK(251):POKE56,PEEK(251):CLR :rem 145
```

# 1540/1541 Disk Housekeeping

---

Michael Maione

*This simple utility will help you clean up the clutter on your 1540 or 1541 disk drive. For any VIC or Commodore 64.*

**I**f you experiment with different programming techniques and save each enhancement along the way, your disks tend to get cluttered with outdated routines. This short program will help with your disk housekeeping chores.

Type in the program, save it, and then give it a try. To prevent a disaster, practice first on a disk that does not hold any important programs or files.

## Scratching and Unscratching Files

If you choose the Scratch option, a portion of the disk directory will be displayed—just enough to fit comfortably on the VIC screen along with the query “Scratch program?” If you do not wish to scratch any of the programs listed, press the N key and another portion of the directory will be presented. Repeat this procedure until you find the file you want.

To scratch a file on the list, press the Y key. Then, type in the name of the file to be scratched and press RETURN. The file will be scratched automatically, and the program will restart from the beginning. Continue this process until all unwanted files have been removed from the disk.

When the entire disk directory has been presented, you can end the program by pressing the N key in response to the scratch question.

If the Unscratch option is chosen, the program collects all free blocks off the disk and displays the names of any previously scratched files. You are then prompted with a scratched file. Enter Y to unscratch it. Sometimes the file will be partially scrambled because other files have been written over part of it. In that case, a message is displayed indicating that the file cannot be recovered.

### Abbreviated Directory Listing

Lines 10–40 set the screen color, display the title, and begin the program. Lines 50–190 read eight filenames from the disk directory and print them to the screen. The file sizes and types have been eliminated from the screen display to make it clearer and more concise.

Lines 200–240 branch, depending on whether or not you wish to scratch a file. Line 250 ends the program when all files have been displayed and the N key is pressed. Line 260 returns to the directory for more filenames.

Line 270 gets the filename which is to be scratched and ends the program if you accidentally hit RETURN before you type a filename.

The subroutine in lines 340–380 examines the filename you enter. If the filename is longer than ten characters, it abbreviates the name and adds an asterisk (\*) to the end. This is done so that the filename and the scratch command together will not be longer than one VIC screen line.

Finally, lines 290–330 use the dynamic keyboard technique to scratch the file and run the program again from the beginning.

### Use PRINT# Abbreviation

The Scratch portion of the program runs on both the VIC (any memory configuration) and 64. Since the line length of the 64 screen is 40 characters, abbreviating the filename when it is longer than 10 characters should not be necessary. You may wish to modify or simply eliminate the subroutine in lines 340–380. Commodore 64 users who wish to display more than eight filenames on the screen at one time can adjust line 190 accordingly.

If you are using a VIC, be sure to abbreviate the command PRINT# (by using P SHIFT-R) in line 310, to insure that the filename and the command together do not exceed the 22-character line length of the VIC screen. If they are too long, the RETURNS which are POKed into memory in line 330 will not be entered properly when the END statement is reached.

With a little experimentation, VIC users should be able to eliminate the necessity for the subroutine which abbreviates



the longer filenames. Try using branch statements and a second routine for printing the OPEN, scratch, RUN, and cursor up instructions in lines 290-330. Also try adding lines to validate the disk and reorganize the directory. Finally, add a few lines to read the error channel and to make the program more complete.

### Disk Housekeeping

For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.

```

0 REM IF YOU HAVE AN UNEXPANDED VIC, DELETE 10, 11
, AND 400-1000                                :rem 162
10 DIMA(255),C%(77,28),D%(1),T%(224,1),S%(224,1),L
%(224),R%(77)                                  :rem 38
11 D%(0)=58:D%(1)=42:Z%=CHR$(0):B%=CHR$(3):D%="0"
                                                :rem 234
20 PRINT"{CLR}{6 DOWN}{8 SPACES}SCRATCH":PRINT
                                                :rem 251
30 PRINT"{6 DOWN}S=SCRATCH, U=UNSCRATCH"      :rem 155
40 GETQ$:IFQ$=""THEN400                        :rem 11
45 IFQ$="U"THEN400                             :rem 6
46 IFQ$<>"S"THEN400                           :rem 18
50 PRINT"{CLR}{DOWN}{RVS}DISK"               :rem 23
60 OPEN1,8,0,"$0"                             :rem 77
70 N%=CHR$(0)                                  :rem 152
80 GET#1,A$,A$                                  :rem 190
90 F$="":B=0:GET#1,A$,A$                       :rem 205
95 IFC=0THENPRINT"{RVS}";                    :rem 250
100 IFA$=""THENX=1:GOTO200                     :rem 0
110 GET#1,A$,B$                                :rem 233
120 PRINTTAB(5);                               :rem 184
130 GET#1,A$                                  :rem 89
140 IFA$=""THENPRINT:A=A+1:GOTO190            :rem 40
150 IFA%=CHR$(34)THENB=1:A$=""                :rem 127
160 IFB=1THENF$=F$+A$:PRINTA$;               :rem 13
170 IFLEN(F$)>16THENC=C+1:GOTO90              :rem 144
180 GOTO130                                     :rem 102
190 IFA<8THEN90                                :rem 118
200 PRINT"SCRATCH PROGRAM? Y/N"               :rem 152
210 GETZ$:IFZ$=""THEN210                      :rem 123
220 IFZ$="Y"THEN270                            :rem 67
230 IFZ$="N"THEN250                            :rem 55
240 GOTO210                                    :rem 98
250 IFX=1THENCLOSE1:GOTO390                   :rem 209
260 PRINT"{CLR}{2 DOWN}":A=0:GOTO90          :rem 226
270 CLOSE1:INPUT"{DOWN}WHICH PROGRAM";P$:IFP$=""TH
ENEND                                          :rem 93
280 GOSUB340                                    :rem 177
290 PRINT"{CLR}{8 DOWN}"                      :rem 135

```

## 5: Utilities and Programming Aids

---

```
300 PRINT"OPEN15,8,15" :rem 242
310 PRINT"{2 DOWN}PR15,"CHR$(34)"S0:"X$CHR$(34)" :rem 76
320 PRINT"{2 DOWN}RUN":PRINT"{10 UP}" :rem 50
330 POKE631,13:POKE632,13:POKE633,13:POKE198,3:END :rem 147
340 FORA4=1TOLEN(P$):R$=MID$(P$,A4,1) :rem 86
350 X$=X$+R$ :rem 110
360 NEXT :rem 216
370 IFLEN(X$)>10THENX$=LEFT$(X$,10)+"*" :rem 158
380 RETURN :rem 123
390 CLOSE15:RUN :rem 167
400 PRINT"{3 DOWN}LOADING FREE SECTORS":OPEN15,8,1 :rem 110
5,"I"+D$:GOSUB3020
410 OPEN3,8,3,"$" +D$:GOSUB3020 :rem 99
420 A0=1:GET#3,A$:A=ASC(A$+Z$) :rem 100
430 READA1:IFA=A1THEN470 :rem 168
440 F%=F%+1:IFF%=3THEN510 :rem 112
450 READA1:IF A1=0THEN430 :rem 149
460 GOTO450 :rem 108
470 READA1:IFA1=0THEN490 :rem 157
480 READB1:FORJ=A0TOA1:R%(J)=B1:NEXTJ:A0=J:GOTO470 :rem 88
490 IFA=1ORA=65THEND1=1:T9=35:S9=3:D9=18 :rem 118
500 IFA=67THEND1=257:T9=77:S9=4:D9=39 :rem 151
510 IFT9=0THENCLOSE3:PRINT"?? DISK NOT RECOGNIZED :rem 132
{SPACE}??":STOP
520 FORJ=1TOD1:GET#3,A$:NEXTJ :rem 18
530 FORJ=1TOT9:T1=0 :rem 147
540 IFJ=51THENGET#3,A$,A$,A$,A$ :rem 190
550 GET#3,A$:C=ASC(A$+Z$) :rem 81
560 K1=0:FORK=0TOS9-1:GET#3,A$:A=ASC(A$+Z$) :rem 217
570 FORL=0TO7:A%=A/2:D1=A-A%*2:IFK1<=R%(J)THENC%(J :rem 13
,K1)=D1
580 A=A%:T1=T1+D1:K1=K1+1:NEXTL,K :rem 169
590 NEXTJ :rem 39
600 CLOSE3 :rem 63
610 OPEN2,8,2,"#0":GOSUB3020 :rem 255
900 K=0:PRINT"{CLR}LOOKING FOR SCRATCHED":PRINT"FI :rem 130
LES..."
910 T=D9:S=1 :rem 163
920 GOSUB 2000 :rem 221
930 FORD=2TO255STEP32:IFA(D)<>0ORA(D+1)=0THEN980 :rem 111
940 IFK=0THENPRINT"DO YOU WANT TO RECOVER:" :rem 161
950 GETX$:FORK=D+3TOD+18:PRINTCHR$(A(K));:NEXTK:PR :rem 58
INT"? ";
960 GETX$:IFX$<>"Y"ANDX$<>"N"THEN960 :rem 128
```

```

970 PRINTX$:IFX$="Y"THEN1010 :rem 185
980 NEXTD :rem 36
990 T=A(0):S=A(1):IFT=D9THEN920 :rem 35
1000 PRINT"THAT'S ALL ":GOTO1270 :rem 83
1010 T6=T:S6=S:D6=D:T=A(D+1):S=A(D+2):L%(0)=A(D+28
)+A(D+29)*256:L%=0 :rem 169
1020 GETX$:PRINT"IS THIS FILE:" :rem 88
1030 PRINT" 1. SEQUENTIAL" :rem 239
1040 PRINT" 2. PROGRAM" :rem 14
1050 PRINT" 3. USR" :rem 242
1060 IFA(D+19)=0THEN1080 :rem 38
1070 PRINT" 4. RELATIVE" :rem 87
1080 PRINT"{2 SPACES}WHICH NUMBER? "; :rem 80
1090 GETX$:IFX$=""THEN1090 :rem 229
1100 X=ASC(X$)-48:IFX<1ORX>4GOTO1090 :rem 144
1110 PRINTX$:X=X+128 :rem 185
1120 IFX=132THENT%(0,1)=A(D+19):S%(0,1)=A(D+20):IF
T%(0,1)=0THEN1020 :rem 91
1130 IFT>T9ORS<0THENT=0 :rem 195
1140 IFT<1ORS>R%(T)THENPRINT" BAD CHAIN!":GOTO1260
:rem 235
1150 IFC%(T,S)=0THENPRINT" ALLOCATED BLOCKS!":GOTO
1260 :rem 243
1160 GOSUB3000:L%=L%+1 :rem 192
1170 FORJ=0TO1:PRINT#15,"M-R";CHR$(J);B$:GET#15,A$
:rem 115
1180 A(J)=ASC(A$+Z$):NEXTJ :rem 220
1190 T4=T:S4=S:T=A(0):S=A(1):IFT<>0THEN1130:rem 10
1200 T=T%(0,1):S=S%(0,1):T%(0,1)=0:IFT<>0THEN1130
:rem 132
1210 IFL%<>L%(0)THENPRINT" INCORRECT BLOCK COUNT!"
:GOTO1260 :rem 42
1220 T=T6:S=S6:D=D6 :rem 104
1230 GOSUB 3000 :rem 9
1240 PRINT#15,"M-W";CHR$(D);B$;CHR$(1);CHR$(X)
:rem 51
1250 PRINT#15,"U2:2,";D$;T;S:GOSUB3020:GOTO1300
:rem 227
1260 PRINT"SORRY - IT WON'T WORK" :rem 181
1270 CLOSE2 :rem 114
1300 CLOSE2:PRINT#15,"V0":CLOSE15:FORQW=1TO10000:N
EXT:RUN :rem 59
2000 REM GRAB FULL DISK BLOCK :rem 139
2010 GOSUB3000 :rem 6
2020 FORJ=0TO255:PRINT#15,"M-R";CHR$(J);B$:GET#15,
A$ :rem 217
2030 A(J)=ASC(A$+Z$):NEXTJ:RETURN :rem 241
3000 REM READ BLOAD :rem 37
3010 PRINT#15,"B-R"2;VAL(D$);T;S :rem 49
3020 REM GET ERROR STATUS :rem 247

```

## 5: Utilities and Programming Aids

---

```
3030 INPUT#15,E,E$,E1,E2           :rem 42
3040 IFE<>0THENPRINT"{RVS}DISK ERROR:{OFF}"E;E$,E1
    ;E2                             :rem 21
3050 RETURN                         :rem 168
10000 DATA 1,17,20,24,19,30,17,35,16,0 :rem 44
10010 DATA 65,17,20,24,18,30,17,35,16,0 :rem 102
10020 DATA 67,39,28,53,26,64,24,77,22,0 :rem 126
```

# ML Tracer

---

Original Program by Thomas G. Gordon  
VIC and 64 Versions by Tim Victor

*Debugging machine language can be a trying experience; trying to study a program in ROM can be just as frustrating, even with a disassembler. Here's a utility that will help you solve both problems: a single-step ML tracer for the VIC or 64.*

**A**n anyone who has ever worked with machine language knows how helpful it can be to single-step through a program. "ML Tracer" allows you to step through a machine language routine one event at a time and print out the contents of all microprocessor registers after each instruction. It also allows you to follow all branches, jumps, and returns. The program will display the address, opcode, mnemonic, and operand of each instruction.

When ML Tracer is run, there will be a ten-second delay while the DATA statements are read. You'll then be asked for the hex address of the ML program you wish to examine.

You can change the contents of any register before each instruction is executed. Press A for the accumulator, X for the X register, Y for the Y register, S for the stack pointer, P for the processor status register, or I for the instruction pointer (program counter). When you're through loading registers, press RETURN once more to execute the next instruction.

Hexadecimal numbers are used for all input and output. If you enter an address as a one-, two-, or three-digit hexadecimal number, zeros will be added on the left to make a four-digit number. If too many digits are entered, the rightmost four digits will be used. The same applies to changing the value in a register. The number that you enter will be converted to a two-digit hexadecimal number using the same rules.

## The Execution Subroutine

The program is written mostly in BASIC but contains two machine language subroutines. The first, the initialization subroutine, copies the lowest three pages (768 bytes) of RAM, which are used by BASIC, to a location above the BASIC program. The other, the execution subroutine, exchanges the two

three-page blocks of data, loads all the registers with their saved values, then executes one instruction (which has been POKEd in from BASIC). When the instruction has been executed, the registers are saved and BASIC's original lower three pages of memory are restored.

Lines 10000–10031 contain four-character extended mnemonics for the 6502's instruction set. The fourth character is a tag code identifying the addressing mode of the instruction. In lines 110–120, the mode is identified and the proper subroutine is called.

There are several instructions which cannot be allowed to execute in the machine language subroutine. If any control transfer instructions (JMP, JSR, RTS, RTI, or a conditional branch) should be executed, control would not be properly returned to the BASIC program. These instructions are simulated in BASIC instead, so that they appear to execute successfully. The SEI and CLI instructions are ignored, since interrupts are always disabled during the execution subroutine.

### How Does It Work?

The simplest way to see how the program works is to trace through an example. Suppose the instruction LDA #\$20 resides at addresses \$03C0–\$03C1. For this instruction, the extended mnemonic is LDAB, where LDA stands for Load Accumulator, and B is the tag code for immediate addressing. The hexadecimal representation for LDA immediate is \$A9, which is equivalent to decimal 169.

Line 50, the top of the main loop, calls the keyboard pause routine at line 7000, which also handles changing registers. In line 55, the variable C is loaded with 169 by PEEKing the memory addressed by B, the instruction pointer. The value of B, 960 in this example, is then converted to hexadecimal characters in line 2000 and PRINTed.

In line 60, NOP instructions are POKEd into the execution routine to take up space after one- or two-byte instructions. The hexadecimal value of the opcode is printed next; then the mnemonic is retrieved from the array R\$( ). If the mnemonic is a blank, this instruction is undefined and an error message is displayed. Otherwise, the standard (three-character) mnemonic

is PRINTed, the opcode is POKEd into the execution routine at OP, and the program counter is incremented to 961.

The ASCII code for B is 66, so the ON GOSUB in line 120 transfers control to line 400. Here, the symbol for the addressing mode is printed. The one-byte operand routine, at line 3000, PEEKs location 961, pointed to by the program counter. This number is POKEd into OP+1, then converted to hexadecimal and PRINTed. After incrementing the program counter to point to the start of the next instruction, a RETURN is executed at line 3000.

At line 5000, the execution routine is SYSed, the contents of the registers are displayed, and control passes back to line 120. Here, a GOTO 50 takes us back to the top of the loop, where the instruction at \$3C2 will be executed.

## The Benefits of Tracing

You will find that this program is most useful for testing small ML programs, such as those called as subroutines from BASIC. It's also good for examining sections of larger programs when you're not sure how a particular routine works. If you're learning machine language, you'll find that the register display is an enormous help in understanding the effects and side effects of each instruction, especially the bits (flags) of the processor status register.

Do be careful, though. Any program is vulnerable when dealing with something as powerful as machine language, and this one is no exception. There are more ways to kill a BASIC program from ML than anyone can name in one sitting, so always be conscientious about saving your programs. After you type this one in, save it before you even think about running it. One typographical error could cause the program to erase itself, or at least lock up the computer.

There are also some ML programs that this tracer can't follow, such as those which disconnect the keyboard or video display (whether intentionally or accidentally). But if everything is saved on disk or tape (for real security, take the disk or cassette out of the drive), you can experiment as much as you want. Then, if disaster strikes, all you have to do is turn the computer off and reload the program.

## 5: Utilities and Programming Aids

---

### ML Tracer for the VIC and 64

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
10 GOSUB6000 :rem 167
35 POKEA,0:POKE X,0:POKEY,0:POKEP,52:POKES,255
   :rem 63
40 PRINT"START ADDRESS (HEX)";:H$="C000":INPUTH$
   :rem 106
45 H$=RIGHT$(H$,4):GOSUB1500:B=D:PRINT"ANY KEY TO
   {SPACE}STEP" :rem 9
50 GOSUB7000:D=FRE(0) :rem 197
55 PRINT:C=PEEK(B):D=B:GOSUB2000:PRINTH$ " ";
   :rem 148
60 POKEOP+1,234:POKEOP+2,234 :rem 127
70 D=C:GOSUB2000:PRINTRIGHT$(H$,2) " "; :rem 170
80 IFR$(C)=" "THENPRINT"INVALID OPCODE":PRINT:GOTO3
   5 :rem 229
90 R$=LEFT$(R$(C),3):PRINTR$ " ";:POKEOP,C:B=B+1
   :rem 175
100 IFR$="BRK"THENPRINT:GOTO35 :rem 141
110 U$=RIGHT$(R$(C),1):IFU$=" "THENGOSUB200:GOTO50
   :rem 126
120 ONASC(U$)-64GOSUB300,400,500,600,700,800,900,1
   000,1100,1200,1300:GOTO50 :rem 156
199 REM{4 SPACES}>IMPLIED MODE< :rem 42
200 IFR$="RTS"THENGOSUB4000:B=D:GOSUB4000:B=D*256+
   B+1:GOSUB5005:RETURN :rem 42
203 IFR$<>"RTI"THEN208 :rem 16
205 GOSUB4000:POKEP,D:GOSUB4000:B=D:GOSUB4000:B=D*
   256+B:GOSUB5005:RETURN :rem 204
208 IFR$="SEI"ORR$="CLI"THENGOSUB5005:RETURN:rem 4
210 GOSUB5000:RETURN :rem 242
299 REM{4 SPACES}>ABSOLUTE MODE< :rem 134
300 PRINT"$";:GOSUB2500 :rem 68
310 IFR$="JMP"THENB=PEEK(OP+1)+PEEK(OP+2)*256:GOSU
   B5005:RETURN :rem 34
320 IFR$<>"JSR"THEN340 :rem 13
330 B=B-1:D=INT(B/256):GOSUB3500:D=B-INT(B/256)*25
   6:GOSUB3500 :rem 249
335 B=PEEK(OP+1)+PEEK(OP+2)*256:GOSUB5005:RETURN
   :rem 141
340 GOSUB5000:RETURN :rem 246
399 REM{4 SPACES}>IMMEDIATE MODE< :rem 183
400 PRINT"#$";:GOSUB3000:GOSUB5000:RETURN :rem 253
499 REM{4 SPACES}>ZERO PAGE MODE< :rem 134
500 PRINT"$";:GOSUB3000:GOSUB5000:RETURN :rem 219
599 REM{4 SPACES}>ABSOLUTE,X< :rem 232
600 PRINT"$";:GOSUB2500:PRINT",X";:GOSUB5000:RETUR
   N :rem 170
699 REM{4 SPACES}>ABSOLUTE,Y< :rem 234
```



```

700 PRINT"$";:GOSUB2500:PRINT",Y";:GOSUB5000:RETUR
    N                                     :rem 172
799 REM{4 SPACES}>(INDIRECT,X)<         :rem 46
800 PRINT("$";:GOSUB3000:PRINT",X");:GOSUB5000:RET
    URN                                  :rem 249
899 REM{4 SPACES}>(INDIRECT),Y<         :rem 48
900 PRINT("$";:GOSUB3000:PRINT"),Y";:GOSUB5000:RET
    URN                                  :rem 251
999 REM{4 SPACES}>ZERO PAGE,X<         :rem 234
1000 PRINT"$";:GOSUB3000:PRINT",X";:GOSUB5000:RETU
    RN                                    :rem 209
1099 REM{3 SPACES}>ZERO PAGE,Y<        :rem 19
1100 PRINT"$";:GOSUB3000:PRINT",Y";:GOSUB5000:RETU
    RN                                    :rem 211
1199 REM{3 SPACES}>RELATIVE JUMP<      :rem 202
1200 PRINT"TO ";:D=PEEK(B):B=B+1:D=D+(D>127)*256:D
    =B+D:B1=D                             :rem 52
1210 GOSUB2000:PRINT"$H$;:BM=BM(INT(C/64)):BC=BMA
    NDPEEK(P)                             :rem 254
1220 IFBC=(INT(C/32)AND1)*BMTHENB=B1   :rem 88
1230 GOSUB5005:RETURN                   :rem 42
1299 REM{3 SPACES}>INDIRECT JUMP<     :rem 193
1300 PRINT("";:GOSUB2500:PRINT")";:B=PEEK(OP+1)+PE
    EK(OP+2)*256                          :rem 118
1310 B=PEEK(B)+PEEK(B+1)*256:GOSUB5005:RETURN
                                           :rem 160
1499 REM{3 SPACES}> HEX TO DEC <      :rem 137
1500 D=0:FORI=1TOLEN(H$):J=ASC(MID$(H$,I,1))-48:D=
    D*H+J+7*(J>9):NEXT:RETURN           :rem 180
1999 REM{3 SPACES}> DEC TO HEX <      :rem 142
2000 H$="":FORI=1TO4:E=INT(D/H):J=D-E*H:H$=CHR$(J+
    48-7*(J>9))+H$:D=E:NEXT             :rem 192
2005 RETURN                             :rem 167
2499 REM{3 SPACES}> 2BYTE OPERAND <   :rem 165
2500 D=PEEK(B+1):POKEOP+2,D:GOSUB2000:PRINTRIGHT$(
    H$,2);:GOSUB3000:B=B+1:RETURN       :rem 90
2999 REM{3 SPACES}> 1BYTE OPERAND <   :rem 169
3000 D=PEEK(B):POKEOP+1,D:GOSUB2000:PRINTRIGHT$(H$
    ,2);:B=B+1:RETURN                   :rem 124
3499 REM{3 SPACES}> PUSH <            :rem 119
3500 J=PEEK(S):POKEML+512+J,D          :rem 194
3505 IFJ=0THENPRINT:PRINT"WARNING: STACK OVERFLOW"
    :J=256                                :rem 114
3510 POKES,J-1:RETURN                  :rem 57
3999 REM{3 SPACES}> POP <             :rem 43
4000 J=PEEK(S):D=PEEK(ML+513+J)       :rem 23
4005 IFJ=255THENPRINT:PRINT"WARNING: STACK UNDERFL
    OW":J=-1                              :rem 221
4010 POKES,J+1:RETURN                  :rem 51

```

## 5: Utilities and Programming Aids

```
4999 REM{3 SPACES}> EXECUTE ONE INSTRUCTION <
:rem 148
5000 SYSML+23 :rem 237
5005 PRINT:FORK=0TO4:D=PEEK(A+K):GOSUB2000:rem 107
5010 PRINTMID$(" A= X= Y= S= P=",3*K+1,3);PRINTRI
GHT$(H$,2);:NEXT:PRINT:RETURN :rem 143
5999 REM{3 SPACES}> INITIAL STUFF < :rem 208
6000 ML=2*4096+8*256 :rem 245
6001 A=ML+240:X=A+1:Y=X+1:S=Y+1:P=S+1:H=16:OP=ML+9
2 :rem 239
6002 DIMR$(255):DIMBM(3):FORI=0TO3:READB:BM(I)=B:N
EXT :rem 204
6003 FORT=0TO255:READR$(T):NEXT :rem 154
6004 READR$:IFR$<>"END"THENPRINT"ERROR IN OPCODES"
:PRINT"CHECK FOR TYPO'S":END :rem 133
6005 I=0:FORT=MLTOML+164:READB:POKET,B:I=I+B:NEXT
:rem 128
6008 IFI<>17737THENPRINT"ERROR IN ML DATA":PRINT"C
HECK FOR TYPO'S":END :rem 36
6010 SYSML :rem 95
6015 PRINT"{CLR}{7 DOWN}{5 RIGHT}6502 ML TRACER
{4 DOWN}" :rem 163
6020 RETURN :rem 168
6999 REM{2 SPACES}> PAUSE < :rem 189
7000 GETA$:IFA$=" "THEN7000 :rem 177
7010 IFA$="I"THEND=B:L=4:GOSUB7100:B=D:GOTO7000
:rem 40
7020 IFA$="A"THEND=PEEK(A):L=2:GOSUB7100:POKEA,D:G
OTO7000 :rem 177
7030 IFA$="X"THEND=PEEK(X):L=2:GOSUB7100:POKEX,D:G
OTO7000 :rem 247
7040 IFA$="Y"THEND=PEEK(Y):L=2:GOSUB7100:POKEY,D:G
OTO7000 :rem 251
7050 IFA$="S"THEND=PEEK(S):L=2:GOSUB7100:POKES,D:G
OTO7000 :rem 234
7060 IFA$="P"THEND=PEEK(P):L=2:GOSUB7100:POKEP,D:G
OTO7000 :rem 226
7070 RETURN :rem 174
7100 PRINTA$="";:GOSUB2000:INPUTH$:H$=RIGHT$(H$,L)
:GOSUB1500:RETURN :rem 124
9000 DATA128,64,1,2 :rem 207
10000 DATABRK ,ORAF,,,,ORAC,ASLC, :rem 142
10001 DATAPHP ,ORAB,ASL ,,,ORAA,ASLA, :rem 112
10002 DATABPLJ,ORAG,,,,ORAH,ASLH, :rem 228
10003 DATACLC ,ORAE,,,,ORAD,ASLD, :rem 133
10004 DATAJSRA,ANDF,,,BITC,ANDC,ROLC, :rem 244
10005 DATAPLP ,ANDB,ROL ,,BITA,ANDA,ROLA, :rem 148
10006 DATAMIJ,ANDG,,,,ANDH,ROLH, :rem 209
10007 DATASEC ,ANDE,,,,AMDD,ROLD, :rem 128
10008 DATARTI ,EORF,,,,EORC,LSRC, :rem 191
```

```

10009 DATAPHA ,EORB,LSR , ,JMPA,EORA,LSRA, :rem 187
10010 DATABVCJ ,EORG, , , ,EORH,LSRH, :rem 249
10011 DATACLI ,EORE, , , ,EORD,LSRD, :rem 163
10012 DATARTS ,ADCF, , , ,ADCC,RORC, :rem 138
10013 DATAPLA ,ADCB,ROR , ,JMPK,ADCA,RORA, :rem 140
10014 DATABVSJ ,ADCG, , , ,ADCH,RORH, :rem 211
10015 DATASEI ,ADCE, , , ,ADCD,RORD, :rem 118
10016 DATA ,STAF, , , ,STYC,STAC,STXC, :rem 36
10017 DATADEY , ,TXA , ,STYA,STAA,STXA, :rem 192
10018 DATABCCJ ,STAG, , , ,STYH,STAH,STXI, :rem 73
10019 DATATYA ,STAE,TXS , , ,STAD, , :rem 143
10020 DATALDYB ,LDAF,LDXB, ,LDYC,LDAC,LDXC, :rem 24
10021 DATATAY ,LDAB,TAX , ,LDYA,LDAALDXA, :rem 149
10022 DATABCSJ ,LDAG, , , ,LDYH,LDAH,LDXI, :rem 248
10023 DATACLV ,LDAE,TSX , ,LDYD,LDAD,LDXE, :rem 173
10024 DATACPYB ,CMPF, , , ,CPYC,CMPC,DECC, :rem 250
10025 DATAINY ,CMPB,DEX , ,CPYA,CMPA,DECA, :rem 148
10026 DATABNEJ ,CMPG, , , ,CMPH,DECH, :rem 201
10027 DATACLD ,CMPE, , , ,CMPD,DECD, :rem 116
10028 DATACPXB ,SBCF, , , ,CPXC,SBCC,INCC, :rem 250
10029 DATAINX ,SBCB,NOP , ,CPXA,SBCA,INCA, :rem 160
10030 DATABEQJ ,SBCG, , , ,SBCI,INCI, :rem 199
10031 DATASED ,SBCE, , , ,SBCD,INCD, :rem 118
10032 DATAEND :rem 231
20000 DATA162,0,181,0,157,0,41,189 :rem 167
20001 DATA0,1,157,0,42,189,0,2 :rem 217
20002 DATA157,0,43,232,208,236,96,120 :rem 68
20003 DATA162,0,181,0,168,189,0,41 :rem 172
20004 DATA149,0,152,157,0,41,189,0 :rem 174
20005 DATA1,168,189,0,42,157,0,1 :rem 75
20006 DATA152,157,0,42,189,0,2,168 :rem 180
20007 DATA189,0,43,157,0,2,152,157 :rem 180
20008 DATA0,43,232,208,213,186,138,174 :rem 125
20009 DATA243,40,154,141,243,40,172,242 :rem 165
20010 DATA40,174,241,40,173,244,40,72 :rem 62
20011 DATA173,240,40,40,234,234,234,8 :rem 62
20012 DATA141,240,40,104,141,244,40,142 :rem 147
20013 DATA241,40,140,242,40,186,138,174 :rem 167
20014 DATA243,40,154,141,243,40,162,0 :rem 56
20015 DATA181,0,168,189,0,41,149,0 :rem 180
20016 DATA152,157,0,41,189,0,1,168 :rem 179
20017 DATA189,0,42,157,0,1,152,157 :rem 179
20018 DATA0,42,189,0,2,168,189,0 :rem 84
20019 DATA43,157,0,2,152,157,0,43 :rem 124
20020 DATA232,208,213,88,96 :rem 100

```

# LIST Freezer

Doug Ferguson

*This very short routine will prove indispensable. It allows you to pause or freeze a program listing on your monitor screen. For any VIC or 64.*

**T**he VIC-20 and Commodore 64 would greatly benefit from some way to pause listings on the screen. When you're writing or debugging a program, especially if you lack a printer, you can waste a lot of time typing LIST again and again just to get a look at your BASIC code.

"LIST Freezer" is an elegant solution to the problem. It patches directly into the LIST routine in ROM without interfering with anything else. Once it's activated, there is never any need to turn it off. It also eliminates the screen ripple effect seen in some other LIST pause routines.

## The LIST Freezer

Start by typing in, saving, and then running LIST Freezer. Because it destroys the BASIC loader part of itself in line 80, be sure to save it before typing RUN for the first time. Then, load a BASIC program and give it a try.

To use LIST Freezer, list any BASIC program and hold down the SHIFT key. The listing will pause. To freeze it entirely and free your hands, press SHIFT LOCK. You can restart the listing at any time by releasing SHIFT or SHIFT LOCK.

## Technical Details

Here's how it works. Line 20 sets the low-byte/high-byte address of a machine language routine at the top of RAM. On either the VIC or the 64, the routine occupies 23 bytes of memory.

Line 30 redefines the computer's memory size to protect the routine from BASIC programming. It also moves the LIST vector at memory addresses 774-775 (\$0306-\$0307) to reroute the indirect jump to ROM (address \$A717 in the 64 or \$C717 in the VIC).

The remaining lines create the patch routine at the top of RAM. Line 50 adjusts the patch to work on either the VIC or 64.

Notice that the program assumes the normal LIST vector at power-up; line 20 thus prevents you from accidentally trying

to activate the routine more than once while the power is on.

Also note that the routine clears out the keyboard buffer when activated. Actually this was necessary only for the VIC. However, it causes no harm on the 64 and was left in to make the routine universal.

### LIST Freezer for the VIC and 64

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

20 L=232:H=PEEK(56)-1:Q=PEEK(775):IF Q<167 THEN 80
                                     :rem 236
30 POKE 55,L:POKE 51,L:POKE 56,H:POKE 52,H:POKE 77
   4,L:POKE 775,H
                                     :rem 74
40 FOR X=L+H*256 TO X+21:READ D:POKE X,D:NEXT
                                     :rem 51
50 POKE X,Q
                                     :rem 105
60 DATA 72,152,72,32,159,255,169,1,44,141,2,208,24
   6
                                     :rem 209
70 DATA 169,0,133,198,104,168,104,76,26
                                     :rem 136
80 NEW
                                     :rem 82
    
```

# REFMAP: A Cross-Reference Map Utility for the Expanded VIC

Kenneth D. Day

*“REFMAP” is a cross-reference map utility written in 6502 machine language for any VIC with 3K or more memory expansion.*

“REFMAP” is a useful utility for any BASIC programmer. When called by a SYS command giving the decimal entry address of the routine, the program prints an alphabetized list of all variables and functions in any current BASIC program, as well as the numbers of the lines in which they appear.

The utility is particularly useful for debugging your own programs or modifying another person’s work. If the BASIC program has been crunched to save memory, or if the program is quite long, this utility can save a programmer from a rather tedious and eye-straining task.

When you run this program, a relocating BASIC loader will locate the utility at the top of available memory and tell you how to call it (for example, with SYS 12288).

Once loaded, the machine language program occupies only 983 bytes. However, the relocating loader program is too large to run within the memory of an unexpanded VIC-20.

## Display Format

REFMAP provides an alphabetical listing of the names of all variables and functions within the current BASIC program and also the line numbers of the line in which the variables or functions occur. For example, assume that the following BASIC program has been stored in memory:

```
10 INPUT X,Y
20 Z=X+Y
30 PRINT X
40 PRINT Y
50 PRINT Z,X
60 END
```

Calling REFMAP would result in the following information being written to the screen:

```

REFMAP
X 10 20 30
  50
Y 10 20 40
Z 20 50
READY.

```

If there is more output than can be printed on the screen at one time, READY will not appear at the bottom of the screen. To display the next page, press RETURN.

The sorting sequence for variable names takes the following order:

<blank> \$ % ( ) 0-9 A-Z

While blanks are not really part of a variable name, variable names are treated as if they were four characters long. For every character less than four in the actual variable name, a blank is added on the end. This assures that shorter names will occur alphabetically before longer names.

The left and right parentheses are not part of any variable name, but are added onto the ends of the names of arrays and functions. An array named N2% will appear in the display as N2%( in order to distinguish it from a nonarray variable by the same name. That makes the program think that three-character array names are four characters long.

Names of functions are displayed with a right parenthesis on the end, so that a function named FNA would appear as A). This is an improvement over many other cross-reference map utilities since it eliminates confusion of arrays and function names.

### Program 1. REFMAP

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

6 L=PEEK(644)*256+PEEK(643)-983:POKE644,INT(L/256)
  :POKE643,L-PEEK(644)*256                                :rem 182
7 FORI=56TO52STEP-2:POKEI,PEEK(644):POKEI-1,PEEK(6
  43):NEXT                                                :rem 110
8 PRINT"{CLR}LOADING REFMAP INTO{3 SPACES}MEMORY"
  :rem 104
9 P=L+16:FORI=1TO25:READS$:FORJ=1TO40:C$=LEFT$(S$,
  2):S$=RIGHT$(S$,80-J*2):V=0                            :rem 66
10 IFLEFT$(C$,1)="X"THEN14                                :rem 193

```

## 5: Utilities and Programming Aids

```
11 D1=ASC(LEFT$(C$,1)):D2=ASC(RIGHT$(C$,1))
12 V=-(D1>64)*(D1-55)*16+-(D1<65)*(D1-48)*16+-(D2>
64)*(D2-55)+-(D2<65)*(D2-48) :rem 108
13 POKEP,V:P=P+1:NEXT:NEXT :rem 39
14 P=P-1:IFP=L+22THEN20 :rem 202
15 IFPEEK(P)<48ORPEEK(P)>51THEN14 :rem 98
16 IFP=L+141ORP=L+151ORP=L+161ORP=L+171ORP=L+181OR
P=L+191ORP=L+201ORP=L+211THEN14 :rem 2
17 IFP=L+300ORP=L+406ORP=L+409ORP=L+410ORP=L+414OR
P=L+418ORP=L+440ORP=L+473THEN14 :rem 203
18 IFP=L+474ORP=L+482ORP=L+486ORP=L+732ORP=L+816OR
P=L+99THEN14 :rem 221
19 V=PEEK(P)*256+PEEK(P-1)+L-12288:POKEP,INT(V/256
):POKEP-1,V-PEEK(P)*256:P=P-1:GOTO14 :rem 85
20 L=L+22:PRINT"{CLR}TYPE ";"{RVS}SYS";L;"{OFF}TO
{SPACE}USE":NEW:END :rem 101
51 DATA50414D464552A2018628A2018629A99320D2FFA91DA
20820D2FFCAD0FAA206BD0F3020D2FFCAD0F7 :rem 93
52 DATAA90D20D2FF20D2FF38A52DE52BC902D00160A9408D0
430A9208D05308D06308D0730A95A8D08308D :rem 20
53 DATA0930A9308D0A308D0B30A52B8526A52C8527A000B12
68D0030C8B1268D0130C8C8C820EF32B06BAD :rem 255
54 DATA0430CD0C303020D0F1AD0530CD0D303016D0E7AD063
0CD0E30300CD0DDAD0730CDF30300210D3AD :rem 12
55 DATA0830CD0C3030CBD01EAD0930CD0D3030C1D014AD0A3
0CD0E3030B7D00AAD0B30CDF3030ADF0ABAD :rem 118
56 DATA0C308D0830AD0D308D0930AD0E308D0A30AD0F308D0
B301890BDAD0430CD0C30D019AD0530CD0D30 :rem 118
57 DATAD011AD0630CD0E30D009AD0730CD0F30D00160AD083
08D0430AD09308D0530AD0A308D0630AD0B30 :rem 44
58 DATA8D0730C930F00BA2008E0C3020F1314C5A306020D83
2AD043020D2FFAD053020D2FFAD063020D2FF :rem 240
59 DATAAD073020D2FFA99DA20420D2FFCAD0FA60C8B126C92
2D002C860C900D0F360C8B126C93AD002C860 :rem 107
60 DATAC900D0F360AD00308526AD01308527A003B12699003
08810F8A00460C9003057C930300EC93A3008 :rem 118
61 DATAC941304BC95A10471860C924F0FAC925F0F6C928D02
2C00730EE888AE0C30F00188B126C8C8AE0C :rem 78
62 DATA30F001C8C9A5F004A928D0D4A929D0D0C9303013C95
B100FC93A3004C9413007C8B126C900D0BB38 :rem 7
63 DATA06E629A628E001F00BA2018628A90D20D2FFE629203
B31A52B8526A52C8527A003B1269900308810 :rem 194
64 DATAF8A004B126C98FD00E207D31AD0130D0F2A90D20D2F
F60C922D0062061311890E2C98ED006206F31 :rem 8
65 DATA1890D8C900F0DACD0430F006C8B1261890CBC8B126D
00DAE0530E020D0C32083331890BD0AE0530E0 :rem 28
66 DATA20F00E209431CD0530F05C20AA311890A4209431B0E
020AA31189097C8B12620AA31C900D00AAE06 :rem 193
```



```
67 DATA30E020D08C1890C6AE0630E020F008CD0630F00A189
   0DB20AA3190D6B0B0C8B126D009AE0730E020 :rem 247
68 DATAD0A7F0A2AE0730E020D006C928D097F098C928D094F
   08FA929CD0530F088D0ACA629E014301020E4 :rem 55
69 DATAFFC900F0F9A99320D2FFA200862960A9208D0C308D0
   D308D0E308D0F30B126C98FF020C983D00620 :rem 60
70 DATA6F311890F0C9001004C81890E8C922D006206131189
   0DEC900D00A207D31AD0130D0D23860C8C941 :rem 184
71 DATA30CBC95B10C78D0C30B126C8C900D00318901920943
   1900218608D0D30C928F0F7C929F0F3B126C8 :rem 244
72 DATAC900D00B207D3118AD0130D001386020AA31B0158D0
   E30C928F00EB126C900F008C928D0048D0F30 :rem 203
73 DATAC8186020D832A629D003203B31A628E001D00AA91DA
   20420D2FFCAD0FAE628A528C904D006A20186 :rem 11
74 DATA28E629AD0330AC02302091D320DDDBD0001C900F00
   3E8D0F6A00688CAD0FCA92020D2FF88D0FAA0 :rem 107
75 DATA01A900201ECB60XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX :rem 52
```

# Data Files for the VIC and 64

Brian Prescott

*You can give yourself more free memory by storing files on tape or disk. The programs presented here show you how to set up, write to, and read from either tape or disk files. For any VIC or 64.*

**B**eginning programmers often balk at handling files with the Datassette. But some serious applications require the use of the same data in more than one program or require the use of several sets of data with the same program. Such situations call for data files.

Here's a trio of simple programs that create data files, read them back, and display the contents. Not only will they help you grasp the techniques involved, but you can also use them to create files or incorporate them into your own programs.

Program 1 and Program 2 create data files. Program 3 reads the files and prints the contents to the screen. Each program can be used with either tape or disk. Tape users should omit the REMs from line 247 in Program 1, line 195 in Program 2, and line 372 in Program 3. Disk users should remove the REMs from line 245 in Program 1, line 590 in Program 2, and line 370 in Program 3.

Program 1 prompts you for each item. It then writes the items onto a data file. This method is convenient. But if an incorrect entry is typed in and stored, the only way to correct it is to create a new file. That means you have to enter all the data again.

Program 2 solves the problem but is slightly less convenient to use. First load the program, then add DATA statements at lines 540-570. Running the program creates the files.

## Creating a File

The programs are fairly straightforward, but a few comments are in order. The first program asks you for the number of items to be in the file and DIMensions a string array to hold them. Be sure to dimension the array to the number of data entries plus one. You're then asked for a filename. It's best to use a name that identifies the file. Using "+1" as the filename

ends the program. After the array is filled and the file written to tape or disk, the program displays the contents on the screen. You could modify the program to allow display and possible editing before the file is created.

The second program does the same job in a slightly different way. The data lines must be organized properly to avoid problems. The first data item will be read as the filename, so be sure the filename is the first item entered. To signal the end of a file, use `-1`. This is included at the end of the DATA statements. You can create several files at one time, as you can see from the data included. To signal the end of data, use `+1`. That stops the program.

To see what's on the files, load and run Program 3. If you're using tape, you can ask for any file, but be sure to rewind the tape to some point before the starting point of the file you want.

## Opening, Filling, and Closing Files

A data file is like a desk drawer. First you open it, then you put something in or take something out, and then you close it.

In the first two programs you will see the statement `OPEN 1,1,1, "filename"`. The three numbers following OPEN serve three different purposes. The first is the file number. You can pick any number from 1 to 127, but 1 is most commonly used. The second is the device number. Tape drives are always device number 1, and single disk drives are always device number 8. The last number is the secondary address, which is important for tape files. A 1 here means "write to the tape file." Thus, `OPEN 1,1,1` tells the computer to open file number 1 on the cassette drive for writing.

Once a file is opened, you can print to it. In the first two programs, you will see `PRINT#1, data`. `PRINT#` works like `PRINT`, except that a question mark (?) cannot be used as an abbreviation. Use `P SHIFT-R` instead, followed by the file number. In addition, you have to put a comma between the file number and the data you are writing. After you finish writing the file, `CLOSE` it.

Opening a file for reading is similar, except that the secondary address is zero. After the file is open, you can `INPUT#` or `GET#` from it. You can read and write any type of data—floating-point numbers, integers, or strings.

Since the size of a data file can vary, it is advisable to

indicate how long the file is or where it ends. One method is to PRINT# the number of records as the first item in the file. This is best when you are setting up arrays. The computer reads the first number in the file, then DIMENSIONS the array. Another way to mark the length of a file is to make up an end-of-file marker. In the sample programs, "-1" acts as the marker.

### Tape Files on a Disk Drive?

Knowing the basics of tape files is helpful even if you decide to buy a disk drive. There are a variety of ways to store information on a disk; one of them is very similar to tape files.

Sequential disk files store information in the order it is received (tape files are always sequential). To transfer information from tape to disk, simply open the tape file for reading, open a sequential disk file for writing, and then input the data from tape, print it to the disk, input more, print more, and so on until you reach the end of the file.

### Program 1. Keyboard Data File Maker

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```
200 REM{11 SPACES}* FILES WRITTEN *           :rem 106
210 REM{11 SPACES}* FROM KEYBOARD *           :rem 80
215 CLR                                         :rem 121
220 INPUT"{CLR}{5 DOWN}NO. OF ITEMS IN FILE";N
                                                :rem 114
225 DIMW$(N+1)                                 :rem 233
230 INPUT"{DOWN}FILENAME";NAME$:IFNAME$="+1"THEN E
ND                                             :rem 44
240 PRINT"{DOWN}ON THE PROMPT,":PRINT"TYPE EACH IT
EM,":PRINT"FOLLOWED BY {RVS}RETURN{OFF}"
                                                :rem 67
245 REM OPEN1,8,1,NAME$+",S,W"                 :rem 133
247 REM OPEN1,1,1,NAME$                       :rem 15
250 FORX=1TON:INPUTW$(X):PRINT#1,W$(X):IFW$(X)<>"-
1"THEN NEXT                                  :rem 5
260 CLOSE1:FORX=0TON:PRINTW$(X):NEXT          :rem 122
265 PRINT"HIT ANY KEY"                        :rem 36
270 GETA$:IFA$=""THEN 270                    :rem 85
280 GOTO200                                    :rem 101
290 END                                        :rem 114
```

## Program 2. Improved Tape Data File Maker

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

500 REM{10 SPACES}* FILE WRITTEN *           :rem 26
510 REM{10 SPACES}*{2 SPACES}FROM DATA{3 SPACES}*
                                           :rem 28
520 REM                                         :rem 123
540 DATAW21,DELIGHT,CHAPTER,FARTHER,BUILT,JOYFUL,S
    TORIES                                     :rem 11
541 DATABOUGHT,SCARF,FILLED,SAILED,REMAIN :rem 186
550 DATACLOAK,PLACED,DRIVING,FEAST,STRONG,-1,W22,F
    LYING                                       :rem 64
551 DATASOMETIMES,HIGHWAY,SNOWING,CLOSING   :rem 52
560 DATABEDTIME,PUSHED,BRUSHES,DREAMING,BOOKCASE,P
    ULLED                                       :rem 196
561 DATAAIRPLANE,BUYING,SPENDING,SKATED    :rem 104
570 DATADECEMBER,-1,+1                       :rem 255
580 READNAME$:R$=CHR$(13)                     :rem 161
585 IFNAME$="+1"THENGOTO650                   :rem 86
590 REM OPEN1,8,1,NAME$+",S,W"               :rem 136
595 REM OPEN1,1,1,NAME$                       :rem 21
600 READWRD$:PRINT#1,WRD$;R$                 :rem 204
620 IFWRD$<>"-1"THEN600                      :rem 25
630 CLOSE1                                    :rem 64
640 GOTO580                                    :rem 112
650 CLOSE1:END                                :rem 83

```

## Program 3. File Reader

*For error-free program entry, be sure to use "The Automatic Proofreader," Appendix C.*

```

300 REM{11 SPACES}*{4 SPACES}FILE{4 SPACES}*
                                           :rem 235
310 REM{11 SPACES}*{3 SPACES}READER{3 SPACES}*
                                           :rem 127
320 DIM W$(20)                                :rem 157
350 INPUT"{CLR}{8 DOWN}WHAT FILE";NAME$     :rem 91
360 IFNAME$="+1"THEN END                       :rem 80
362 INPUT "HOW MANY ITEMS";IT                :rem 236
370 REM FOR DISK OPEN1,8,0,NAME$+",S,R"      :rem 144
372 REM FOR TAPE OPEN1,1,0,NAME$             :rem 30
374 FOR X=1 TO IT                             :rem 139
380 INPUT#1,W$(X)                             :rem 207
410 NEXT:CLOSE1:FOR X=1 TO IT                 :rem 220
415 PRINT "ITEM # "X,W$(X):NEXT              :rem 222
420 PRINT"ANOTHER FILE?"                     :rem 215
430 GETA$:IFA$=""THEN430                     :rem 81
450 IF A$="Y" THEN RUN                         :rem 139
460 END                                        :rem 113

```

□ □ □ □ □

□ □ □ □ □

# Appendices

---

□ □ □ □ □

□ □ □ □ □



# A Beginner's Guide to Typing In Programs

---

## What Is a Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential*, but without a program, it isn't going anywhere. Most of the programs published in this book are written in a computer language called BASIC. BASIC is easy to learn and is built into all Commodore 64s.

## BASIC Programs

This book includes programs for both the VIC and the 64. Be sure that you type in only those programs written for your machine; don't type in the 64 version if you have a VIC-20.

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one right way of stating something. Every letter, character, or number is significant. A common mistake is substituting a letter such as O for the numeral 0, a lowercase l for the numeral 1, or an uppercase B for the numeral 8. Also, you must enter all punctuation, such as colons and commas, just as it appears in the book. Spacing can be important. To be safe, type in the listings *exactly* as they appear.

## Braces and Special Characters

The exception to this typing rule is when you see the braces, such as {DOWN}. Anything within a set of braces is a special character or characters that cannot easily be listed on a printer. When you come across such a special statement, refer to "How to Type In Programs" (Appendix B).

## About DATA Statements

Some programs contain a section or sections of DATA statements. These lines provide information needed by the program. Some DATA statements contain actual programs (in machine language), while others may contain graphics codes. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could lock up or crash. The keyboard and STOP key may seem dead, and the screen may go blank. But

don't panic. No damage has been done. To regain control, turn off your computer and then turn it back on. This will erase whatever program was in memory, *so always save a copy of your program before you run it.* If your computer crashes, you can load the program and look for your mistake.

Sometimes a mistyped DATA statement will cause an error message when the program is run. The error message may refer to the program line that READs the data. *However, the error is still in the DATA statements.*

### **Get to Know Your Machine**

You should familiarize yourself with your computer before attempting to type in a program. Learn the statements you use to store and retrieve programs from tape or disk. You'll want to save a copy of your program, so that you won't have to type it in every time you want to use it. Learn to use your machine's editing functions. How do you change a line if you made a mistake? You can always retype the line, but you should at least know how to backspace. Do you know how to enter reverse-video, lowercase, and control characters? It's all explained in your manual.

In order to insure accurate entry of each program line, we have included a checksum program. Please read "The Automatic Proofreader" (Appendix C) before typing in any of the programs in this book.

### **A Quick Review**

1. Type in the program a line at a time, in order. Press RETURN at the end of each line. Use backspace or the back arrow to correct mistakes.
2. Check the line you've typed against the line in the book. You can check the entire program again if you get an error when you run the program.

# How to Type In Programs

---

Many of the programs in this book contain special control characters (cursor controls, color keys, reverse video, etc.). To make it easy to know exactly what to type when entering one of these programs into your computer, we have established the following listing conventions.

Generally, VIC or 64 program listings will contain words within braces which spell out any special characters: {DOWN} would mean to press the cursor down key. {5 SPACES} would mean to press the space bar five times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listings. For example, S would mean to type the S key while holding the SHIFT key. This would appear on your screen as a heart symbol. If you find an underlined key enclosed in braces (for example, {10 N}), you should type the key as many times as indicated. In that case, you would enter ten shifted N's.

If a key is enclosed in special brackets, [`<` `>`], you should hold down the *Commodore key* while pressing the key inside the special brackets. (The Commodore key is the key in the lower-left corner of the keyboard.) Again, if the key is preceded by a number, you should press the key as many times as necessary.

Rarely, in programs for the 64, you'll see a solitary letter of the alphabet enclosed in braces. These characters can be entered by holding down the CTRL key while typing the letter in the braces. For example, {A} would indicate that you should press CTRL-A. You should never have to enter such a character on the VIC.

## Quote Mode

You know that you can move the cursor around the screen with the CRSR keys. Sometimes a programmer will want to move the cursor under program control. That's why you see all the {LEFT}'s, {HOME}'s, and {BLU}'s in our programs. The only way the computer can tell the difference between direct and programmed cursor control is the quote mode.

Once you press the quote (the double quote, SHIFT-2), you are in the quote mode. If you type something and then try to change it by moving the cursor left, you'll only get a bunch

of reverse-video lines. These are the symbols for cursor left. The only editing key that isn't programmable is the DEL key; you can still use DEL to back up and edit the line. Once you type another quote, you are out of quote mode.

You also go into quote mode when you INSerT spaces into a line. In any case, the easiest way to get out of quote mode is to just press RETURN. You'll then be out of quote mode and you can cursor up to the mistyped line and fix it.

In order to insure accurate entry of each program line, we have included a checksum program. Please read "The Automatic Proofreader" (Appendix C) before typing in any of the programs in this book.

Refer to the following table when entering cursor and color control keys:

When You Read:	Press:	See:	When You Read:	Press:	See:
{CLR}	SHIFT CLR/HOME		{1}	COMMODORE 1	
{HOME}	CLR/HOME		{2}	COMMODORE 2	
{UP}	SHIFT		{3}	COMMODORE 3	
{DOWN}			{4}	COMMODORE 4	
{LEFT}	SHIFT		{5}	COMMODORE 5	
{RIGHT}			{6}	COMMODORE 6	
{RVS}	CTRL 9		{7}	COMMODORE 7	
{OFF}	CTRL 0		{8}	COMMODORE 8	
{BLK}	CTRL 1		{F1}	f1	
{WHT}	CTRL 2		{F2}	SHIFT f1	
{RED}	CTRL 3		{F3}	f3	
{CYN}	CTRL 4		{F4}	SHIFT f3	
{PUR}	CTRL 5		{F5}	f5	
{GRN}	CTRL 6		{F6}	SHIFT f5	
{BLU}	CTRL 7		{F7}	f7	
{YEL}	CTRL 8		{F8}	SHIFT f7	
				SHIFT	

# The Automatic Proofreader

Charles Brannon

“The Automatic Proofreader” will help you type in program listings without typing mistakes. It is a short error-checking program that hides itself in memory. When activated, it lets you know immediately after typing a line from a program listing if you have made a mistake. Please read these instructions carefully before typing any programs in this book.

## Preparing the Proofreader

1. Using the listing below, type in the Proofreader. Be very careful when entering the DATA statements—don't type an l instead of a 1, an O instead of a 0, extra commas, etc.
2. Save the Proofreader on tape or disk at least twice *before running it for the first time*. This is very important because the Proofreader erases part of itself when you first type RUN.
3. After the Proofreader is saved, type RUN. It will check itself for typing errors in the DATA statements and warn you if there's a mistake. Correct any errors and save the corrected version. Keep a copy in a safe place—you'll need it again and again, every time you enter a program from this book, *COMPUTE!'s Gazette* or *COMPUTE!* magazine.
4. When a correct version of the Proofreader is run, it activates itself. You are now ready to enter a program listing. If you press RUN/STOP-RESTORE, the Proofreader is disabled. To reactivate it, just type the command SYS 886 and press RETURN.

## Using the Proofreader

All listings in this book have a *checksum number* appended to the end of each line, for example, *:rem 123*. *Don't enter this statement when typing in a program*. It is just for your information. The rem makes the number harmless if someone does type it in. It will, however, use up memory if you enter it, and it will confuse the Proofreader, even if you entered the rest of the line correctly.

When you type in a line from a program listing and press RETURN, the Proofreader displays a number at the top of your screen. *This checksum number must match the checksum number in the printed listing*. If it doesn't, it means you typed the line differently than the way it is listed. Immediately

recheck your typing. Remember, don't type the rem statement with the checksum number; it is published only so you can check it against the number which appears on your screen.

The Proofreader is not picky about spaces. It will not notice extra spaces or missing ones. This is for your convenience, since spacing is generally not important. But occasionally proper spacing *is* important, so be extra careful with spaces.

Due to the nature of a checksum, the Proofreader will not catch all errors. Since  $1 + 3 + 5 = 3 + 1 + 5$ , the Proofreader cannot catch errors of transposition. Thus, the Proofreader will not notice if you type GOTO 385 where you mean GOTO 835. In fact, you could type in the line in any order and the Proofreader wouldn't notice. The Proofreader should help you catch most typing mistakes, but keep this in mind if a program that checks out with the Proofreader still seems to have errors.

There's another thing to watch out for: If you enter the line by using abbreviations for commands, the checksum will not match up. But there is a way to make the Proofreader check it. After entering the line, LIST it. This eliminates the abbreviations. Then move the cursor up to the line and press RETURN. It should now match the checksum. You can check whole groups of lines this way.

### Special Tape SAVE Instructions

When you're through typing in a listing, you must disable the Proofreader before saving the program on tape. Disable the Proofreader by pressing RUN/STOP-RESTORE (hold down the RUN/STOP key and sharply hit the RESTORE key). This procedure is not necessary for disk SAVES, *but you must disable the Proofreader this way before a tape SAVE.*

SAVE to tape erases the Proofreader from memory, so you'll have to load and run it again if you want to type another listing. SAVE to disk does not erase the Proofreader.

### Hidden Perils

The Proofreader's home in memory is not a very safe haven. Since the cassette buffer is wiped out during tape operations, you need to disable the Proofreader with RUN/STOP-RESTORE before you save your program. This applies only to tape use. Disk users have nothing to worry about.

Not so for 64 owners with tape drives. What if you type

in a program in several sittings? The next day, you come to your computer, load and run the Proofreader, then try to load the partially completed program so you can add to it. But since the Proofreader is trying to hide in the cassette buffer, it is wiped out!

What you need is a way to load the Proofreader after you've loaded the partial program. The problem is, a tape LOAD to the buffer destroys what it's supposed to load.

After you've typed in and run the Proofreader, enter the following lines in direct mode (without line numbers) exactly as shown:

```
A$="PROOFREADER.T": B$="{10 SPACES}": FOR X = 1
  TO 4: A$=A$+B$: NEXTX
FOR X = 886 TO 1018: A$=A$+CHR$(PEEK(X)): NEXTX
OPEN 1, 1,1,A$:CLOSE1
```

After you enter the last line, you will be asked to press record and play on your cassette recorder. Put this program at the beginning of a new tape. This gives you a new way to load the Proofreader. Anytime you want to bring the Proofreader into memory without disturbing anything else, put the cassette in the tape drive, rewind, and enter:

```
OPEN1:CLOSE1
```

You can now start the Proofreader by typing SYS 886. To test this, PRINT PEEK (886) should return the number 173. If it does not, repeat the steps above, making sure that A\$ ("PROOFREADER.T") contains 13 characters and that B\$ contains 10 spaces.

You can now reload the Proofreader into memory whenever LOAD or SAVE destroys it, restoring your personal typing helper.

## The Automatic Proofreader

```
100 PRINT "{CLR}PLEASE WAIT...":FOR I=886 TO 1018:READ
  A:CK=CK+A:POKE I,A:NEXT
110 IF CK<>17539 THEN PRINT "{DOWN}YOU MADE AN ERRO
  R":PRINT "IN DATA STATEMENTS.":END
120 SYS886:PRINT "{CLR}{2 DOWN}PROOFREADER ACTIVATE
  D.":NEW
886 DATA 173,036,003,201,150,208
892 DATA 001,096,141,151,003,173
898 DATA 037,003,141,152,003,169
904 DATA 150,141,036,003,169,003
```

## Appendix C

---

910 DATA 141,037,003,169,000,133  
916 DATA 254,096,032,087,241,133  
922 DATA 251,134,252,132,253,008  
928 DATA 201,013,240,017,201,032  
934 DATA 240,005,024,101,254,133  
940 DATA 254,165,251,166,252,164  
946 DATA 253,040,096,169,013,032  
952 DATA 210,255,165,214,141,251  
958 DATA 003,206,251,003,169,000  
964 DATA 133,216,169,019,032,210  
970 DATA 255,169,018,032,210,255  
976 DATA 169,058,032,210,255,166  
982 DATA 254,169,000,133,254,172  
988 DATA 151,003,192,087,208,006  
994 DATA 032,205,189,076,235,003  
1000 DATA 032,205,221,169,032,032  
1006 DATA 210,255,032,210,255,173  
1012 DATA 251,003,133,214,076,173  
1018 DATA 003



# Using the Machine Language Editor: MLX

---

By Charles Brannon

Remember the last time you typed in the BASIC loader for a long machine language program? You typed in hundreds of numbers and commas. Even then, you couldn't be sure if you typed it in right. So you went back, proofread, tried to run the program, crashed, went back again, proofread, corrected a few typing errors, ran again, crashed again, rechecked your typing . . . .

Frustrating, wasn't it?

Now, "MLX" comes to the rescue. MLX makes it easy to enter all those long machine language programs with a minimum of fuss. It lets you enter the numbers from a special list that looks similar to DATA statements, and it checks your typing on a line-by-line basis. It won't let you enter illegal characters when you should be typing numbers. It won't let you enter numbers greater than 255. It will prevent you from entering the numbers on the wrong line. In short, MLX will make proofreading obsolete.

## Tape or Disk Copies

In addition, MLX will generate a ready-to-use tape or disk copy of your machine language program. You can then use the LOAD command to read the program into the computer, just like you would with a BASIC program. Specifically, you enter LOAD "*program name*",1,1 (for tape) or LOAD "*program name*",8,1 (for disk).

To start the program, you need to enter a SYS command that transfers control from BASIC to your machine language program. The starting SYS will always be given in the article which presents the machine language program in MLX format.

## Using MLX

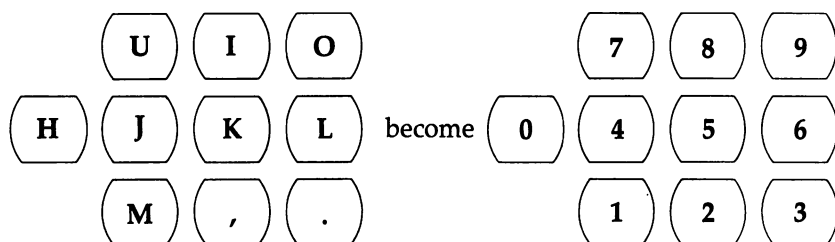
Type in and save MLX (you'll want to use it in the future). When you're ready to type in the machine language program, run MLX. MLX will ask you for two numbers: the starting address and the ending address. Then you'll get a prompt showing the specified starting address; that tells you to type in the corresponding first line of the program.

Subsequent prompts will ask you to type in subsequent lines from the MLX listing. Each line is six numbers plus a checksum. If you enter any of the six numbers wrong, or enter the checksum wrong, the 64 will sound a buzzer and prompt you to reenter the entire line. If you enter the line correctly, a pleasant bell tone will sound and you may go on to enter the next line.

### A Special Editor

You are not using the normal 64 BASIC editor with MLX. For example, it will only accept numbers as input. If you make a typing error, press the INST/DEL key; the entire number is deleted. You can press it as many times as necessary, back to the start of the line. If you enter three-digit numbers as listed, the computer automatically prints the comma and goes on to accept the next number. If you enter less than three digits, you can press either the space bar or RETURN key to advance to the next number. The checksum automatically appears in reverse video for emphasis.

To make it even easier to enter these numbers, MLX redefines part of the keyboard as a numeric keypad (lines 581-584).



When testing it, I've found MLX to be an extremely easy way to enter long listings. With the audio cues provided, you don't even have to look at the screen if you're a touch-typist.

## Done at Last!

When you get through typing, assuming you type your machine language program all in one session, you can then save the completed and bug-free program to tape or disk. Follow the instructions displayed on the screen. If you get any error messages while saving, you probably have a bad disk, a full disk, or a typo in MLX. Sorry, MLX can't check itself!

## Command Control

What if you don't want to enter the whole program in one sitting? MLX lets you enter as much as you want, save the completed portion, and then reload your work from tape or disk when you want to continue. MLX recognizes these commands:

**SHIFT-S:** Save

**SHIFT-L:** Load

**SHIFT-N:** New Address

**SHIFT-D:** Display

Hold down SHIFT while you press the appropriate key. You will jump out of the line you've been typing, so I recommend you do it at a prompt. Use the Save command to store what you've been working on. It will write the tape or disk file as if you've finished. Remember what address you stop on. Then, the next time you run MLX, answer all the prompts as you did before and insert the disk or tape containing the stored file. When you get the entry prompt, press SHIFT-L to reload the file into memory. You'll then use the New Address command (SHIFT-N) to resume typing.

## New Address and Display

After you press SHIFT-N, enter the address where you previously stopped. The prompt will change and you can continue typing. Always enter a New Address that matches up with one of the line numbers in the special listing, or else the checksums won't match up. You can use the Display command to display a section of your typing. After you press SHIFT-D, enter two addresses within the line number range of the listing. You can stop the display by pressing any key.

## Tricky Stuff

You can use the Save and Load commands to make copies of the complete machine language program. Use the Load

## Appendix D

---

command to reload the tape or disk, then insert a new tape or disk and use the Save command to create a new copy.

One quirk about tapes made with the MLX Save command: When you load them, the message "FOUND program" may appear twice. The tape will load just fine, however.

Programmers will find MLX to be an interesting program which protects the user from most typing mistakes. Some screen formatting techniques are also used. Most interesting is the use of ROM Kernal routines for loading and saving blocks of memory. To use these routines, just POKE the starting address (low byte/high byte) into memory locations 251 and 252, and POKE the ending address into locations 254 and 255. Any error code for the SAVE or LOAD can be found in location 253 (an error would be a code less than ten).

I hope you will find MLX to be a true labor-saving program. Since it has been tested by entering actual programs, you can count on it as an aid for generating bug-free machine language. Be sure to save MLX; it will be used for future applications in other COMPUTE! books.

### MLX

```
100 PRINT "{CLR}[6]";CHR$(142);CHR$(8);:POKE53281,1
      :POKE53280,1                               :rem 67
101 POKE 788,52:REM DISABLE RUN/STOP           :rem 119
110 PRINT "{RVS}{39 SPACES}";                  :rem 176
120 PRINT "{RVS}{14 SPACES}{RIGHT}{OFF}[*]£[RVS]
      {RIGHT}{RIGHT}{2 SPACES}[*]{OFF}[*]£[RVS]£
      {RVS}{14 SPACES}";                       :rem 250
130 PRINT "{RVS}{14 SPACES}{RIGHT} [G]{RIGHT}
      {2 RIGHT} {OFF}£[RVS]£[*]{OFF}[*]{RVS}
      {14 SPACES}";                             :rem 35
140 PRINT "{RVS}{41 SPACES}"                   :rem 120
200 PRINT "{2 DOWN}{PUR}{BLK}{9 SPACES}MACHINE LANG
      UAGE EDITOR{5 DOWN}"                     :rem 6
210 PRINT "[5]{2 UP}STARTING ADDRESS?{8 SPACES}
      {9 LEFT}";                               :rem 143
215 INPUTS:F=1-F:C$=CHR$(31+119*F)             :rem 166
220 IFS<256OR(S>40960ANDS<49152)ORS>53247THENGOSUB
      3000:GOTO210                             :rem 235
225 PRINT:PRINT:PRINT                         :rem 180
230 PRINT "[5]{2 UP}ENDING ADDRESS?{8 SPACES}
      {9 LEFT}";:INPUTE:F=1-F:C$=CHR$(31+119*F)
      :rem 20
240 IFE<256OR(E>40960ANDE<49152)ORE>53247THENGOSUB
      3000:GOTO230                             :rem 183
```

```

250 IFE<STHENPRINTC$;"{RVS}ENDING < START
    {2 SPACES}":GOSUB1000:GOTO 230           :rem 176
260 PRINT:PRINT:PRINT                       :rem 179
300 PRINT "{CLR}";CHR$(14):AD=S:POKEV+21,0 :rem 225
310 A=1:PRINTRIGHT$("0000"+MID$(STR$(AD),2),5);":
    ;                                         :rem 33
315 FORJ=ATO6                               :rem 33
320 GOSUB570:IFN=-1 THENJ=J+N:GOTO320      :rem 228
390 IFN=-211 THEN 710                       :rem 62
400 IFN=-204 THEN 790                       :rem 64
410 IFN=-206 THENPRINT:INPUT "{DOWN}ENTER NEW ADDRESS
    S";ZZ                                     :rem 44
415 IFN=-206 THENIFZZ<SORZZ>ETHENPRINT "{RVS}OUT OF
    {SPACE}RANGE":GOSUB1000:GOTO410        :rem 225
417 IFN=-206 THENAD=ZZ:PRINT:GOTO310       :rem 238
420 IF N<>-196 THEN 480                     :rem 133
430 PRINT:INPUT "DISPLAY:FROM";F:PRINT,"TO";:INPUTT
    ;                                         :rem 234
440 IFF<SORF>EORT<SORT>ETHENPRINT"AT LEAST";S;
    {LEFT}, NOT MORE THAN";E:GOTO430      :rem 159
450 FORI=FTOTSTEP6:PRINT:PRINTRIGHT$("0000"+MID$(S
    TR$(I),2),5);":";                       :rem 30
451 FORK=0TO5:N=PEEK(I+K):PRINTRIGHT$("00"+MID$(ST
    R$(N),2),3);":";                       :rem 66
460 GETA$:IFA$>" " THENPRINT:PRINT:GOTO310 :rem 25
470 NEXTK:PRINTCHR$(20);:NEXTI:PRINT:PRINT:GOTO310
    ;                                         :rem 50
480 IFN<0 THEN PRINT:GOTO310                :rem 168
490 A(J)=N:NEXTJ                            :rem 199
500 CKSUM=AD-INT(AD/256)*256:FORI=1TO6:CKSUM=(CKSU
    M+A(I))AND255:NEXT                      :rem 200
510 PRINTCHR$(18);:GOSUB570:PRINTCHR$(146);:rem 94
511 IFN=-1 THENA=6:GOTO315                 :rem 254
515 PRINTCHR$(20):IFN=CKSUMTHEN530         :rem 122
520 PRINT:PRINT"LINE ENTERED WRONG : RE-ENTER":PRI
    NT:GOSUB1000:GOTO310                    :rem 176
530 GOSUB2000                               :rem 218
540 FORI=1TO6:POKEAD+I-1,A(I):NEXT:POKE54272,0:POK
    E54273,0                                 :rem 227
550 AD=AD+6:IF AD<E THEN 310               :rem 212
560 GOTO 710                                :rem 108
570 N=0:Z=0                                 :rem 88
580 PRINT"{}";                             :rem 81
581 GETA$:IFA$=" " THEN581                  :rem 95
582 AV=- (A$="M")-2*(A$=",")-3*(A$=".")-4*(A$="J")-
    5*(A$="K")-6*(A$="L")                   :rem 41
583 AV=AV-7*(A$="U")-8*(A$="I")-9*(A$="O"):IFA$="H
    " THENA$="0"                             :rem 134
584 IFAV>0 THENA$=CHR$(48+AV)              :rem 134

```

## Appendix D

```

585 PRINTCHR$(20);:A=ASC(A$):IFA=13ORA=44ORA=32THE
    N670                                     :rem 229
590 IFA>128THENN=-A:RETURN                  :rem 137
600 IFA<>20 THEN 630                         :rem 10
610 GOSUB690:IFI=1ANDT=44THENN=-1:PRINT"{OFF}
    {LEFT} {LEFT}";:GOTO690                 :rem 62
620 GOTO570                                  :rem 109
630 IFA<48ORA>57THEN580                     :rem 105
640 PRINTA$;:N=N*10+A-48                   :rem 106
650 IFN>255 THEN A=20:GOSUB1000:GOTO600    :rem 229
660 Z=Z+1:IFZ<3THEN580                     :rem 71
670 IFZ=0THENGOSUB1000:GOTO570            :rem 114
680 PRINT", ";:RETURN                       :rem 240
690 S%=PEEK(209)+256*PEEK(210)+PEEK(211)  :rem 149
691 FORI=1TO3:T=PEEK(S%-I)                 :rem 67
695 IFT<>44ANDT<>58THENPOKES%-I,32:NEXT    :rem 205
700 PRINTLEFT$("{3 LEFT}",I-1);:RETURN     :rem 7
710 PRINT"{CLR}{RVS}*** SAVE ***{3 DOWN}" :rem 236
715 PRINT"{2 DOWN}({PRESS}{RVS}RETURN{OFF} ALONE TO
    CANCEL SAVE){DOWN}"                   :rem 106
720 F$="":INPUT"{DOWN} FILENAME";F$:IFF$=""THENPRI
    NT:PRINT:GOTO310                       :rem 71
730 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR {RVS}D
    {OFF}ISK: (T/D)"                      :rem 228
740 GETA$:IFAS<>"T"ANDA$<>"D"THEN740      :rem 36
750 DV=1-7*(A$="D"):IFDV=8THENF$="0: "+F$:OPEN15,8,
    15,"S"+F$:CLOSE15                     :rem 212
760 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$):POKE782
    ,ZK/256                                :rem 3
762 POKE781,ZK-PEEK(782)*256:POKE780,LEN(T$):SYS65
    469                                    :rem 109
763 POKE780,1:POKE781,DV:POKE782,1:SYS65466:rem 69
765 K=S:POKE254,K/256:POKE253,K-PEEK(254)*256:POKE
    780,253                                :rem 17
766 K=E+1:POKE782,K/256:POKE781,K-PEEK(782)*256:SY
    S65496                                  :rem 235
770 IF(PEEK(783)AND1)OR(191ANDST)THEN780  :rem 111
775 PRINT"{DOWN}DONE.{DOWN}":GOTO310      :rem 113
780 PRINT"{DOWN}ERROR ON SAVE.{2 SPACES}TRY AGAIN.
    ":IFDV=1THEN720                       :rem 171
781 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$;E2$:CLOS
    E15:GOTO720                             :rem 103
790 PRINT"{CLR}{RVS}*** LOAD ***{2 DOWN}" :rem 212
795 PRINT"{2 DOWN}({PRESS}{RVS}RETURN{OFF} ALONE TO
    CANCEL LOAD)"                         :rem 82
800 F$="":INPUT"{2 DOWN} FILENAME";F$:IFF$=""THENP
    RINT:GOTO310                           :rem 144
810 PRINT:PRINT"{2 DOWN}{RVS}T{OFF}APE OR {RVS}D
    {OFF}ISK: (T/D)"                      :rem 227

```

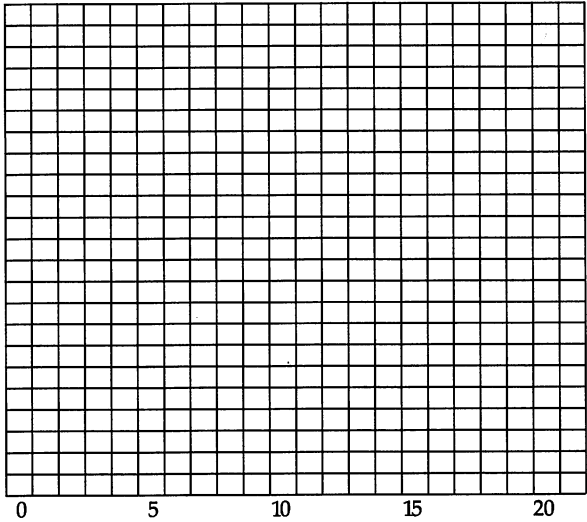
```
820 GETA$:IFA$<>"T"ANDA$<>"D"THEN820 :rem 34
830 DV=1-7*(A$="D"):IFDV=8THENF$="0:"+F$ :rem 157
840 T$=F$:ZK=PEEK(53)+256*PEEK(54)-LEN(T$):POKE782
,ZK/256 :rem 2
841 POKE781,ZK-PEEK(782)*256:POKE780,LEN(T$):SYS65
469 :rem 107
845 POKE780,1:POKE781,DV:POKE782,1:SYS65466:rem 70
850 POKE780,0:SYS65493 :rem 11
860 IF(PEEK(783)AND1)OR(191ANDST)THEN870 :rem 111
865 PRINT"{DOWN}DONE.":GOTO310 :rem 96
870 PRINT"{DOWN}ERROR ON LOAD.{2 SPACES}TRY AGAIN.
{DOWN}":IFDV=1THEN800 :rem 172
880 OPEN15,8,15:INPUT#15,E1$,E2$:PRINTE1$,E2$:CLOS
E15:GOTO800 :rem 102
1000 REM BUZZER :rem 135
1001 POKE54296,15:POKE54277,45:POKE54278,165
:rem 207
1002 POKE54276,33:POKE 54273,6:POKE54272,5 :rem 42
1003 FORT=1TO200:NEXT:POKE54276,32:POKE54273,0:POK
E54272,0:RETURN :rem 202
2000 REM BELL SOUND :rem 78
2001 POKE54296,15:POKE54277,0:POKE54278,247
:rem 152
2002 POKE 54276,17:POKE54273,40:POKE54272,0:rem 86
2003 FORT=1TO100:NEXT:POKE54276,16:RETURN :rem 57
3000 PRINTC$;"{RVS}NOT ZERO PAGE OR ROM":GOTO1000
:rem 89
```

# Screen Location Table (VIC)

---

**Row**

- 0 7680 (4096)
- 7702 (4118)
- 7724 (4140)
- 7746 (4162)
- 7768 (4184)
- 5 7790 (4206)
- 7812 (4228)
- 7834 (4250)
- 7856 (4272)
- 7878 (4294)
- 10 7900 (4316)
- 7922 (4338)
- 7944 (4360)
- 7966 (4382)
- 7988 (4404)
- 15 8010 (4426)
- 8032 (4448)
- 8054 (4470)
- 8076 (4492)
- 8098 (4514)
- 20 8120 (4536)
- 8142 (4558)
- 22 8164 (4580)



**Column**

Note: Numbers in parentheses are for VICs with 8K or more of memory expansion.



# Screen Location Table (64)

---

Row	Column
0	1024
	1064
	1104
	1144
	1184
5	1224
	1264
	1304
	1344
	1384
10	1424
	1464
	1504
	1544
	1584
15	1624
	1664
	1704
	1744
	1784
20	1824
	1864
	1904
	1944
24	1984

# Screen Color Memory Table (VIC)

---

**Row**

- 0 38400 (37888)
- 38422 (37910)
- 38444 (37932)
- 38466 (37954)
- 38488 (37976)
- 5 38510 (37998)
- 38532 (38020)
- 38554 (38042)
- 38576 (38064)
- 38598 (38086)
- 10 38620 (38108)
- 38642 (38130)
- 38664 (38152)
- 38686 (38174)
- 38708 (38196)
- 15 38730 (38218)
- 38752 (38240)
- 38774 (38262)
- 38796 (38284)
- 38818 (38306)
- 20 38840 (38328)
- 38862 (38350)
- 22 38884 (38372)

**Column**

Note: Numbers in parentheses are for VICs with 8K or more of memory expansion.

# Screen Color Memory Table (64)

---

Row	Column
0	55296
	55336
	55376
	55416
5	55456
	55496
	55536
	55576
	55616
10	55656
	55696
	55736
	55776
	55816
15	55856
	55896
	55936
	55976
	56016
20	56056
	56096
	56136
	56176
24	56216
	56256

# Screen Color Codes

---

Color:	Black	White	Red	Cyan	Purple	Green	Blue	Yellow
Code:	0	1	2	3	4	5	6	7

## Additional Color Codes for 64

Color:	Orange	Brown	Light Red	Dark Gray	Medium Gray	Light Green	Light Blue	Light Gray
Code:	8	9	10	11	12	13	14	15

# Screen and Border Colors (VIC Only)

---


































Screen	Border							
	Black	White	Red	Cyan	Purple	Green	Blue	Yellow
Black	8	9	10	11	12	13	14	15
White	24	25	26	27	28	29	30	31
Red	40	41	42	43	44	45	46	47
Cyan	56	57	58	59	60	61	62	63
Purple	72	73	74	75	76	77	78	79
Green	88	89	90	91	92	93	94	95
Blue	104	105	106	107	108	109	110	111
Yellow	120	121	122	123	124	125	126	127
Orange	136	137	138	139	140	141	142	143
Light Orange	152	153	154	155	156	157	158	159
Pink	168	169	170	171	172	173	174	175
Light Cyan	184	185	186	187	188	189	190	191
Light Purple	200	201	202	203	204	205	206	207
Light Green	216	217	218	219	220	221	222	223
Light Blue	232	233	234	235	236	237	238	239
Light Yellow	248	249	250	251	252	253	254	255












































































To set screen and border colors, select the desired combination from the table above and POKE the corresponding value into location 36879.

# ASCII Codes

---

ASCII	CHARACTER	ASCII	CHARACTER
5	WHITE	50	2
8	DISABLE	51	3
	SHIFT COMMODORE	52	4
9	ENABLE	53	5
	SHIFT COMMODORE	54	6
13	RETURN	55	7
14	LOWERCASE	56	8
17	CURSOR DOWN	57	9
18	REVERSE-VIDEO ON	58	:
19	HOME	59	;
20	DELETE	60	<
28	RED	61	=
29	CURSOR RIGHT	62	>
30	GREEN	63	?
31	BLUE	64	@
32	SPACE	65	A
33	!	66	B
34	"	67	C
35	#	68	D
36	\$	69	E
37	%	70	F
38	&	71	G
39	'	72	H
40	(	73	I
41	)	74	J
42	*	75	K
43	+	76	L
44	,	77	M
45	-	78	N
46	.	79	O
47	/	80	P
48	0	81	Q
49	1	82	R

ASCII	CHARACTER	ASCII	CHARACTER
83	S	120	
84	T	121	
85	U	122	
86	V	123	
87	W	124	
88	X	125	
89	Y	126	
90	Z	127	
91	[	129	ORANGE
92	£	133	f1
93	]	134	f3
94	↑	135	f5
95	←	136	f7
96		137	f2
97		138	f4
98		139	f6
99		140	f8
100		141	SHIFTED RETURN
101		142	UPPERCASE
102		144	BLACK
103		145	CURSOR UP
104		146	REVERSE-VIDEO OFF
105		147	CLEAR SCREEN
106		148	INSERT
107		149	BROWN
108		150	LIGHT RED
109		151	GRAY 1
110		152	GRAY 2
111		153	LIGHT GREEN
112		154	LIGHT BLUE
113		155	GRAY 3
114		156	PURPLE
115		157	CURSOR LEFT
116		158	YELLOW
117		159	CYAN
118		160	SHIFTED SPACE
119		161	

ASCII	CHARACTER	ASCII	CHARACTER
162		200	
163		201	
164		202	
165		203	
166		204	
167		205	
168		206	
169		207	
170		208	
171		209	
172		210	
173		211	
174		212	
175		213	
176		214	
177		215	
178		216	
179		217	
180		218	
181		219	
182		220	
183		221	
184		222	
185		223	
186		224	SPACE
187		225	
188		226	
189		227	
190		228	
191		229	
192		230	
193		231	
194		232	
195		233	
196		234	
197		235	
198		236	
199		237	



ASCII	CHARACTER
238	☐
239	■
240	☐
241	☐
242	☐
243	☐
244	☐
245	■
246	☐
247	☐
248	■
249	☐
250	☐
251	☐
252	☐
253	☐
254	☐
255	π

0-4, 6, 7-12, 15, 16, 21-27, 128, 130-132, and 143 are not used.

# Screen Codes

---

POKE	Uppercase and Full Graphics Set	Lower- and Uppercase	POKE	Uppercase and Full Graphics Set	Lower- and Uppercase
0	@	@	31	←	←
1	A	a	32	-space-	
2	B	b	33	!	!
3	C	c	34	"	"
4	D	d	35	#	#
5	E	e	36	\$	\$
6	F	f	37	%	%
7	G	g	38	&	&
8	H	h	39	'	'
9	I	i	40	(	(
10	J	j	41	)	)
11	K	k	42	*	*
12	L	l	43	+	+
13	M	m	44	,	,
14	N	n	45	-	-
15	O	o	46	.	.
16	P	p	47	/	/
17	Q	q	48	0	0
18	R	r	49	1	1
19	S	s	50	2	2
20	T	t	51	3	3
21	U	u	52	4	4
22	V	v	53	5	5
23	W	w	54	6	6
24	X	x	55	7	7
25	Y	y	56	8	8
26	Z	z	57	9	9
27	[	[	58	:	:
28	£	£	59	;	;
29	]	]	60	<	<
30	↑	↑	61	=	=

POKE	Uppercase and Full Graphics Set	Lower- and Uppercase	POKE	Uppercase and Full Graphics Set	Lower- and Uppercase
62	>	>	99		
63	?	?	100		
64			101		
65		A	102		
66		B	103		
67		C	104		
68		D	105		
69		E	106		
70		F	107		
71		G	108		
72		H	109		
73		I	110		
74		J	111		
75		K	112		
76		L	113		
77		M	114		
78		N	115		
79		O	116		
80		P	117		
81		Q	118		
82		R	119		
83		S	120		
84		T	121		
85		U	122		
86		V	123		
87		W	124		
88		X	125		
89		Y	126		
90		Z	127		
91			128-255 reverse-video of 0-127		
92					
93					
94					
95					
96	-space-	-space-			
97					
98					

# VIC Keycodes

---

Key	Keycode	Key	Keycode
A	17	6	58
B	35	7	3
C	34	8	59
D	18	9	4
E	49	0	60
F	42	+	5
G	19	-	61
H	43	£	6
I	12	CLR/HOME	62
J	20	INST/DEL	7
K	44	←	8
L	21	@	53
M	36	*	14
N	28	↑	54
O	52	:	45
P	13	;	22
Q	48	=	46
R	10	RETURN	15
S	41	,	29
T	50	.	37
U	51	/	30
V	27	CRSR ↑↓	31
W	9	CRSR ↵	23
X	26	f1	39
Y	11	f3	47
Z	33	f5	55
1	0	f7	63
2	56	SPACE	32
3	1	RUN/STOP	24
4	57	NO KEY	
5	2	PRESSED	64

The keycode is the number found at location 197 for the current key being pressed. Try this one-line program:

```
10 PRINT PEEK (197):GOTO 10
```

### Values Stored at Location 653

Code	Key(s) pressed
0	(No key pressed)
1	SHIFT
2	Commodore
3	SHIFT and Commodore
4	CTRL
5	SHIFT and CTRL
6	Commodore and CTRL
7	SHIFT, Commodore, and CTRL

# Commodore 64 Keycodes

---

Key	Keycode	Key	Keycode
A	10	6	19
B	28	7	24
C	20	8	27
D	18	9	32
E	14	0	35
F	21	+	40
G	26	-	43
H	29		48
I	33	CLR/HOME	51
J	34	INST/DEL	0
K	37	←	57
L	42	@	46
M	36	*	49
N	39		54
O	38	:	45
P	41	;	50
Q	62	=	53
R	17	RETURN	1
S	13	,	47
T	22	.	44
U	30		55
V	31	CRSR↑↓	7
W	9	CRSR↔	2
X	23	f1	4
Y	25	f3	5
Z	12	f5	6
1	56	f7	3
2	59	SPACE	60
3	8	RUN/STOP	63
4	11	NO KEY	
5	16	PRESSED	64

The keycode is the number found at location 197 for the current key being pressed. Try this one-line program:

```
10 PRINT PEEK (197):GOTO 10
```

### Values Stored at Location 653

Code	Key(s) pressed
0	(No key pressed)
1	SHIFT
2	Commodore
3	SHIFT and Commodore
4	CTRL
5	SHIFT and CTRL
6	Commodore and CTRL
7	SHIFT, Commodore, and CTRL

□ □ □ □ □

□ □ □ □ □

# Index

---

- addressing mode 220
- algorithms
  - for calendars 140
  - minimax 8-9
- alpha-beta cutoff 9
- amplitude. See volume
- arrays 162
  - cross-reference and 229
  - files and 232
  - integer and floating-point 177
  - referencing 196
- ASC function 194
- ASCII codes 193-94, 260-63
- "Automatic Proofreader, The" program v, 243-46
- background color 117, 205-6
- BASIC computer language 129
- BASIC program, cross-reference listing of 228-31
- BRK interrupt 183
- "Build a Quiz" program v, 105-14
- calendars 140-55
  - algorithm 140
- "Canyon Runner" program v, 50-63
  - character color 117, 205-6
  - characters, high-resolution 177-79
- "Chess" program v, 3-20
- CLI ML instruction 220
- CLOSE statement, tape files and 233
- color
  - background 117, 205-6
  - text 117, 205-6
- "Color Chart" program v, 205-7
- COMPUTE!'s First Book of VIC* 169
- COMPUTE!'s Mapping the Commodore 64* 120
- concatenating files 132
- cross-reference listing of BASIC program 228-31
- crunching 177
- "Cursor GET" program 208-10
- custom characters 177-79
- data base 162
- DATA statements 239-40
  - data storage and 162-63
  - sprites and 172
- Datasette 232
- debugging machine language 219-21
- DIM statement 194
- disk
  - directory 214
  - files 213-15, 234
  - ID 171
  - "Disk Housekeeping" program 213-18
  - DOS wedge 132
  - education 67-113
  - educational software, concepts of 73
  - Eliza* program 156
  - Epson printers 122-23
  - "File Copier" program 211-12
  - filenames 125
    - tape 232-33
  - "File Reader" program 232, 235
  - files, concatenating 132
  - floating-point arrays 177
  - foreign character sets 123-24
  - "French Tutor" program v, 88-99
  - frequency (sound) 189-90
  - Gemini printers 122-23
  - GET statement v, 130, 208-9
  - graphics characters 124
  - high-resolution characters 177-79
  - "Homonym Practice for the VIC and 64" program v, 82-87
  - "Improved Tape Data File Maker" program 232, 235
  - "Indexer, The" program v, 162-65
  - indexing 163-64
  - INPUT statement 208
  - integer arrays 177
  - intelligence, computer 3
  - interrupts, manipulating 183-85
  - IRQ interrupt 183-85, 187
  - "Jackpot" program v, 21-29
  - JMP ML instruction 220
  - joystick 4, 31, 40, 51-52, 169, 171, 178, 185-86
  - JSR ML instruction 220
  - keyboard buffer 120
  - "Keyboard Data File Maker" program 232, 234-35
  - keycodes (64) 267
  - keycodes (VIC) 266
  - "Learning to Count" program 67-72
  - LISP computer language 156
  - "LIST Freezer for the VIC and 64" program 226-27
  - LIST vector 226

---

"Magic Pointer Demo" program 188  
"Magic Pointer for the VIC" program 183-88  
"Making Calendars" program v  
"Memo Writer" program v, 129-39  
operation 130-32  
minimax algorithm 8-9  
"ML Tracer" program 219-25  
"MLX" program v, 247-53  
"Monthly Calendar Printer" program 141, 144-52  
"Monthly Screen Calendar" program 141, 142-44  
"Mozart Machine, The" program v, 196-201  
"Multichar" program 177-82  
multicolor mode 178  
music, reading 100-102  
"Nirras's Labyrinth" program 30-39  
NOP ML instruction 220  
OPEN statement, tape files and 233  
polling 185  
PRINT statement 130  
PRINT# statement 214, 233  
printer codes 117, 122-24  
programmable characters 88, 90-92, 177-79  
quote mode 241-42  
raster interrupt 206  
READ statement 194  
recursion 9  
"REFMAP" program 228-31  
registers, reading 219  
repeating keys 130  
"Robot Math" program v, 73-81  
RTI ML instruction 184, 220  
RTS ML instruction 220  
scratching files 213  
screen/border color table (VIC) 259  
screen code files 131  
screen codes 264-65  
screen color codes 258  
screen color memory table (64) 257  
screen color memory table (VIC) 256

SCREEN Kernal routine 120  
screen location table (64) 255  
screen location table (VIC) 254  
scrolling 129  
"SDA: A Sprite Design Aid for the Commodore 64" program v, 171-76  
SEI ML instruction 184, 220  
shift-space, filenames and 125  
6502 microprocessor 183  
"64 Screen Formatter" program (64) 83, 87  
"64 Sound Shaper" program 189-90, 191  
sound 189-201  
differences VIC/64 189  
sound register 193  
"SpeedScript Customizer" program 117-28  
future releases and 121  
SpeedScript word processor  
default values 118-19  
sprites 171-72, 187  
square waveform 189  
stack 183-84  
stalemate 6  
status register 221  
subroutine 9  
tape files 232-34  
text color 117, 205-6  
"Therapy" program v, 156-61  
"Trident" program 40-49  
typing in programs 239-42  
unscratching files 213  
"Up or Down?" program v, 100-104  
variable storage, minimizing 177  
vectors 183-84  
list 226  
"VIC Hi-Res Sketchpad" program 169-70  
"VIC Piano" program 192-95  
"VIC Sound Shaper" program 189-91  
voices, musical 192  
volume 189  
Weizenbaum, Joseph 156





Notes

---





Notes

---



If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!** Magazine. Use this form to order your subscription to **COMPUTE!**.

For Fastest Service  
Call Our **Toll-Free** US Order Line  
**800-334-0868**  
In NC call **919-275-9809**

## COMPUTE!

P.O. Box 5406  
Greensboro, NC 27403

My computer is:

- Commodore 64    TI-99/4A    Timex/Sinclair    VIC-20    PET  
 Radio Shack Color Computer    Apple    Atari    Other \_\_\_\_\_  
 Don't yet have one...

- \$24 One Year US Subscription  
 \$45 Two Year US Subscription  
 \$65 Three Year US Subscription

Subscription rates outside the US:

- \$30 Canada  
 \$42 Europe, Australia, New Zealand/Air Delivery  
 \$52 Middle East, North Africa, Central America/Air Mail  
 \$72 Elsewhere/Air Mail  
 \$30 International Surface Mail (lengthy, unreliable delivery)

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_

Zip \_\_\_\_\_

Country \_\_\_\_\_

Payment must be in US funds drawn on a US bank, international money order, or charge card.

- Payment Enclosed    Visa  
 MasterCard    American Express

Acct. No. \_\_\_\_\_

Expires \_\_\_\_\_ / \_\_\_\_\_

Your subscription will begin with the next available issue. Please allow 4-6 weeks for delivery of first issue. Subscription prices subject to change at any time.

□ □ □ □ □

□ □ □ □ □

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!'s Gazette** for Commodore.

For Fastest Service  
Call Our **Toll-Free** US Order Line  
**800-334-0868**  
In NC call **919-275-9809**

## **COMPUTE!'s GAZETTE**

P.O. Box 5406  
Greensboro, NC 27403

My computer is:

Commodore 64     VIC-20     Other\_\_\_\_\_

- \$24 One Year US Subscription
- \$45 Two Year US Subscription
- \$65 Three Year US Subscription

Subscription rates outside the US:

- \$30 Canada
- \$45 Air Mail Delivery
- \$30 International Surface Mail

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_

Zip \_\_\_\_\_

Country \_\_\_\_\_

Payment must be in US funds drawn on a US bank, international money order, or charge card. Your subscription will begin with the next available issue. Please allow 4-6 weeks for delivery of first issue. Subscription prices subject to change at any time.

- Payment Enclosed     Visa
- MasterCard             American Express

Acct. No. \_\_\_\_\_

Expires \_\_\_\_\_ / \_\_\_\_\_

The *COMPUTE!'s Gazette* subscriber list is made available to carefully screened organizations with a product or service which may be of interest to our readers. If you prefer not to receive such mailings, please check this box .

□ □ □ □ □

□ □ □ □ □



# COMPUTE! Books

Ask your retailer for these **COMPUTE! Books** or order directly from **COMPUTE!**

Call toll free (in US) **800-334-0868** (in NC 919-275-9809) or write COMPUTE! Books, P.O. Box 5406, Greensboro, NC 27403.

Quantity	Title	Price*	Total
_____	All About the Commodore 64, Volume 1	<b>\$12.95</b>	_____
_____	COMPUTE!'s First Book of Commodore 64	<b>\$12.95</b>	_____
_____	COMPUTE!'s Second Book of Commodore 64	<b>\$12.95</b>	_____
_____	COMPUTE!'s First Book of Commodore 64 Sound & Graphics	<b>\$12.95</b>	_____
_____	COMPUTE!'s Reference Guide to Commodore 64 Graphics	<b>\$12.95</b>	_____
_____	COMPUTE!'s Beginner's Guide to Commodore 64 Sound	<b>\$12.95</b>	_____
_____	COMPUTE!'s First Book of Commodore 64 Games	<b>\$12.95</b>	_____
_____	COMPUTE!'s Second Book of Commodore 64 Games	<b>\$12.95</b>	_____
_____	Commodore 64 Games for Kids	<b>\$12.95</b>	_____
_____	COMPUTE!'s Commodore Collection, Volume 1	<b>\$12.95</b>	_____
_____	Commodore Peripherals: A User's Guide	<b>\$ 9.95</b>	_____
_____	Creating Arcade Games on the Commodore 64	<b>\$14.95</b>	_____
_____	Machine Language Routines for the Commodore 64	<b>\$14.95</b>	_____
_____	Mapping the Commodore 64	<b>\$14.95</b>	_____
_____	The VIC and 64 Tool Kit: BASIC	<b>\$16.95</b>	_____
_____	Machine Language for Beginners	<b>\$14.95</b>	_____
_____	The Second Book of Machine Language	<b>\$14.95</b>	_____

\*Add \$2.00 per book for shipping and handling. Outside US add \$5.00 air mail or \$2.00 surface mail.

**Shipping & handling: \$2.00/book** \_\_\_\_\_  
**Total payment** \_\_\_\_\_

All orders must be prepaid (check, charge, or money order).

All payments must be in US funds.

NC residents add 4.5% sales tax.

Payment enclosed.

Charge  Visa  MasterCard  American Express

Acct. No. \_\_\_\_\_ Exp. Date \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

\*Allow 4-5 weeks for delivery.

Prices and availability subject to change.

Current catalog available upon request.

□ □ □ □ □

□ □ □ □ □

□ □ □ □ □

□ □ □ □ □

**D**o you have a VIC-20 or Commodore 64—or maybe even both? Then you need *COMPUTE!'s Commodore Collection, Volume 2*. It's an exciting collection of dynamic programs—including some published here for the first time. With versions for both machines, the programs will show you dozens of ways to unleash the power of your machine.

Here are some of the things you'll find inside:

- "Chess," a sophisticated computer version of the classic board game.
- "Canyon Runner," an action-packed arcade adventure in which you maneuver a high-speed aircraft between steep canyon walls.
- "Learning to Count," a delightful introduction to counting.
- "French Tutor," a language tutorial featuring redefined characters.
- "Build a Quiz," a program that lets you create multiple-choice quizzes on any subject.
- "Memo Writer," a simple yet effective word processor for the VIC or 64.
- "Therapy," an intriguing program that psychoanalyzes as it entertains.
- "The Mozart Machine," a musical impresario that turns your computer into a composer.
- "LIST Freezer," a valuable utility to make programming easier.
- "Sound Shaper," a versatile sound editor designed to help you create just the right sound.
- And more.

Each article is clear and concise, and every program has been extensively tested. Complete program listings are included, too, making it easy to customize programs to suit your particular needs.

It makes no difference whether you're an advanced programmer or a newcomer to the world of Commodore computing. *COMPUTE!'s Commodore Collection, Volume 2* is sure to entertain and inform you.