
COMPUTE!'s COMMODORE 64/128 COLLECTION V•O•L•U•M•E 2

More than 30 selected arcade and educational games, home applications, programming utilities, and *SpeedScript* enhancements for the Commodore 64 and 128 personal computers.

All programs run on the Commodore 128 in 64 mode.

A **COMPUTE! Books** Publication

McGraw-Hill
Ryerson

**COMPUTE!'s
COMMODORE
64/128
COLLECTION**

V•O•L•U•M•E

2

COMPUTE!™ Publications, Inc. 
Part of ABC Consumer Magazines, Inc.
One of the ABC Publishing Companies
Greensboro, North Carolina

The following articles were originally published in *COMPUTE!'s Gazette*, copyright 1985, COMPUTE! Publications, Inc.: "Banners" (December); "The Construction Set" (December); "SpeedCheck: Word Processor Spelling Checker" (December)

The following articles were originally published in *COMPUTE!* magazine, copyright 1986, COMPUTE! Publications, Inc.: "RAM Expander" (November); "SpeedView" (November)

The following articles were originally published in *COMPUTE!'s Gazette*, copyright 1986, COMPUTE! Publications, Inc.: "The Fast Assembler" (January); "Sprint: A Compiler for the 64" (January); "Custom Labels" (February); "The Coordinator" (March); "Number Construction Kit" (March); "Turbo Copy" (April); "Super Synth" (May); "Word Counter" (May); "Solarpix" (June); "Budget Planner" (July); "Address Cataloger" (August); "Pirate Cove" (August); "Eagles and Gators" (September); "Video Jigsaw" (September); "Window Wizard" (September); "Directory Extension" (October); "Math Attack" (October); "Ringside Karate" (October); "Sequential File Converter for *SpeedScript*" (October); "Fill-64" (November); "Turbo Format" (November)

Copyright 1987, 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

10 9 8 7 6 5 4 3 2 1

ISBN 0-87455-089-0

The authors and publisher have made every effort in the preparation of this book to insure the accuracy of the programs and information. However, the information and programs in this book are sold without warranty, either express or implied. Neither the authors nor COMPUTE! Publications, Inc. will be liable for any damages caused or alleged to be caused directly, indirectly, incidentally, or consequentially by the programs or information in this book.

The opinions expressed in this book are solely those of the authors and are not necessarily those of COMPUTE! Publications, Inc.

COMPUTE! Publications, Inc., Post Office Box 5406, Greensboro, NC 27403, (919) 275-9809, is part of ABC Consumer Magazines, Inc., one of the ABC Publishing Companies, and is not associated with any manufacturer of personal computers. Commodore 64, Commodore 64C, and Commodore 128 are trademarks of Commodore Electronics Limited.

Contents

Foreword	v
Chapter 1. For Fun	1
Raceway	
<i>Allan Sedore</i>	3
Brick the Wall	
<i>William J. Urban</i>	14
Ringside Karate	
<i>Anthony Bertram</i>	18
Eagles and Gators	
<i>Philip I. Nelson</i>	34
The Idol of Orion	
<i>Richard Caravella</i>	54
Chapter 2. For School	63
Solarpix	
<i>Simon Edgeworth</i>	65
Video Jigsaw	
<i>Fred Karg</i>	74
Number Construction Kit	
<i>Gerald W. Rightmer</i>	81
Pirate Cove	
<i>Barbara Schulak</i>	86
Math Attack	
<i>Dan Dickerman</i>	94
Chapter 3. For Home	101
The Construction Set	
<i>Fred Karg</i>	103
Budget Planner	
<i>Gregory L. Smith</i>	126
Address Cataloger	
<i>Kevin Mykytyn</i>	137
Custom Labels	
<i>R. E. Metzler, Jr.</i>	152
Banners	
<i>Doug Smoak</i>	157

Chapter 4. For <i>SpeedScript</i>	163
SpeedCheck: An Expandable Spelling Checker	
<i>Ottis R. Cowper</i>	165
SpeedView: An 80-Column Preview Program for <i>SpeedScript</i>	
<i>Mark Schreiner</i>	186
Word Counter	
<i>Thomas K. Tucker</i>	190
Sequential File Converter for <i>SpeedScript</i>	
<i>Ronald Carnell</i>	193
Chapter 5. For Programming	197
Sprint: A Compiler for the 64	
<i>Robert A. Stoerrle</i>	199
The Fast Assembler	
<i>Yves Han</i>	223
Window Wizard	
<i>Ronald Carnell</i>	241
Fill-64	
<i>Hubert Cross</i>	260
Sequential File Reader	
<i>Gene Nash</i>	281
64 RAM Expander	
<i>Alton C. Williams</i>	284
Chapter 6. For Everything Else	287
Turbo Format	
<i>Ross Ouwinga</i>	289
Turbo Copy	
<i>Thomas Tempelmann</i>	298
Directory Extension	
<i>Kevin Mykytyn and Mark Tuttle</i>	308
Super Synth	
<i>Lawrence Cotton</i>	314
The Coordinator	
<i>Lee Noel, Jr.</i>	322
Appendices	337
A. A Beginner's Guide to Typing In Programs	339
B. How to Type In Programs	341
C. The Automatic Proofreader	
<i>Philip I. Nelson</i>	344
D. MLX: Machine Language Entry Program	
<i>Ottis R. Cowper</i>	347
Index	357
Disk Coupon	361

Foreword

COMPUTE!'s Commodore 64/128 Collection, Volume 2 contains the best published and unpublished games, educational tools, applications, programming aids, and general utilities from COMPUTE! Publications. Ready to type in and run on the Commodore 64, 64C, and 128 (in its 64 mode), the more than 40 BASIC and machine language programs here offer something for every Commodore owner.

Games, both entertaining and educational, open up new worlds in your computer. Experience the fright of entering a dangerous dungeon in "The Idol of Orion," or the challenge of head-to-head motorcycle racing in "Raceway." Watch the graceful motion of the solar system in "Solarpix," or protect your city from marauding math problems in "Math Attack."

Software for the home can help you organize, or advertise, or just while away the hours. "Budget Planner" and "Address Cataloger" let you keep track of finances and friends. "Banners" puts more power in your printer—use it to print signs and banners for yard sales or reunions.

An entire chapter offers a spelling checker, a word counter, a document preview option, and more enhancements to *SpeedScript*, COMPUTE! Publications' ever-popular word processor.

Programmers, too, will find some impressive applications and aids in *COMPUTE!'s Commodore 64/128 Collection, Volume 2*. A fast machine language assembler and a very powerful BASIC compiler are but two applications. Or add more memory—without paying a dime—to your 64.

General utilities and applications round out the book, with such valuable programs as "Turbo Format" and "Directory Extension" to help manage your disks.

Every article is clearly written and easy to understand. All the programs have been thoroughly tested and are ready to type in. We've even included our high-quality error-checking utilities, "The Automatic Proofreader" and "MLX."

If you prefer, you can buy a disk with all the programs in this book from COMPUTE! Publications by using the coupon in the back, or by calling toll-free 1-800-346-6767 (in N.Y. 1-212-887-8525).

Chapter 1

For Fun

Raceway

Allan Sedore

Fight through crowds of motorcycles as you race yours down long highways. Pelting rain, slippery ice, and dark nights won't stop you, but a crash can. This all machine language game includes smooth scrolling, great graphics, and dynamite sound.

"Raceway" is a fast-paced game that tests your skill as a highway motorcycle racer. Written entirely in machine language, Raceway gives you all the fast action you can take. The smooth vertical scrolling takes you down roads and through torments of Mother Nature. More dangerous, though, are your opponents, who stop at nothing and run you down without a second thought.

Limited Time

In addition to all the other obstacles you face, you must also race against the clock. You start with 100 seconds—when the time's gone, so are you. But be quick and fearless and you'll be rewarded. Every time your score increases by 100 points, the clock returns to 100 seconds. The more road you cover, the more your score goes up. Maintain a fast speed, and you'll have plenty of time.

Go!

Since Raceway is written in machine language, it must be typed in with "MLX," the machine language entry program found in Appendix D. Run MLX, and when prompted for the starting and ending address, enter these numbers:

Starting address 0801

Ending address 162D

Once you've entered all the data for Raceway, save a copy to disk or tape. Load and run it as you would a BASIC program, with the statement

LOAD"filename",8

for disk, or

LOAD"filename"

for tape.

Type **RUN** and the demo screen appears. Plug a joystick into port 2. When you're ready to start, press the joystick button.

Your motorcycle is colored blue and is in the middle of the screen. Push the joystick forward to increase speed, pull it back to decrease your speed, and shove it left or right to move in those directions.

The clock is near the upper left, your score at the upper right, and the number of your remaining motorcyles is shown by the small figures at the bottom right.

As you race down the road, road conditions change from dry pavement to rain-slick to slippery ice. Lateral movements when you're driving on wet roads or ice can be dangerous, so be careful.

Raceway is over when you run out of time. To start another game, just press the joystick button when you see the demo screen.

Raceway

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```
0801:0B 08 00 00 9E 32 30 36 EC
0809:31 00 00 00 4C 40 0F AD E0
0811:F7 30 85 02 A2 1E BD D8 DB
0819:30 E8 9D D8 30 CA CA E0 DF
0821:FF D0 F3 A5 02 8D D8 30 66
0829:4C D0 0F 85 02 A2 06 BD 32
0831:F8 30 E8 9D F8 30 CA CA A9
0839:E0 FF D0 F3 4C D0 0F F8 CF
0841:30 60 23 24 23 24 23 24 3C
0849:23 24 A9 F0 2D 18 D0 09 AC
0851:0C 8D 18 D0 A9 06 8D 20 7B
0859:D0 A9 0C 8D 21 D0 20 80 A3
0861:14 A9 C0 85 FB 85 FC A9 EF
0869:FF 8D 17 D0 8D 1C D0 A9 F4
0871:00 8D 1B D0 8D 1D D0 EA C2
0879:EA EA A9 A8 8D 00 D0 A9 31
0881:CA 8D 01 D0 A9 FF 8D 15 05
0889:D0 A9 00 85 FE A9 BE 8D 6E
0891:F8 07 A2 01 8E 25 D0 CA B9
```


0899:8E 26 D0 A2 0E 8E 27 D0 88
08A1:A9 15 8D FF 03 A9 0F 8D E7
08A9:18 D4 A9 20 8D 3E 03 8D 2B
08B1:FA 03 A9 68 8D 02 D0 A9 7B
08B9:88 8D 04 D0 A9 C8 8D 06 90
08C1:D0 A9 BE 8D F9 07 8D FA 57
08C9:07 8D FB 07 A9 00 8D 18 31
08D1:0F A9 26 8D 19 0F A9 02 CB
08D9:8D 1A 0F A9 3F 8D 00 0F F2
08E1:A9 36 8D 01 0F A9 30 8D 23
08E9:02 0F A9 00 8D 26 0F 8D A4
08F1:27 0F 8D 28 0F A9 20 8D 7A
08F9:11 0F 8D 12 0F 8D 13 0F 0D
0901:A9 00 8D 0C 0F A9 01 8D 09
0909:0D 0F 8D 0E 0F A9 BE 8D 22
0911:FC 07 8D FD 07 A9 A8 8D B2
0919:08 D0 A9 E8 8D 0A D0 A9 07
0921:4C 8D 03 0F A9 3A 8D 04 63
0929:0F A9 67 8D 0A 0F A9 E2 B5
0931:8D 0B 0F A9 00 8D 0F 0F AC
0939:8D 10 0F A9 20 8D 14 0F 01
0941:8D 15 0F A9 34 8D 09 D0 96
0949:A9 54 8D 05 D0 8D 07 D0 E2
0951:A9 74 8D 03 D0 8D 0B D0 DA
0959:EA EA A9 01 8D 40 03 8D E1
0961:41 03 A9 00 8D 42 03 8D 13
0969:43 03 8D 47 03 A9 15 8D 7A
0971:44 03 A9 02 8D 08 D4 A9 9B
0979:81 8D 0C D4 A9 CC 8D 0D 27
0981:D4 AD 1E D0 A9 03 8D 50 FE
0989:03 A9 00 A2 07 9D 00 31 91
0991:CA E0 FF D0 F8 A9 80 8D 4B
0999:0B D4 AD 1E D0 20 46 0D 9E
09A1:A9 00 8D 29 0F 8D 2A 0F DE
09A9:A9 11 8D 13 D4 A9 B9 8D 06
09B1:14 D4 A2 02 8A 9D 26 D0 5F
09B9:E8 E0 07 D0 F7 AD 55 03 8A
09C1:C9 01 D0 06 20 50 0F 4C 20
09C9:CE 09 20 80 13 A9 00 85 56
09D1:C6 A9 20 A9 00 8D 12 D4 7F
09D9:4C 00 0A 00 00 00 00 00 53
09E1:00 00 00 00 00 00 00 00 F3
09E9:00 00 00 00 00 00 00 00 FB
09F1:00 00 00 00 00 00 00 00 04
09F9:00 00 00 00 00 00 00 C6 D2
0A01:FC A5 FC C9 05 D0 0A 20 D8
0A09:10 08 20 00 0C 20 B7 0F 8A
0A11:EA CE 3B 03 AD 3B 03 C9 10
0A19:00 D0 08 A9 06 8D 3B 03 DC
0A21:20 00 0B 20 00 0D AD FA 33

```

0A29:03 C9 01 D0 03 4C 67 11 87
0A31:EE 18 0F AD 18 0F C9 05 15
0A39:D0 05 A9 00 8D 18 0F EE 06
0A41:19 0F AD 19 0F C9 40 D0 DD
0A49:05 A9 26 8D 19 0F EE 1A E4
0A51:0F AD 1A 0F C9 FD D0 05 79
0A59:A9 02 8D 1A 0F 20 00 0E 1D
0A61:20 30 0F 20 00 10 20 66 5C
0A69:0F 20 20 0C 20 8F 0F AD DC
0A71:FD 03 C9 00 D0 05 A9 E0 4D
0A79:8D FD 03 4C 00 0A FC 60 7B
0A81:D8 03 20 10 08 CE D8 03 F7
0A89:10 F8 60 3B 03 C9 01 D0 B5
0A91:03 20 10 08 4C 00 0A 08 30
0A99:4D C9 2F 8D C9 2F A9 20 04
0AA1:4D C8 2F 8D C8 2F A9 20 C3
0AA9:4D CB 2F 8D CB 2F A9 80 05
0AB1:4D E7 2F 8D E7 2F A9 80 F4
0AB9:4D EA 2F 8D EA 2F A9 02 57
0AC1:4D E9 2F 8D E9 2F A9 02 17
0AC9:4D EC 2F 8D EC 2F 60 2F 92
0AD1:A9 02 4D EC 2F 8D EC 2F 6C
0AD9:60 EA 2F 8D EA 2F A9 02 01
0AE1:4D E9 2F 8D E9 2F A9 02 37
0AE9:4D EC 2F 8D EC 2F 60 31 B4
0AF1:35 36 3A 99 A3 31 30 29 7A
0AF9:22 1E D4 D2 C1 C3 CB 4C 6F
0B01:B7 0C 85 FD A9 0F 25 FD 58
0B09:49 0F 85 FD 29 01 C9 01 F9
0B11:D0 0D A5 FB C9 22 F0 07 07
0B19:C6 FB C6 FB 20 BC 0B A5 D9
0B21:FD 29 02 C9 02 D0 0D A5 70
0B29:FB C9 C0 F0 07 E6 FB E6 89
0B31:FB 20 D3 0B A5 FD 29 04 F3
0B39:C9 04 D0 16 CE 00 D0 AD 76
0B41:00 D0 C9 61 D0 0C EE 00 6F
0B49:D0 EE 00 D0 EE 00 D0 20 C9
0B51:72 0B A5 FD 29 08 C9 08 FC
0B59:D0 16 EE 00 D0 AD 00 D0 49
0B61:C9 EF D0 0C CE 00 D0 CE 1A
0B69:00 D0 CE 00 D0 20 72 0B 84
0B71:60 A9 01 4D FE 03 8D FE 35
0B79:03 8D 04 D4 AD 01 D0 C9 1F
0B81:DB F0 03 4C 28 0B 60 CE E3
0B89:01 D0 A2 02 CE 00 0F CE 2C
0B91:01 0F CE 02 0F CE 03 0F AE
0B99:CE 04 0F CA E0 00 D0 EC 3C
0BA1:60 EE 01 D0 A2 02 EE 00 CB
0BA9:0F EE 01 0F EE 02 0F EE A0
0BB1:03 0F EE 04 0F CA E0 00 90

```

0BB9:D0 EC 60 CE 40 03 AD 40 16
0BC1:03 C9 00 F0 05 E6 FB E6 7D
0BC9:FB 60 A9 03 8D 40 03 4C 1B
0BD1:88 0B CE 41 03 AD 41 03 31
0BD9:C9 00 F0 05 C6 FB C6 FB F2
0BE1:60 A9 03 8D 41 03 4C A2 1D
0BE9:0B 52 45 56 4F 1D 45 4D EE
0BF1:41 47 1C 00 00 00 00 00 FD
0BF9:00 00 00 00 AD 00 DC CE 06
0C01:FF 03 AD FF 03 C9 00 D0 9F
0C09:45 A9 10 8D FF 03 4C 4F FC
0C11:0C 8D 42 03 AD 55 03 C9 9D
0C19:01 D0 03 4C 0D 08 60 20 74
0C21:E4 FF C9 20 F0 01 60 20 53
0C29:E4 FF C9 20 D0 F9 60 8D AB
0C31:21 D0 8D 28 D0 8D 29 D0 22
0C39:8D 2A D0 8D 2B D0 8D 2C 79
0C41:D0 60 00 00 00 00 00 00 D9
0C49:00 00 00 00 00 00 A9 01 B5
0C51:4D FE 03 8D FE 03 8D 04 2C
0C59:D4 CE FD 03 AD FD 03 C9 B4
0C61:00 D0 2D A2 04 BD 38 05 0A
0C69:C9 39 F0 06 FE 38 05 4C 62
0C71:7C 0C A9 30 9D 38 05 CA A5
0C79:4C 66 0C E0 02 D0 11 A0 F6
0C81:09 AD 1A 05 C9 39 F0 0B 3D
0C89:EE 1A 05 88 C0 00 D0 F1 62
0C91:4C 99 0C A9 39 8D 1B 05 8D
0C99:AD FF 03 C9 01 D0 16 A9 A6
0CA1:28 4D 84 2F 8D 84 2F 8D 0F
0CA9:B7 2F A9 28 4D 87 2F 8D 95
0CB1:87 2F 8D BA 2F 60 AD 55 62
0CB9:03 C9 01 D0 01 60 AD 00 D7
0CC1:DC 4C 03 0B 11 11 11 11 6C
0CC9:11 11 11 11 11 11 11 11 E1
0CD1:9E 22 3B 00 E8 14 6F AD 4D
0CD9:F7 30 85 02 A2 1E BD D8 AC
0CE1:30 E8 9D D8 30 CA CA E0 B0
0CE9:FF D0 F3 83 22 D4 48 49 2B
0CF1:53 20 45 44 49 54 4F 52 35
0CF9:20 48 41 53 20 42 45 CE F4
0D01:FC 03 AD FC 03 C9 00 D0 EF
0D09:22 A9 03 8D FC 03 A2 04 15
0D11:BD 1A 05 C9 30 F0 06 DE FD
0D19:1A 05 4C 2C 0D A9 39 9D EC
0D21:1A 05 CA E0 02 D0 01 CA 11
0D29:4C 11 0D A2 04 BD 1A 05 C9
0D31:C9 30 D0 10 CA E0 02 D0 06
0D39:01 CA EA E0 FF D0 EE A9 BC

```

0D41:01 8D FA 03 60 A9 FE 2D A3
0D49:0E DC 8D 0E DC A9 FB 25 DE
0D51:01 85 01 A2 FF BD 00 D0 5F
0D59:9D 00 30 BD 00 D1 9D 00 A6
0D61:31 BD 00 D2 9D 00 32 BD BF
0D69:00 D3 9D 00 33 BD 00 D4 91
0D71:9D 00 34 BD 00 D5 9D 00 4F
0D79:35 BD 00 D6 9D 00 36 BD 22
0D81:00 D7 9D 00 37 CA E0 FF EB
0D89:D0 CB A9 04 05 01 85 01 AC
0D91:A9 01 0D 0E DC 8D 0E DC 59
0D99:A2 00 BD 3C 12 9D D8 30 69
0DA1:BD 5C 12 9D 18 31 E8 E0 06
0DA9:20 D0 EF A2 00 BD 7C 12 32
0DB1:9D F0 32 E8 E0 08 D0 F5 6A
0DB9:A2 00 BD C0 12 9D 40 2F 9F
0DC1:E8 E0 C0 D0 F5 60 AD 21 5B
0DC9:D0 29 0F C9 00 D0 0B A2 11
0DD1:02 8A 9D 26 D0 E8 E0 07 98
0DD9:D0 F7 A9 0C 8D 21 D0 60 43
0DE1:8D 20 D0 A2 18 9D 00 D4 1B
0DE9:CA E0 FF D0 F7 60 59 20 C2
0DF1:41 4E 44 22 00 30 16 75 4D
0DF9:00 83 22 4C A0 0E 60 A2 9E
0E01:04 A0 08 8E F8 03 B9 03 7B
0E09:D0 C9 00 F0 EE BD 00 0F 8C
0E11:C9 80 90 08 A2 02 8E F7 F6
0E19:03 4C 25 0E A2 FE 8E F7 75
0E21:03 4D F7 03 8D F6 03 AE 3E
0E29:F8 03 DE 21 0F BD 21 0F 31
0E31:C9 00 D0 59 AD F6 03 ED 1F
0E39:20 0F 9D 21 0F AD F7 03 11
0E41:79 03 D0 99 03 D0 BD 07 6C
0E49:0F D9 03 D0 F0 07 BD 26 16
0E51:0F C9 00 F0 38 8C 17 0F A7
0E59:AE 17 0F AC F8 03 B9 0C 92
0E61:0F C9 00 D0 06 FE 02 D0 85
0E69:4C 6F 0E DE 02 D0 AE F8 E0
0E71:03 AC 17 0F DE 11 0F A9 11
0E79:01 9D 26 0F A9 01 9D 07 C6
0E81:0F BD 11 0F C9 01 10 05 1F
0E89:A9 00 9D 26 0F CA 88 88 CD
0E91:E0 FF F0 03 4C 04 0E 60 5B
0E99:00 00 00 00 00 00 00 A9 5F
0EA1:20 9D 11 0F AD 19 0F EA 23
0EA9:EA EA AD 1A 0F 9D 07 0F 59
0EB1:AE 18 0F BD 1B 0F AE F8 54
0EB9:03 99 02 D0 AD 1A 0F 29 28
0EC1:01 9D 0C 0F AD 18 0F C9 ED
0EC9:00 D0 08 A9 00 9D 0C 0F 53

```

0ED1:4C DD 0E C9 04 D0 05 A9 01
0ED9:01 9D 0C 0F A9 00 9D 26 FE
0EE1:0F 4C 0E 0E 88 A8 C8 E8 9C
0EE9:7D 6D 03 06 00 00 00 20 01
0EF1:20 20 78 01 01 78 0B 30 75
0EF9:7F 0A 06 2B EE CD C4 50 54
0F01:36 30 65 45 0E 16 58 43 04
0F09:40 81 40 00 01 00 01 00 B9
0F11:20 20 20 20 20 00 06 01 5B
0F19:32 A4 68 88 A8 C8 E8 7F C8
0F21:21 3A 1E 17 3A 00 00 00 65
0F29:00 00 43 4F 4E 54 41 AD 98
0F31:1E D0 29 01 C9 01 D0 06 C1
0F39:AD 1E D0 4C 00 11 60 A9 43
0F41:01 8D 55 03 A9 60 8D 30 38
0F49:0F A2 00 9A 4C 20 16 18 68
0F51:A2 08 A0 0F 20 F0 FF A2 2F
0F59:09 BD EA 0B 20 D2 FF CA 90
0F61:E0 FF D0 F5 60 AD 55 03 D0
0F69:C9 01 F0 01 60 A9 06 8D 1E
0F71:3B 03 A9 01 20 0B 0B A9 20
0F79:10 2D 00 DC C9 00 F0 01 E9
0F81:60 A9 00 8D 55 03 A9 AD CA
0F89:8D 30 0F 4C 4B 08 A2 04 E4
0F91:A0 08 B9 03 D0 C9 01 F0 0A
0F99:04 C9 00 D0 11 FE 28 D0 DE
0FA1:BD 28 D0 29 0F C9 0B D0 DB
0FA9:05 A9 02 9D 28 D0 CA 88 71
0FB1:88 E0 FF D0 DD 60 A9 00 1D
0FB9:AA 69 06 E8 18 C5 FB 90 37
0FC1:F8 86 FC 60 3A 54 32 B2 DD
0FC9:56 00 D9 17 7E 00 8B AD 78
0FD1:37 31 85 02 A2 1E BD 18 C9
0FD9:31 E8 9D 18 31 CA CA E0 2B
0FE1:FF D0 F3 A5 02 8D 18 31 B4
0FE9:EE 18 0F AD 18 0F C9 05 D7
0FF1:D0 05 A9 00 8D 18 0F 60 3A
0FF9:32 B2 56 3A 9F 31 35 AD 26
1001:42 03 C9 00 F0 31 C9 01 1D
1009:F0 2A C9 02 F0 29 C9 03 48
1011:F0 1F C9 04 F0 21 C9 05 8F
1019:F0 1A C9 06 F0 19 C9 07 58
1021:F0 0F C9 08 F0 08 C9 09 7B
1029:F0 0D EA EA EA EA 4C 8D 3A
1031:10 4C 85 10 4C 6F 10 4C AA
1039:41 10 4C 00 10 4C 00 10 49
1041:CE 43 03 AD 43 03 C9 00 8E
1049:D0 23 CE 44 03 AD 44 03 13
1051:C9 00 D0 19 A9 0C 20 C7 87
1059:0D EE 42 03 AD 42 03 C9 7A

1061:0B D0 05 A9 00 20 12 0C 27
1069:A9 15 4C 33 14 60 A9 03 D8
1071:8D 21 D0 AD 18 0F C9 01 27
1079:D0 07 20 18 14 EA 20 00 D5
1081:0B 4C 41 10 A9 00 20 30 21
1089:0C 4C 41 10 CE 46 03 A9 2B
1091:00 20 28 14 AD 46 03 C9 56
1099:00 D0 A5 A9 0C 8D 46 03 63
10A1:AE 47 03 E8 E0 08 D0 02 A4
10A9:A2 00 8E 47 03 0E BE 10 3F
10B1:AD BE 10 C9 00 D0 05 A9 ED
10B9:01 8D BE 10 A9 01 9D 00 23
10C1:31 CA E0 FF D0 02 A2 07 24
10C9:CA E0 FF D0 02 A2 07 CA 08
10D1:E0 FF D0 02 A2 07 A9 00 21
10D9:9D 00 31 20 00 0B 4C 41 F6
10E1:10 8D 44 03 A9 00 AA 9D 66
10E9:00 31 E8 E0 08 D0 F8 60 57
10F1:4C 41 10 85 00 56 B2 4D EE
10F9:42 3A 4C B2 32 3A 8D A9 BD
1101:81 20 42 14 A9 09 8D 27 29
1109:D0 A9 BD 8D F8 07 20 4D FF
1111:11 AD 00 D0 E9 0C 8D 00 CE
1119:D0 A9 01 8D 1D D0 20 4D C0
1121:11 A9 0A 6D 01 D0 8D 01 B5
1129:D0 A9 FE 2D 17 D0 8D 17 FE
1131:D0 20 4D 11 A9 0C 6D 00 D6
1139:D0 8D 00 D0 A9 00 8D 1D B9
1141:D0 20 4D 11 A9 80 8D 0B 04
1149:D4 4C 50 14 EE 48 03 AD 80
1151:48 03 C9 00 D0 F6 EE 49 1B
1159:03 AD 49 03 C9 30 D0 EC 5F
1161:A9 00 8D 49 03 60 A9 00 8B
1169:8D 03 D0 8D 05 D0 8D 07 93
1171:D0 8D 09 D0 8D 0B D0 CE 96
1179:50 03 AE 50 03 E0 FF D0 CB
1181:03 4C 15 12 BD 91 07 9D DB
1189:B9 07 A9 0C 8D 21 D0 A9 7C
1191:C0 85 FB 85 FC A9 FF 8D 69
1199:17 D0 A9 A8 8D 00 D0 A9 F2
11A1:CA 8D 01 D0 A9 00 85 FE 11
11A9:8D 26 0F 8D 27 0F 8D 28 8F
11B1:0F 8D 29 0F 8D 2A 0F 8D 95
11B9:43 03 8D 47 03 A9 BE 8D 2E
11C1:F8 07 A9 0E 8D 27 D0 A9 8C
11C9:15 8D FF 03 A9 20 8D 3E 31
11D1:03 8D FA 03 8D 11 0F 8D C4
11D9:12 0F 8D 13 0F 8D 14 0F 91
11E1:8D 15 0F A9 50 8D 00 0F 54
11E9:A9 36 8D 01 0F A9 30 8D 3D

11F1:02 0F A9 65 8D 03 0F A9 A4
11F9:45 8D 04 0F A9 01 8D 40 40
1201:03 8D 41 03 EA EA AD 1E DE
1209:D0 AD 1E D0 A9 06 9D B9 2C
1211:DB 4C 00 0A A9 00 8D 15 54
1219:D0 18 A2 08 A0 0F 20 F0 F2
1221:FF A2 09 BD EA 0B 20 D2 81
1229:FF CA E0 FF D0 F5 A2 10 CF
1231:20 4D 11 CA E0 FF D0 F8 29
1239:4C 0D 08 18 18 18 18 18 B2
1241:18 18 18 18 18 18 18 18 65
1249:18 00 00 00 00 00 00 00 79
1251:00 00 00 00 00 00 00 00 75
1259:00 00 18 BD BD BD B9 9C 51
1261:53 46 92 0E 0E 6B BF 14 A5
1269:58 59 8D 8D 8D 16 90 A5 26
1271:19 8D 64 F3 71 C2 E6 CA 80
1279:CA CA BD 18 18 3C 7E 3C D9
1281:3C 18 18 40 50 2D 20 20 68
1289:22 57 04 62 44 62 15 4C 5D
1291:44 2D 00 64 60 3A 41 C4 9C
1299:0A 00 00 A6 02 2C 40 32 A0
12A1:3B 8D 31 3A D9 53 24 50 45
12A9:24 20 20 00 20 20 91 22 B2
12B1:3A 1F 35 6C 78 00 30 15 61
12B9:3A 93 25 21 20 32 E6 00 2E
12C1:0C 00 0C 0B 0C 00 C0 20 20
12C9:30 33 0C 0C E8 38 43 6F 33
12D1:E8 82 7A 10 0E BB 60 06 81
12D9:DA CC 0E 56 B0 0E 67 E4 37
12E1:3A D5 AC 3B AA 63 0F AE 91
12E9:60 03 A7 B8 0E 5A AC 0E C0
12F1:69 EC 01 3F 30 83 8C 00 C2
12F9:0C 03 0C 00 C0 06 AD 00 DF
1301:3C 00 00 3C 00 00 14 00 31
1309:03 FF C0 3D 55 7C D5 7D 62
1311:57 7F EB FD CE EB B3 0E BB
1319:FF B0 03 D7 C0 03 55 C0 C6
1321:00 D7 00 03 BE C0 03 AA 17
1329:C0 00 EB 00 00 FF 00 00 2D
1331:FF 00 00 7D 00 00 3C 00 A7
1339:00 14 00 00 3C 00 AC 00 9F
1341:3C 00 0C EB 30 0C EB 30 7F
1349:04 EB 10 0C EB 30 00 EB 3B
1351:00 03 AA C0 03 AA C0 03 E0
1359:AA C0 03 BE C0 0E BE B0 BD
1361:0E FF B0 CE F7 B3 FE BE DC
1369:BF 7E AA BD FE AA BF CF 32
1371:AA F3 03 FF C0 00 FF 00 50
1379:00 C3 00 00 00 00 00 A9 3A

```

1381:30 8D B2 13 A9 98 20 D2 6D
1389:FF A0 13 A2 07 20 A8 13 81
1391:E8 20 A8 13 E8 20 A8 13 A6
1399:A9 1E 20 D2 FF A9 60 8D 42
13A1:B2 13 E8 4C DD 13 60 18 DB
13A9:20 F0 FF A9 71 20 D2 FF 68
13B1:A9 60 8D 0F D4 A9 21 8D 84
13B9:12 D4 A9 20 8D 12 D4 A9 5D
13C1:00 8D 5B 03 8D 5C 03 EE B9
13C9:5B 03 AD 5B 03 C9 00 D0 D9
13D1:F6 EE 5C 03 AD 5C 03 C9 99
13D9:B0 D0 EC 60 20 A8 13 A0 9A
13E1:13 A2 07 A9 05 20 D2 FF 04
13E9:20 F0 FF A9 20 20 D2 FF 1E
13F1:E8 20 F0 FF A9 20 20 D2 93
13F9:FF E8 20 F0 FF A9 20 20 74
1401:D2 FF E8 20 F0 FF A9 20 AC
1409:20 D2 FF 60 EA EA EA EA BF
1411:EA EA EA EA EA EA EA A9 F7
1419:04 8D 0F D4 A9 81 8D 12 56
1421:D4 A9 04 8D 18 D4 60 20 6C
1429:30 0C A9 FF 8D 0F D4 4C 40
1431:1D 14 A9 00 8D 12 D4 A9 2A
1439:0F 8D 18 D4 A9 15 4C E2 B9
1441:10 8D 0B D4 A9 00 8D 12 FD
1449:D4 A9 0F 8D 18 D4 60 A9 7F
1451:04 8D 6C 03 A2 FF 88 C0 83
1459:00 D0 FB CA E0 00 D0 F6 81
1461:CE 6C 03 AD 6C 03 C9 00 4A
1469:D0 EA 4C 67 11 8D 76 14 74
1471:8C 77 14 CA BD 0A 16 20 4F
1479:D2 FF E0 00 D0 F5 60 A9 EF
1481:06 20 4D 15 18 A2 00 A0 9B
1489:00 20 F0 FF A9 56 A0 15 D4
1491:A2 A0 20 6E 14 A0 05 A2 ED
1499:A0 20 74 14 88 D0 F8 A9 0D
14A1:CF 8D 76 14 A2 27 20 74 8B
14A9:14 EA 18 A2 06 A0 02 20 9A
14B1:F0 FF A9 F6 A0 15 A2 14 A9
14B9:20 6E 14 18 A2 06 A0 20 20
14C1:20 F0 FF A9 0A A0 16 A2 72
14C9:14 20 6E 14 A9 08 8D DE 7A
14D1:14 A9 D8 8D DF 14 A0 19 0C
14D9:A2 15 A9 01 9D F0 DB CA 11
14E1:D0 FA 18 A9 28 6D DE 14 97
14E9:8D DE 14 90 03 EE DF 14 C3
14F1:88 D0 E5 A9 27 8D 0B 15 84
14F9:8D 10 15 A9 04 8D 0C 15 AD
1501:A9 D8 8D 11 15 A0 19 A9 FF
1509:A0 8D 0F 08 A9 06 8D 0F D8

```


1511:DC 18 A9 28 6D 0B 15 8D B6
1519:0B 15 18 A9 28 6D 10 15 D7
1521:8D 10 15 90 06 EE 0C 15 DA
1529:EE 11 15 88 D0 D9 A9 01 7C
1531:8D 86 02 18 A2 17 A0 21 59
1539:20 F0 FF A9 FF 20 D2 FF 70
1541:20 D2 FF 20 D2 FF A9 1B 37
1549:8D 11 D0 60 8D 86 02 A9 D2
1551:0B 8D 11 D0 60 20 26 25 88
1559:26 25 26 25 26 25 26 DD E6
1561:20 20 20 5E 20 20 20 5E AD
1569:20 20 20 5E 20 20 20 5E B5
1571:20 20 20 DD 26 25 26 25 CC
1579:26 25 26 25 26 20 24 23 34
1581:24 23 24 23 24 23 24 DD 11
1589:20 20 20 5D 20 20 20 5D C4
1591:20 20 20 5D 20 20 20 5D CC
1599:20 20 20 DD 24 23 24 23 D6
15A1:24 23 24 23 24 20 25 26 6F
15A9:25 26 25 26 25 26 25 DD E0
15B1:20 20 20 5C 20 20 20 5C DB
15B9:20 20 20 5C 20 20 20 5C E3
15C1:20 20 20 DD 25 26 25 26 18
15C9:25 26 25 26 25 20 23 24 2B
15D1:23 24 23 24 23 24 23 DD 0B
15D9:20 20 20 5B 20 20 20 5B F2
15E1:20 20 20 5B 20 20 20 5B FA
15E9:20 20 20 DD 23 24 23 24 22
15F1:23 24 23 24 23 39 39 2E FB
15F9:39 39 05 9D 9D 9D 9D 9D C5
1601:11 92 3A 45 4D 49 54 12 40
1609:9E 30 30 30 30 30 05 9D 83
1611:9D 9D 9D 9D 11 92 45 52 B0
1619:4F 43 53 12 9E 4B 49 A9 A7
1621:0D 8D 18 03 A9 08 8D 19 0C
1629:03 4C 4B 08 1D 00 00 00 BC

Brick the Wall

William J. Urban

As fast as you slap new bricks and mortar into the wall, old, decrepit bricks fall out. It's a mason's nightmare. But hidden within the wall are bricks of pure gold—and bricks of deadly blue. Try your hand at bricklaying in this quick-reaction arcade game. Requires a joystick.

You're the bricklayer. Your job is to patch the holes in the wall with new bricks. Sound easy? Maybe.

Type in "Brick the Wall" and save a copy to disk or tape. To make sure you have an error-free copy of the program, use "The Automatic Proofreader," a utility found in Appendix C.

Plug a joystick into port 2, load the program, and type **RUN**.

Brick It Up

You'll see the display with a prompt asking you to choose a starting level. Press a key from 1 (easiest) to 8 (hardest) to begin the game.

Use your joystick to control the white cursor on the wall. You can go up, down, left, and right, but not diagonally. The cursor keeps moving in the designated direction until you change. It also wraps around on all sides. In other words, when you move the cursor off the top edge of the wall, the cursor reappears at the bottom.

You pick up points in Brick the Wall in this fashion:

- Patch a fallen brick = five points multiplied by the level number.
- Pass over a gold brick = ten times the points received for patching a brick.
- Touch a blue brick = collapse the wall and end the game (no points)

The holes appear randomly.

When you patch the last hole of a level, you receive 500 bonus points and continue to the next level. Level 8 is the highest level, but you can still earn bonus points.

Brick the Wall

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

```

AP 5 H=RND(-TI/37)
BC 10 POKE53280,0:POKE53281,0:PRINT"{CLR}";
GC 20 GOSUB10000:DIMN%(280):B$="{RVS}{RED}O[Y]P":C
    =54272:G=0:O=2:B=0
SS 25 FORL=CTOC+24:POKEL,0:NEXT:POKEC+24,15:POKEC+
    6,240
HR 30 PRINT"{CLR}";:FORL=1TO280:PRINTB$;:NEXT
EG 40 S$="{HOME}{22 DOWN}{WHT}{OFF}"
GH 50 M=1024:POKEM+C,1:GOSUB700
SC 60 W$="{YEL}{7 SPACES}YOU KNOCKED THE WALL DOWN
    1"
GK 70 PRINT"{HOME}{24 DOWN}{GRN}{RVS}{7 RIGHT}BEWA
    RE OF THE BLUE BRICK!{HOME}";
GX 80 PRINTS$;"{3 SPACES}{WHT}SCORE"SC;TAB(30)"
    {YEL}LEVEL"A/5
JH 100 GOSUB200:REM ***MAIN LOOP***
XK 110 IFINT(RND(1)*100)<ATHENGOSUB1000
XP 190 GOTO100:REM ***END MAIN LOOP***
FH 200 J=15-(PEEK(56320)AND15)
QX 210 IF0<>2THENGOSUB2000
XE 220 IFJ=0ORJ=5ORJ=6ORJ=9ORJ=10THENJ=G
DP 225 POKEM+C,0
PG 230 IFJ=1THENM=M-40:IFM<1024THENM=M+840
QS 240 IFJ=2THENM=M+40:IFM>1863THENM=M-840
MS 250 IFJ=4THENM=M-1:IFM<1024THENM=1063
GK 260 IFJ=8THENM=M+1:IFM>1863THENM=1824
KE 265 G=J
BK 270 O=(PEEK(M+C)AND15):POKEM+C,1:RETURN
EQ 300 POKEC+4,129:POKEC+1,120
SG 310 FORD=1TO5:NEXTD:POKEC+4,32
PX 390 RETURN
EP 400 POKEC+4,33:POKEC+1,80
FD 410 FORD=1TO20:NEXTD:POKEC+1,120
DG 420 FORD=1TO20:NEXTD:POKEC+4,32
FF 490 RETURN
FR 500 POKEC+4,33:FORL=1TO2
ES 510 POKEC+1,50:FORD=1TO50:NEXTD
JK 520 POKEC+1,180:FORD=1TO50:NEXTD
MS 530 NEXTL:POKEC+4,32
XA 540 SC=SC+(10*A):PRINTS$;"{3 SPACES}{WHT}SCORE"
    SC
AP 590 RETURN
XH 600 POKEC+4,33:FORL=150TO50STEP-1
SK 610 POKEC+1,L:NEXTL:POKEC+4,32
HJ 620 GOTO3000
DM 700 T=0:FORL=1TOINT(A/5)
FE 710 H=INT(RND(1)*837)+1024

```

```

JQ 720 IFPEEK(H)<>207OR(PEEK(H+C)AND15)<>2THEN710
HC 730 GOSUB1030:NEXT G=0:J=0:RETURN
PQ 800 IFB<1THENRETURN
AD 810 FORX=1TOB:H=N%(X)
MQ 820 POKEH+C,0:POKEH+C+1,0:POKEH+C+2,0
XC 830 GOSUB300:NEXTX:B=0:RETURN
BX 1000 H=INT(RND(1)*837)+1024:T=0
SX 1010 IFPEEK(H)<>207OR(PEEK(H+C)AND15)<>2THENRET
URN
GB 1020 IFINT(RND(1)*10)<2THENT=7:GOTO1030
XF 1025 IFINT(RND(1)*10)>7THENB=B+1:N%(B)=H:T=14
CR 1030 POKEH+C,T:POKEH+C+1,T:POKEH+C+2,T
HB 1040 IFT=7ORT=14THENRETURN
EK 1050 Q=Q+1:GOSUB300:IFQ>50THENW$="{YEL}
{4 SPACES}YOUR WALL IS TOO FULL OF HOLES"
ME 1055 IFQ>50THEN3000
SM 1060 PRINT$TAB(20)"[7]HOLES"Q"{LEFT} "
QE 1090 RETURN
QK 2000 IFO=14THEN600
FK 2010 IFO=7THENGOSUB500:GOTO2030
CG 2020 SC=SC+A:GOSUB400:PRINT$;"{3 SPACES}{WHT}S
CORE"SC
CF 2025 Q=Q-1
JK 2030 IFPEEK(M)=207THENPOKEM+C,2:POKEM+C+1,2:POK
EM+C+2,2
RR 2040 IFPEEK(M)=247THENPOKEM+C-1,2:POKEM+C,2:POK
EM+C+1,2
RB 2050 IFPEEK(M)=208THENPOKEM+C-2,2:POKEM+C-1,2:P
OKEM+C,2
GP 2080 PRINT$;TAB(20)"[7]HOLES"Q"{LEFT} "
MA 2085 IFQ<1THENA=A+5:GOSUB4000:GOSUB700
JD 2090 GOTO270
RR 3000 GOSUB800:FORD=1TO300:NEXTD:POKEC+4,129:PRI
NT"{HOME}";
XE 3010 FORL=1TO21:POKEC+1,200:PRINT"{40 SPACES}";
KM 3015 POKEC+1,10:NEXTL
BE 3020 PRINT"{3 DOWN}{35 SPACES}";
SR 3030 POKEC+4,128:PRINT$TAB(20)"{8 SPACES}"
DM 3040 PRINT"{HOME}{WHT}{9 SPACES}PRESS ANY KEY T
O PLAY"
HR 3045 PRINT"{10 DOWN}"W$
QS 3047 GETA$:IFA$THEN3047:REM ERROR TRAP
HK 3050 GETA$:IFA$=""THEN3050
AP 3060 RUN
EF 4000 POKEC+6,252:N1=50:N2=50:FORZ=1TO3
PF 4010 POKEC+1,N1:GOSUB4040:N1=N1+5
SS 4020 POKEC+1,N2:GOSUB4040:N2=N2-5
JG 4030 NEXTZ:FORD=1TO200:NEXTD:GOTO4070
CC 4040 POKEC+4,17:FORD=1TO100:NEXTD
DC 4050 POKEC+4,16:FORD=1TO100:NEXTD

```

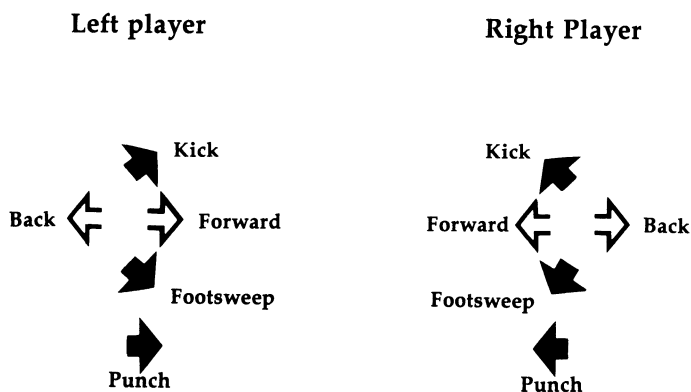
```
DX 4060 RETURN
AX 4070 POKEC+6,240:IFA>40THENA=40
PP 4080 SC=SC+500:PRINTS$;"{3 SPACES}{WHT}SCORE"SC
;TAB(30)"{YEL}LEVEL"A/5
DD 4090 RETURN
FA 10000 B1$="{RVS}O[8 Y]P{DOWN}{10 LEFT}{[H]
{8 SPACES}[N]{DOWN}{10 LEFT}L[8 P]@"
ER 10010 B2$="{RVS}O[3 Y]P{DOWN}{5 LEFT}{[H]
{3 SPACES}[N]{DOWN}{5 LEFT}L[3 P]@" :C$="
{2 UP}":PRINT"{CLR}";
AP 10020 FORR=1TO3:READN:POKE646,N:FORL=1TO4
SX 10025 DATA 7,6,2,6,7,0
AK 10030 PRINTB1$;C$;B1$;C$;B1$;C$;B1$;
GM 10040 PRINTB2$;C$;B1$;C$;B1$;C$;B1$;C$;B2$;:NEX
TL:PRINT"{HOME}";:NEXTL
MR 10045 PRINT"{HOME}{3 DOWN}"
CB 10050 FORL=1TO16:PRINT"{6 RIGHT}{28 SPACES}
{6 RIGHT}";:NEXTL
ER 10053 PRINT"{HOME}{3 DOWN}"
FC 10060 PRINT"{HOME}{4 DOWN}":PRINTTAB(14)"{WHT}B
RICK WALL"
CS 10070 PRINTTAB(7)"{DOWN}USE JOYSTICK IN PORT TW
O"
SQ 10080 PRINTTAB(7)"AND PATCH THE HOLES IN THE"
RD 10090 PRINTTAB(7)"{UP}BRICK WALL."
XG 10100 PRINTTAB(7)"{DOWN}GOLD BRICKS ARE WORTH B
IG"
RK 10110 PRINTTAB(7)"POINTS."
CB 10120 PRINTTAB(7)"{DOWN}CHOOSE DIFFICULTY:KEY 1
-8"
DH 10130 GETA$:IFA$=""THEN10130
RP 10140 IFA$>"0"ANDA$<"9"THEN10150
MG 10145 GOTO10130
KR 10150 A=VAL(A$):A=A*5:PRINT"{HOME}";
DB 10160 FORR=1TO3:READN:POKE646,N:FORL=1TO4
BH 10170 PRINTB1$;C$;B1$;C$;B1$;C$;B1$;
EA 10180 PRINTB2$;C$;B1$;C$;B1$;C$;B1$;C$;B2$;:NEX
TL:PRINT"{HOME}";:NEXTL
MX 10190 RETURN
```

Ringside Karate

Anthony Bertram

Go head to head with another player for the coveted black belt in this outstanding commercial-quality, arcade-style game. Two joysticks are required.

As in real karate, skill, speed, and timing are the necessary ingredients to play "Ringside Karate." Each player controls one fighter with a joystick. Moving the joystick and/or pressing the button in various combinations delivers different kicks, punches, and defensive blocks.



Solid arrows indicate that the joystick button should be pressed.

It may take some practice to come up with the right offensive move. Keep at it.

To block, just press the fire button while the joystick is pushed in any other direction than the ones outlined in the illustration. To win, you must score ten knockdowns, each worth 100 points.

Timing is crucial—you must release the fire button for a moment before you make another attack. If you rapidly press the button, the fighter won't move and probably will be knocked down. Blocking provides a limited defense and is best

used just before a counterattack or when backing up. You'll find the controls much easier to master if you've played an arcade karate game.

Typing It In

Ringside Karate is written entirely in machine language. You need "MLX," the machine language entry program found in Appendix D, to type it in. After loading and running MLX, enter the following addresses at the prompts:

Starting Address: 3000

Ending Address: 45BF

The game is long, but much of the sprite and screen DATA are repeated numbers, which makes the typing a little easier.

To load Ringside Karate, type

LOAD "filename",8,1

(where *filename* is the name you used to save the program) if you're using disk. Type

LOAD "filename",1,1

if you're using tape.

To start the game, type

SYS 15490

The joystick plugged into port 1 controls the fighter dressed in white, while the joystick in port 2 controls the fighter in black. When the game is over, a winner is declared, and another game may be started by pressing the button on the joystick plugged into port 1 (which controls the white fighter).

Ringside Karate

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```
3000:00 0F C0 00 15 50 00 3F 65
3008:F0 00 3F A0 00 3F B0 00 31
3010:3B A8 00 3A A0 00 2A A0 D5
3018:00 2A 00 00 3A 00 00 5E 33
3020:70 01 57 5C 01 75 57 01 37
3028:5D 55 03 57 5A 00 D5 AA 8B
```

```

3030:00 36 A8 00 1D 70 00 15 F2
3038:70 00 15 70 00 3F F0 0F 68
3040:00 00 00 00 00 00 00 00 A0
3048:00 00 00 00 00 00 00 00 A8
3050:00 00 00 00 00 00 00 00 B0
3058:00 00 00 00 00 00 02 00 BC
3060:00 02 80 00 0A C0 00 6B 10
3068:00 00 6C 00 00 70 00 00 18
3070:C0 00 00 00 00 00 00 00 31
3078:00 00 00 00 00 00 00 0B E3
3080:00 15 70 00 55 5C 00 55 A5
3088:5C 01 55 57 01 55 57 01 84
3090:55 57 05 5F 55 05 70 15 BC
3098:05 70 15 15 C0 05 15 C0 90
30A0:05 15 C0 05 17 00 01 17 03
30A8:00 01 17 00 01 17 0F 01 AF
30B0:17 3F C1 2B FF FE 3F C0 92
30B8:3F 00 00 00 00 00 00 0F C7
30C0:00 00 00 00 00 00 00 00 21
30C8:00 00 00 00 00 00 00 00 29
30D0:00 00 C0 00 00 C0 00 00 4C
30D8:C0 00 00 70 00 00 70 00 81
30E0:00 70 00 00 70 00 00 70 51
30E8:00 00 70 00 00 70 00 00 19
30F0:70 00 00 AC 00 00 FC 00 4E
30F8:00 00 00 00 00 00 00 0F 68
3100:00 15 70 00 55 5C 00 55 27
3108:5F 01 55 57 01 55 57 01 87
3110:55 57 01 5F 55 01 70 15 AD
3118:01 70 15 01 70 05 01 70 D3
3120:05 01 70 05 01 5C 01 01 20
3128:5C 01 00 57 01 00 5B C1 EE
3130:00 6F F1 00 BF FE 03 F0 9D
3138:0F 00 00 00 00 00 00 0F 31
3140:00 00 00 00 00 00 00 00 A2
3148:FC 00 01 55 00 03 FF 00 AA
3150:03 FA 00 03 FB 00 03 BA C3
3158:00 03 AA 00 02 AA 00 02 8D
3160:A0 00 03 A0 00 05 E7 00 61
3168:15 75 00 1D 55 00 1D 55 BE
3170:00 17 55 00 15 A9 00 35 C7
3178:A9 00 0F 57 00 01 57 00 B9
3180:00 00 00 00 00 00 00 00 E2
3188:00 00 00 00 00 00 00 00 EA
3190:00 00 00 00 00 00 80 00 F4
3198:00 00 00 00 00 00 00 00 FA
31A0:00 00 00 00 FF C2 80 55 64
31A8:6A 80 55 6A 80 5F C0 00 B4
31B0:70 00 00 C0 00 00 00 00 57
31B8:00 00 00 00 00 00 00 00 1B

```


31C0:00 01 57 00 03 FF 00 01 67
31C8:57 00 01 57 00 01 57 00 1F
31D0:01 55 00 01 55 00 01 55 1B
31D8:00 01 5D 00 01 5C 00 01 A1
31E0:5C 00 05 5C 00 15 70 00 0D
31E8:55 C0 01 57 00 05 5C 00 88
31F0:15 70 0F 0A 70 FF 02 AF B3
31F8:FF 0F F0 00 00 00 00 00 3D
3200:00 00 00 00 00 00 00 00 64
3208:00 00 00 00 C0 00 00 70 E2
3210:00 00 5C 00 00 57 00 00 5D
3218:55 C0 00 15 C0 00 35 C0 D9
3220:00 35 C0 00 05 C0 00 05 1A
3228:C0 00 05 C0 00 05 C0 00 2F
3230:C5 C0 00 C5 C0 00 FA B0 B0
3238:00 FF F0 00 00 00 00 00 BA
3240:00 3F 00 00 55 40 00 FF 20
3248:C0 00 FE 80 00 FE C0 00 72
3250:EE A0 00 EA 80 00 AA 80 DC
3258:00 A8 00 00 A8 00 00 EA 17
3260:D5 00 7A D5 01 5F 55 01 8D
3268:55 50 05 55 50 05 55 50 13
3270:15 D5 54 17 C5 57 15 A5 2C
3278:5D 05 A1 75 00 00 D5 00 04
3280:00 00 00 00 00 00 00 00 E4
3288:00 00 00 00 00 00 00 00 EC
3290:A0 00 00 A0 00 02 B0 00 B8
3298:6A C0 00 6B 00 00 6C 00 F1
32A0:00 70 00 00 C0 00 18 00 57
32A8:01 58 00 15 58 00 55 5C BE
32B0:05 55 5C 15 55 F0 55 5C 3F
32B8:00 55 70 00 55 C0 00 00 2E
32C0:00 03 55 00 00 55 00 00 E5
32C8:55 00 00 55 00 01 55 00 DB
32D0:01 57 00 01 5C 00 01 50 D0
32D8:00 05 70 00 05 70 00 05 7B
32E0:70 00 15 C0 00 15 C0 00 02
32E8:15 C0 00 15 C0 00 17 00 8D
32F0:00 57 00 00 57 00 00 AB 91
32F8:0F 00 FF FF 00 00 00 00 E4
3300:57 00 00 7C 00 00 70 00 BA
3308:00 C0 00 00 00 00 00 00 9E
3310:00 00 00 00 00 00 00 00 76
3318:00 00 00 00 00 00 00 00 7E
3320:00 00 00 00 00 00 00 00 86
3328:00 00 00 00 00 00 00 00 8E
3330:00 00 00 0F FC 00 FF FF 6F
3338:FF FC 0F FF 00 00 00 00 BF
3340:00 FC 00 01 55 00 03 FF A6
3348:00 03 FA 00 03 FB 00 03 D9

```

3350:BA 80 03 AA 00 02 AA 00 9C
3358:02 A0 00 03 A0 00 01 E0 FF
3360:00 05 75 55 05 55 05 39
3368:D5 FF 05 75 C0 01 5F C0 3B
3370:03 55 6A 00 D5 6A 00 55 A8
3378:C0 00 55 C0 00 FF C0 00 77
3380:00 00 00 00 00 00 00 00 E6
3388:00 00 00 00 B0 00 00 AC 21
3390:00 00 B0 00 00 B0 00 00 CF
3398:B0 00 00 B0 00 00 B0 00 C3
33A0:00 B0 00 00 B0 00 00 00 B8
33A8:00 00 00 00 00 F0 00 00 D2
33B0:A0 00 00 80 00 00 20 00 AF
33B8:00 00 00 00 00 00 00 0B 2A
33C0:00 15 7F 00 55 55 00 55 B1
33C8:55 00 55 55 01 55 55 01 E2
33D0:55 FF 01 5F 00 01 5C 00 B4
33D8:01 70 00 01 70 00 01 70 E1
33E0:00 01 70 00 01 70 00 01 60
33E8:70 00 01 70 00 01 70 00 93
33F0:01 70 0F 02 B3 FF 03 FF 99
33F8:F0 00 00 00 00 00 00 0F E6
3400:00 00 00 00 00 00 50 00 09
3408:00 50 00 00 50 00 00 50 57
3410:00 00 50 00 00 50 00 00 C3
3418:50 00 00 A0 00 00 20 00 F2
3420:00 20 00 00 00 00 00 00 90
3428:00 00 00 00 00 00 00 00 90
3430:00 00 00 C0 00 00 F0 00 86
3438:00 00 00 00 00 00 00 00 A0
3440:00 00 00 00 00 00 00 00 A8
3448:00 00 00 00 00 00 00 00 B0
3450:00 00 00 00 00 00 00 00 B8
3458:00 00 00 00 00 00 00 00 C0
3460:00 00 00 00 00 00 00 00 C8
3468:03 F0 00 05 54 00 0F FC 9C
3470:00 0F E8 00 0F EC 00 0E F3
3478:EA 00 0E A8 00 0A A8 02 1E
3480:00 00 00 00 00 00 00 00 E8
3488:00 00 00 00 00 00 00 00 F0
3490:00 00 00 00 00 00 00 00 F8
3498:00 00 00 00 00 00 00 00 01
34A0:00 00 00 00 00 00 00 00 09
34A8:00 00 00 00 00 00 00 00 11
34B0:00 00 00 00 00 00 00 00 19
34B8:00 00 00 00 00 00 00 02 23
34C0:00 0A 80 00 0E 80 00 07 35
34C8:97 00 05 D5 00 15 55 00 F9
34D0:55 55 00 55 5C 01 5D 5C 8C
34D8:03 C5 5C 0D 75 5C 0D 5F 27

```

```

34E0:FF 35 55 55 35 D5 55 35 77
34E8:F5 55 35 C5 55 B5 C0 00 A7
34F0:B5 C0 FF BA CF FF FF FF 8E
34F8:FF 00 00 00 00 00 00 02 63
3500:00 00 00 00 00 00 00 00 6A
3508:00 C0 00 00 70 00 00 5F 85
3510:00 00 55 A8 00 15 A8 00 55
3518:0F 00 00 35 C0 00 D5 70 7F
3520:00 55 50 00 55 5C 00 50 56
3528:57 00 40 15 C0 00 15 A0 68
3530:FF 05 A8 FF 01 FC FF FF EC
3538:FC 00 00 00 00 00 00 02 23
3540:00 00 00 00 00 00 00 00 AA
3548:00 00 00 00 00 00 00 00 B2
3550:00 00 00 00 00 00 00 00 BA
3558:00 00 00 00 00 00 00 00 C2
3560:00 00 00 00 00 00 00 00 CA
3568:00 00 00 00 00 00 00 00 D2
3570:00 00 00 00 00 00 00 00 DA
3578:00 00 00 00 00 00 00 00 E2
3580:00 00 00 00 00 00 00 00 EA
3588:00 00 00 00 00 00 00 00 F2
3590:00 00 00 00 00 00 00 00 FA
3598:00 00 00 00 00 00 00 00 03
35A0:00 00 00 00 00 00 00 00 0B
35A8:00 00 00 00 00 00 00 00 13
35B0:00 00 00 00 00 00 00 00 1B
35B8:00 00 00 00 00 00 00 00 23
35C0:00 00 00 00 00 00 00 00 2B
35C8:00 00 00 00 00 00 00 00 33
35D0:00 00 00 00 00 00 00 00 3B
35D8:00 00 00 00 80 00 1E A0 24
35E0:00 DE A1 55 DE AB 55 DF BC
35E8:EA D5 DF AB 5F DF ED 55 A0
35F0:FF FF 55 00 03 D5 00 00 75
35F8:FF 00 00 00 00 00 00 00 63
3600:00 00 00 00 00 00 00 00 6C
3608:00 00 00 00 00 00 00 00 74
3610:00 00 00 00 00 00 07 00 8A
3618:00 15 C0 00 55 70 01 55 A5
3620:70 D5 5D 70 D5 71 70 D7 1A
3628:C1 70 D5 7F 70 75 55 F2 3B
3630:5D 55 5E 57 F5 5A D6 AD 56
3638:5A F6 AF FF 3F F0 00 00 43
3640:00 00 00 00 00 00 00 00 AC
3648:00 00 00 00 00 00 00 00 B4
3650:00 00 00 00 00 00 00 00 BC
3658:00 00 00 00 00 00 00 00 C4
3660:B0 00 02 B0 00 00 AC 00 C9
3668:00 2B 00 00 0B 00 00 03 FA

```

```

3670:00 00 03 00 00 00 00 00 3D
3678:00 00 00 00 00 00 00 0B EF
3680:03 F0 00 05 54 00 0F FC B8
3688:00 0A FC 00 0E FC 00 2A A5
3690:EC 00 0A AC 00 0A A8 00 F8
3698:00 A8 00 00 A4 00 07 9F 02
36A0:C0 1F 7F C0 7F F7 F0 FF EE
36A8:DF F0 AF 7F F0 AA FD C0 1E
36B0:2A 77 C0 03 DF C0 03 FF 60
36B8:00 03 FF 00 01 57 00 0F 5A
36C0:00 00 00 00 00 00 00 00 2D
36C8:00 00 00 00 00 00 00 00 35
36D0:00 00 00 00 00 00 00 00 3D
36D8:00 00 00 00 00 02 00 00 4D
36E0:02 00 00 02 00 00 02 00 72
36E8:00 02 00 00 02 00 00 02 E7
36F0:00 00 02 00 00 05 00 00 B1
36F8:0F 00 00 00 00 00 00 0F FB
3700:02 AB 00 0A AA C0 0A AA 12
3708:C0 2A AA B0 2A AA B0 2A 49
3710:AA B0 AA FA AC AB 02 AC C9
3718:AB 02 AC AC 00 AB AC 00 45
3720:AB AC 00 AB B0 00 2B B0 D6
3728:00 2B B0 00 2B B0 F0 2B A0
3730:B3 FC 2B 7F FF 17 FC 03 6E
3738:FF 00 00 00 00 00 00 0F B5
3740:02 AB 00 0A AA C0 0A AA 52
3748:C0 2A AA B0 2A AA B0 2A 89
3750:AA B0 AA FA B0 AB 02 B0 2E
3758:AB 02 B0 AC 02 B0 AC 02 2C
3760:B0 AC 02 B0 B0 0A B0 B0 5D
3768:0A B0 B0 2A C0 BF 1A C0 BA
3770:BF C6 C0 7F FD C0 F0 0F 64
3778:C0 00 00 00 00 00 00 0F 56
3780:00 00 00 00 00 00 00 00 EE
3788:00 00 00 00 00 00 00 00 F6
3790:00 00 00 00 00 00 00 02 01
3798:00 00 00 00 00 00 00 00 07
37A0:00 00 00 00 02 81 55 02 D1
37A8:AB FF 02 AB FF 00 01 5F 49
37B0:00 00 07 00 00 01 00 00 04
37B8:00 00 00 00 00 00 00 00 27
37C0:00 00 00 00 00 00 3F 00 AD
37C8:00 55 40 00 FF C0 00 AF 47
37D0:C0 00 EF C0 00 AE C0 00 E5
37D8:AA C0 00 AA 80 00 0A 80 10
37E0:00 0A 40 00 79 F0 00 F7 61
37E8:FC 00 FF DF 00 FF DF 00 93
37F0:FF 7F 00 EA FF 00 EA F4 B8
37F8:00 3F 50 00 3F F0 00 00 FE

```

3800:00 00 00 00 00 00 00 00 70
3808:00 00 00 00 00 00 00 00 78
3810:00 02 00 00 0A 00 00 2A 7B
3818:00 00 AA 00 00 AB 00 00 8C
3820:AC 00 00 AC 00 00 AC 00 0B
3828:00 AC 00 00 AC 00 00 AC D5
3830:00 00 AC 00 00 AC 00 01 E9
3838:5F 00 03 FF 00 00 00 00 B8
3840:2A B0 00 15 70 00 2A B0 CB
3848:00 2A B0 00 2A B0 00 AA 18
3850:B0 00 AA B0 00 AA B0 00 85
3858:BA B0 00 CA B0 00 0A B0 49
3860:00 0A AC 00 02 AB 00 00 A7
3868:AA C0 00 2A B0 00 0A AC 47
3870:F0 02 A8 FF 02 5C FF F5 66
3878:7C 00 0F FC 00 00 00 00 D8
3880:00 00 00 00 00 00 00 00 F0
3888:00 00 00 00 00 00 00 00 F8
3890:0A 00 00 0A C0 00 02 B0 61
3898:00 00 AD 00 00 2B 00 00 6B
38A0:0B 00 00 01 1B C0 00 1A 9C
38A8:BC 00 1A AB 00 0A AA F0 E3
38B0:0A AA AC 00 AA AB 00 0A 74
38B8:AA 00 02 AA 00 00 AA 00 BE
38C0:03 F0 00 05 54 00 0F FC FC
38C8:00 0A FC 00 0E FC 00 2A E9
38D0:EC 00 0A AC 00 0A A8 00 3D
38D8:00 A8 00 00 A8 00 55 A9 0D
38E0:00 FD A7 C0 FF 5F C0 5F 30
38E8:FF F0 0F FF F0 0F FF FC 38
38F0:3F FF FC 7F FF 7C 9F FA C4
38F8:FC A7 FA F4 A9 FC 50 00 62
3900:00 00 2A 00 00 0A 00 00 DF
3908:02 00 00 00 00 00 00 00 7B
3910:00 00 00 00 00 00 00 00 82
3918:00 00 00 00 00 00 00 00 8A
3920:00 00 00 00 00 00 00 00 92
3928:00 00 00 00 00 00 00 00 9A
3930:00 00 00 00 0F FC FF FF 0F
3938:FF FF C0 3F 00 00 00 00 B6
3940:AA C0 00 AA C0 00 AA C0 FE
3948:00 AA C0 00 AA B0 00 2A BF
3950:B0 00 0A B0 00 0A B0 00 F0
3958:02 AC 00 02 AC 00 02 AC 2D
3960:00 00 AB 00 00 AB 00 00 F6
3968:AB 00 00 AB 00 00 2B 00 C1
3970:00 2A 00 00 2A 00 FF D5 94
3978:00 FF FF 00 00 00 00 00 EA
3980:00 00 00 00 00 00 00 00 F2
3988:00 00 00 00 00 00 0B 00 11

```

3990:00 2B 00 00 0B 00 00 0B 31
3998:00 00 0B 00 00 0B 00 00 98
39A0:0B 00 00 0B 00 00 0B 00 5F
39A8:00 00 00 00 00 00 00 0F 2A
39B0:00 00 0A 00 00 02 00 00 6C
39B8:08 00 00 00 00 00 00 0B 3A
39C0:00 3F 00 00 55 40 00 FF AE
39C8:C0 00 AF C0 00 EF C0 02 E0
39D0:AE C0 00 AA C0 00 AA 80 51
39D8:00 0A 80 00 0A 40 55 59 33
39E0:F0 FF F7 F0 FF FF F0 55 11
39E8:FD F0 00 F7 F0 35 5F F0 22
39F0:AF FF 70 AF FF F0 05 55 67
39F8:C0 03 FF C0 01 55 C0 00 77
3A00:00 00 00 00 00 00 00 00 74
3A08:0A 00 00 0A 00 00 0A 00 36
3A10:00 0A 00 00 0A 00 00 0A 61
3A18:00 00 0A 00 00 05 00 00 E1
3A20:04 00 00 04 00 00 00 00 D6
3A28:00 00 00 00 00 00 00 00 9C
3A30:00 00 00 00 00 03 00 00 B0
3A38:0F 00 00 00 00 00 00 00 34
3A40:FE AB 00 AA AA C0 AA AA 22
3A48:C0 AA AA C0 AA AA B0 C0 4B
3A50:AA B0 C0 0A B0 C0 0A B0 4C
3A58:C0 02 B0 C0 02 B0 00 02 A4
3A60:B0 00 02 B0 00 02 B0 00 E1
3A68:02 B0 00 02 B0 00 02 B0 64
3A70:3F C2 B0 FF FD 70 C0 FF 7E
3A78:F0 00 00 00 00 00 00 0F 74
3A80:00 00 00 00 00 00 00 00 F4
3A88:00 00 00 00 00 00 00 00 FC
3A90:00 00 00 00 00 00 00 00 05
3A98:00 00 00 00 00 00 00 00 0D
3AA0:00 00 00 00 00 00 00 00 15
3AA8:00 00 00 00 00 00 00 00 1D
3AB0:00 00 00 00 00 00 00 00 25
3AB8:00 00 00 00 00 00 00 0B 38
3AC0:00 00 00 00 00 00 00 00 35
3AC8:00 00 00 00 00 00 00 00 3D
3AD0:00 00 00 00 00 00 00 00 45
3AD8:00 00 00 00 00 00 00 00 4D
3AE0:00 00 00 00 00 00 00 0F 64
3AE8:C0 00 15 50 00 3F F0 00 44
3AF0:2B F0 00 3B F0 00 AB B0 7A
3AF8:00 2A B0 00 2A A0 00 02 E3
3B00:00 00 00 00 00 00 00 00 76
3B08:00 00 00 01 00 00 07 00 9C
3B10:28 0F 00 2A AD 00 2A A4 67
3B18:00 00 F0 00 00 AC 00 02 61

```

```

3B20:AB 00 0A AA 00 0A AA 00 D5
3B28:2A FA 00 AB 02 05 AB 00 A8
3B30:15 AC 0F 3F B0 FF 3F FF 36
3B38:FF 00 00 00 00 00 00 02 B0
3B40:02 A0 00 02 90 00 5E 70 B1
3B48:00 FD F4 00 FF FD 00 5F 34
3B50:FF 40 0F FF C0 0F F7 D0 BB
3B58:0F F1 D0 0F FE B0 F5 7A FE
3B60:B4 AA AA AC AA AB AC AA 04
3B68:AF AC AA A3 AC 00 03 AE 8B
3B70:FF F3 AE FF FF 5E FF FF 33
3B78:FF 00 00 00 00 00 00 02 F0
3B80:00 00 00 00 00 00 00 00 F6
3B88:00 00 00 00 00 00 00 00 FE
3B90:00 00 00 00 00 00 00 00 07
3B98:00 00 00 00 00 00 00 00 0F
3BA0:00 00 00 00 00 00 00 00 17
3BA8:00 00 00 00 00 00 00 00 1F
3BB0:00 00 00 00 00 00 00 00 27
3BB8:00 00 00 00 00 00 00 0B 3A
3BC0:00 00 00 00 00 00 00 00 37
3BC8:00 00 00 00 00 00 00 00 3F
3BD0:00 00 00 00 00 00 00 00 47
3BD8:00 00 00 00 00 00 00 00 4F
3BE0:00 00 00 00 00 00 00 00 57
3BE8:00 00 00 00 00 00 00 00 5F
3BF0:00 00 00 00 00 00 00 00 67
3BF8:00 00 00 00 00 00 00 00 6F
3C00:00 00 00 00 00 00 00 00 78
3C08:00 00 00 00 00 00 00 00 80
3C10:00 00 00 00 00 00 2C 00 E0
3C18:00 AB 00 02 AA C0 02 AA A2
3C20:BF 02 BA A9 02 8E A9 02 8A
3C28:83 E9 02 FE A9 4F AA A5 92
3C30:7A AA 9A 5A A1 EA 5A BA 22
3C38:A5 FF FA AF 00 0F FC 00 14
3C40:00 00 00 00 00 00 00 00 B8
3C48:00 00 00 00 00 00 00 00 C0
3C50:00 00 00 00 00 00 00 00 C8
3C58:00 00 00 00 02 00 00 0A EA
3C60:B4 FF CA B7 FF 6A B7 FD 1F
3C68:AB F7 5F 6A F7 FF DB F7 B6
3C70:FF FF FF FF C0 00 FF 00 EE
3C78:00 00 00 00 00 00 00 00 F0
3C80:00 82 20 A3 40 20 96 40 C7
3C88:A9 00 8D 3C 03 8D 3D 03 17
3C90:8D 3E 03 8D 3F 03 A9 FF F1
3C98:8D 15 D0 8D 1C D0 A9 01 88
3CA0:8D 25 D0 A9 00 8D 26 D0 31
3CA8:A2 00 A9 0A 9D 27 D0 E8 5C

```

3CB0:E0 08 D0 F6 A9 C0 8D F8 89
3CB8:07 A9 C1 8D F9 07 A9 C2 32
3CC0:8D FA 07 A9 C3 8D FB 07 8D
3CC8:A9 D9 8D FC 07 A9 DA 8D 30
3CD0:FD 07 A9 DB 8D FE 07 A9 1D
3CD8:DC 8D FF 07 A9 B4 8D 05 D3
3CE0:D0 8D 07 D0 8D 0D D0 8D E2
3CE8:0F D0 A9 A0 8D 01 D0 8D FB
3CF0:03 D0 8D 09 D0 8D 0B D0 05
3CF8:A9 51 8D 00 D0 8D 04 D0 E1
3D00:A9 69 8D 02 D0 8D 06 D0 15
3D08:A9 B4 8D 08 D0 8D 0C D0 5C
3D10:A9 CC 8D 0A D0 8D 0E D0 8E
3D18:AD 00 DC 29 10 D0 35 A5 6B
3D20:FC D0 31 AD 00 DC 29 0F 23
3D28:C9 0A D0 08 A9 E2 20 D1 8F
3D30:3E 4C BF 3D C9 09 D0 08 C4
3D38:A9 EA 20 D1 3E 4C BF 3D 43
3D40:C9 0B D0 08 A9 DE 20 D1 D7
3D48:3E 4C BF 3D A9 E6 20 D1 BB
3D50:3E 4C BF 3D AD 00 DC 29 19
3D58:10 F0 04 A9 00 85 FC AD EF
3D60:00 DC 29 0F C9 0F D0 03 57
3D68:4C BF 3D 38 C9 08 90 30 E3
3D70:AD 0E D0 18 C9 28 B0 03 34
3D78:4C BF 3D AD 08 D0 38 ED 6D
3D80:00 D0 C9 14 B0 03 4C BF 93
3D88:3D A2 05 CE 08 D0 CE 0A 03
3D90:D0 CE 0C D0 CE 0E D0 CA D0
3D98:D0 F1 EE FF 07 4C BF 3D FB
3DA0:AD 0E D0 38 C9 EB 90 03 35
3DA8:4C BF 3D A2 05 EE 08 D0 CF
3DB0:EE 0A D0 EE 0C D0 EE 0E BD
3DB8:D0 CA D0 F1 EE FF 07 AD BA
3DC0:01 DC 29 10 D0 35 A5 FB BB
3DC8:D0 31 AD 01 DC 29 0F C9 31
3DD0:06 D0 08 A9 C9 20 B1 3E 8E
3DD8:4C 6C 3E C9 07 D0 08 A9 2E
3DE0:C5 20 B1 3E 4C 6C 3E C9 BA
3DE8:05 D0 08 A9 D1 20 B1 3E 66
3DF0:4C 6C 3E A9 CD 20 B1 3E 9F
3DF8:4C 6C 3E AD 01 DC 29 10 35
3E00:F0 04 A9 00 85 FB AD 01 A3
3E08:DC 29 0F C9 0F D0 03 4C C9
3E10:6C 3E 38 C9 08 90 25 AD 70
3E18:00 D0 18 C9 2D B0 03 4C E6
3E20:6C 3E A2 05 CE 00 D0 CE ED
3E28:02 D0 CE 04 D0 CE 06 D0 92
3E30:CA D0 F1 EE FA 07 EE FA 40
3E38:07 4C 6C 3E AD 08 D0 38 24

3E40:ED 00 D0 C9 14 B0 03 4C 20
3E48:6C 3E AD 00 D0 38 C9 DC 18
3E50:90 03 4C 6C 3E A2 05 EE 9B
3E58:00 D0 EE 02 D0 EE 04 D0 22
3E60:EE 06 D0 CA D0 F1 EE FA C3
3E68:07 EE FA 07 AD 1E D0 A2 1E
3E70:64 20 BD 3F AD 1E D0 AA 05
3E78:29 12 C9 12 F0 73 8A 29 FB
3E80:48 C9 48 F0 6C 8A 29 18 A3
3E88:C9 18 F0 65 8A 29 42 C9 AB
3E90:42 F0 5E A5 FB A8 A9 C0 27
3E98:20 B1 3E 98 85 FB A5 FC 47
3EA0:A8 A9 D9 20 D1 3E 98 85 57
3EA8:FC A2 32 20 BD 3F 4C 18 30
3EB0:3D A2 00 9D F8 07 18 69 CB
3EB8:01 E8 E0 04 D0 F5 A9 01 FE
3EC0:85 FB AD F8 07 C9 C0 F0 16
3EC8:07 C9 D5 F0 03 20 6D 40 B8
3ED0:60 A2 00 9D FC 07 18 69 9D
3ED8:01 E8 E0 04 D0 F5 A9 01 1F
3EE0:85 FC AD FC 07 C9 EE F0 13
3EE8:E7 C9 D9 F0 E3 20 6D 40 D0
3EF0:60 AD F8 07 C9 CD F0 70 70
3EF8:AD FC 07 C9 E6 F0 69 AD 84
3F00:F8 07 C9 C0 D0 51 AD FC 26
3F08:07 C9 D9 F0 4A 20 47 40 68
3F10:A2 0A CE 00 D0 CE 02 D0 D2
3F18:CE 04 D0 CE 06 D0 CA D0 DF
3F20:F1 EE FA 07 EE FA 07 A9 3E
3F28:D5 20 B1 3E A9 D9 20 D1 7A
3F30:3E 20 13 40 A2 FF 20 BD 4F
3F38:3F A2 FF 20 BD 3F A2 FF 31
3F40:20 BD 3F A9 03 CD 3F 03 91
3F48:F0 03 4C B4 3C A9 E8 CD FC
3F50:3E 03 D0 F6 4C E8 3F AD 6A
3F58:FC 07 C9 D9 D0 0D AD F8 FC
3F60:07 C9 C0 D0 06 4C 93 3E C0
3F68:4C 93 3E AD FC 07 C9 D9 06
3F70:D0 F6 AD F8 07 C9 C0 F0 2C
3F78:EF 20 47 40 A2 0A EE 08 07
3F80:D0 EE 0A D0 EE 0C D0 EE A9
3F88:0E D0 CA D0 F1 A9 EE 20 DC
3F90:D1 3E A9 C0 20 B1 3E 20 2D
3F98:FF 3F A2 FF 20 BD 3F A2 54
3FA0:FF 20 BD 3F A2 FF 20 BD E5
3FA8:3F A9 03 CD 3D 03 F0 03 49
3FB0:4C B4 3C A9 E8 CD 3C 03 9E
3FB8:D0 F6 4C C6 3F A0 FF 88 58
3FC0:D0 FD CA D0 F8 60 A2 05 21
3FC8:A0 0B 18 20 F0 FF A0 00 28

```

3FD0:B9 A3 45 C9 00 F0 06 20 4A
3FD8:D2 FF C8 D0 F3 A9 00 85 B2
3FE0:C6 A5 C6 F0 FC 4C 82 3C 6E
3FE8:A2 05 A0 0B 18 20 F0 FF E1
3FF0:A0 00 B9 B0 45 C9 00 F0 44
3FF8:E4 20 D2 FF C8 D0 F3 18 D5
4000:AD 3C 03 69 64 8D 3C 03 32
4008:AD 3D 03 69 00 8D 3D 03 59
4010:4C 24 40 18 AD 3E 03 69 1F
4018:64 8D 3E 03 AD 3F 03 69 FF
4020:00 8D 3F 03 A2 08 A0 22 B4
4028:18 20 F0 FF AD 3D 03 AE F1
4030:3C 03 20 CD BD A2 0D A0 A3
4038:22 18 20 F0 FF AD 3F 03 1B
4040:AE 3E 03 20 CD BD 60 A9 D9
4048:0F 8D 18 D4 8D 0C D4 A9 F3
4050:07 8D 0D D4 A9 1C 8D 08 87
4058:D4 A9 00 8D 07 D4 A9 81 E6
4060:8D 0B D4 A2 0A 20 BD 3F BA
4068:A9 80 8D 0B D4 A9 E1 8D DE
4070:00 D4 A9 1E 8D 01 D4 A9 01
4078:44 8D 05 D4 A9 00 8D 06 DA
4080:D4 A9 07 8D 18 D4 A9 81 78
4088:8D 04 D4 A2 0A 20 BD 3F 21
4090:A9 80 8D 04 D4 60 A2 00 65
4098:A9 00 9D 00 D4 E8 E0 18 C5
40A0:D0 F8 60 A9 00 8D 21 D0 B7
40A8:A9 0F 8D 20 D0 A0 00 B9 38
40B0:DF 40 20 D2 FF C8 D0 F7 1F
40B8:B9 DF 41 20 D2 FF C8 D0 31
40C0:F7 B9 DF 42 20 D2 FF C8 E0
40C8:D0 F7 B9 DF 43 20 D2 FF 25
40D0:C8 D0 F7 B9 DF 44 F0 06 7C
40D8:20 D2 FF C8 D0 F5 60 93 5D
40E0:20 12 05 20 20 20 20 20 7A
40E8:20 20 20 20 20 20 20 20 69
40F0:20 20 20 20 20 20 20 20 71
40F8:20 20 20 20 20 20 20 20 79
4100:20 20 20 20 20 20 20 20 82
4108:20 92 20 12 20 92 20 20 10
4110:20 20 20 20 20 20 20 12 84
4118:97 20 20 20 20 20 20 20 56
4120:20 20 20 20 92 20 20 20 36
4128:20 20 20 20 20 20 20 12 9C
4130:05 20 92 20 20 20 20 20 73
4138:20 20 12 20 20 92 20 20 C2
4140:20 20 20 20 20 20 20 12 B4
4148:97 20 20 20 20 20 20 20 86
4150:20 20 20 20 92 20 20 20 66
4158:20 20 20 20 20 20 20 12 CC

```

4160:05 20 92 20 20 20 20 20 A3
4168:20 20 12 20 20 92 20 20 F2
4170:20 20 20 20 20 20 20 DF B2
4178:12 20 20 20 20 20 20 F3
4180:20 20 92 A9 20 20 20 20 E9
4188:20 20 20 20 20 20 12 05 D3
4190:20 92 20 20 20 20 20 20 AF
4198:20 12 20 20 92 20 20 20 2B
41A0:20 20 20 20 20 20 20 20 23
41A8:20 20 20 20 20 20 20 20 2B
41B0:20 20 20 20 20 20 20 20 33
41B8:20 20 A0 12 05 20 92 20 76
41C0:20 20 20 20 20 20 12 20 27
41C8:20 92 20 20 20 20 20 20 E7
41D0:20 20 20 20 20 20 20 20 53
41D8:20 20 20 20 20 20 20 20 5B
41E0:20 20 20 20 20 20 20 20 63
41E8:12 05 20 92 20 12 57 48 23
41F0:49 54 45 92 20 12 20 20 A8
41F8:92 20 20 20 20 20 20 20 B4
4200:20 20 20 20 20 20 20 20 84
4208:20 20 20 20 20 20 20 20 8C
4210:20 20 20 20 20 20 20 12 86
4218:20 92 20 12 53 43 4F 52 0F
4220:45 92 20 12 20 20 92 20 D7
4228:20 20 20 20 20 20 20 20 AC
4230:20 20 20 20 20 20 20 20 B4
4238:20 20 20 20 20 20 20 20 BC
4240:20 20 20 20 20 12 20 92 FE
4248:20 20 20 20 20 20 20 12 BE
4250:20 20 92 20 96 D1 20 D1 4F
4258:20 D1 20 D1 20 D1 20 D1 DC
4260:20 D1 20 D1 20 D1 20 D1 E4
4268:20 D1 20 D1 20 D1 20 D1 EC
4270:20 D1 12 05 20 92 20 20 B7
4278:30 30 30 20 20 12 20 20 D2
4280:92 96 D1 20 D1 20 D1 20 03
4288:D1 20 D1 20 D1 20 D1 20 0D
4290:D1 20 D1 20 D1 20 D1 20 15
4298:D1 20 D1 20 D1 20 D1 20 1D
42A0:12 05 20 92 20 20 20 20 7E
42A8:20 20 20 12 20 20 92 20 31
42B0:96 D1 20 D1 20 D1 20 D1 70
42B8:20 D1 20 D1 20 D1 20 D1 3D
42C0:20 D1 20 D1 20 D1 20 D1 45
42C8:20 D1 20 D1 20 D1 12 05 64
42D0:20 92 20 12 9B 42 4C 41 EE
42D8:43 4B 92 20 12 05 20 20 2B
42E0:92 96 D1 20 12 95 A9 20 EA
42E8:92 A9 9E B7 B7 B7 B7 B7 34

```

42F0:B7 B7 B7 B7 B7 B7 B7 B7 75
42F8:B7 B7 B7 B7 B7 B7 B7 95 5B
4300:DF 12 20 DF 92 96 D1 20 AF
4308:12 05 20 92 20 12 9B 53 D9
4310:43 4F 52 45 92 20 12 05 E8
4318:20 20 92 20 9E CE 96 D1 3A
4320:12 95 20 92 96 20 D1 20 3B
4328:D1 20 D1 20 D1 20 D1 20 AE
4330:D1 20 D1 20 D1 20 D1 20 B6
4338:D1 20 D1 95 12 20 92 96 10
4340:D1 9E CD 96 D1 12 05 20 7B
4348:92 20 20 20 20 20 20 20 08
4350:12 20 20 92 9E CE 96 D1 44
4358:9E CE 12 95 20 92 9E B7 BD
4360:B7 B7 B7 B7 B7 B7 B7 B7 E6
4368:B7 B7 B7 B7 B7 B7 B7 B7 EE
4370:B7 B7 B7 B7 B7 95 12 20 8B
4378:92 9E CD 96 D1 9E CD 12 C9
4380:05 20 92 20 20 30 30 30 38
4388:20 20 12 20 20 92 96 D1 B5
4390:9E CE 1C D1 12 95 20 92 74
4398:1C D1 20 D1 20 D1 20 D1 1D
43A0:20 D1 20 D1 20 D1 20 D1 27
43A8:20 D1 20 D1 20 D1 20 95 F2
43B0:12 20 92 1C D1 9E CD 96 97
43B8:D1 12 05 20 92 20 20 20 C4
43C0:20 20 20 20 12 20 20 92 49
43C8:9E CE 1C D1 12 9B A9 20 65
43D0:20 20 20 20 20 20 20 20 57
43D8:20 20 20 20 20 20 20 20 5F
43E0:20 20 20 20 20 20 20 DF 27
43E8:92 1C D1 9E CD 12 05 20 C4
43F0:92 20 20 20 20 20 20 20 B0
43F8:12 20 20 92 1C D1 12 9B A5
4400:A9 20 20 20 20 20 20 20 4D
4408:20 20 20 20 20 20 20 20 90
4410:20 20 20 20 20 20 20 20 98
4418:20 20 20 DF 92 1C D1 12 75
4420:05 20 92 20 20 20 20 20 69
4428:20 20 12 20 20 92 12 9B 18
4430:A9 20 20 20 20 20 20 20 7D
4438:20 20 20 20 20 20 20 20 C0
4440:20 20 20 20 20 20 20 20 C8
4448:20 20 20 20 20 DF 92 12 A6
4450:05 20 92 20 20 20 20 20 99
4458:20 20 12 20 20 9B 20 20 0D
4460:20 20 20 20 20 20 20 20 E8
4468:20 20 20 20 20 20 20 20 F0
4470:20 20 20 20 20 20 20 20 F8
4478:20 20 20 20 05 20 92 20 0D

```

4480:20 20 20 20 20 20 12 20 EC
4488:20 9B 20 20 20 20 20 20 EF
4490:20 20 20 20 20 20 20 20 19
4498:20 20 20 20 20 20 20 20 21
44A0:20 20 20 20 20 20 20 20 29
44A8:05 20 92 20 20 20 20 20 F1
44B0:20 20 12 20 20 98 20 20 59
44B8:20 20 20 20 20 20 20 20 41
44C0:20 20 20 20 20 20 20 20 49
44C8:20 20 20 20 20 20 20 20 51
44D0:20 20 20 20 05 20 92 20 65
44D8:20 20 20 20 20 20 12 20 45
44E0:20 92 20 97 DF 12 20 92 B5
44E8:A9 20 20 20 20 20 20 20 36
44F0:20 20 20 20 20 20 20 20 79
44F8:20 20 20 20 20 20 20 20 DF 41
4500:12 20 92 A9 20 12 05 20 FB
4508:92 20 20 20 20 20 20 20 CB
4510:12 20 20 92 20 97 20 12 8A
4518:20 20 20 20 20 52 49 4E EB
4520:47 53 49 44 45 20 20 4B C6
4528:41 52 41 54 45 20 20 20 60
4530:20 20 92 20 20 12 05 20 9A
4538:92 20 20 20 20 20 20 20 FB
4540:12 20 20 92 20 97 20 12 BA
4548:20 92 20 20 20 20 20 20 6F
4550:20 20 20 20 20 20 20 20 DA
4558:20 20 20 20 20 20 20 20 E2
4560:20 20 12 20 92 20 20 12 AE
4568:05 20 92 20 20 20 20 20 B3
4570:20 20 12 20 92 20 12 05 95
4578:20 20 20 20 20 20 20 20 03
4580:20 20 20 20 20 20 20 20 0B
4588:20 20 20 20 20 20 20 20 13
4590:20 20 20 20 20 20 20 20 1B
4598:20 20 20 20 20 20 91 91 77
45A0:92 8E 00 9C 57 48 49 54 A4
45A8:45 20 57 49 4E 53 21 00 5F
45B0:9C 42 4C 41 43 4B 20 57 96
45B8:49 4E 53 21 00 20 00 00 78

Eagles and Gators

Philip I. Nelson

Featuring 16 sprites on a split screen, great sound effects, and realistic animation, this all machine language, two-player action game pits creature against creature as each tries to shove the other into oblivion. Requires at least one joystick.

“Eagles and Gators,” an intense two-player action game, displays 16 moving sprites at all times—double the normal limit—and includes highly realistic animation, too.

Since it’s written completely in machine language, you’ll need to type it in with the “MLX” machine language entry program found in Appendix D. Here are the addresses you need to enter the Eagles and Gators data:

Starting address: 0801

Ending address: 1CE8

Follow the MLX instructions carefully, and be sure to save a copy of the data to disk or tape before you leave MLX.

To play the game, plug two joysticks into the joystick ports (see the instructions below for playing with only one joystick). Although the program is written in machine language, you load and run it just as you would a BASIC program, with **LOAD “filename”,8** (or **,1** for tape), followed by **RUN**.

Eagles and Gators begins with an animated title screen which asks whether you want to read the game instructions. Press Y to view the instruction screen or N to skip it. Next, you’re shown the game screen and are prompted to choose one of the nine skill levels. If you’ve never played the game before, you’ll probably want to press 1 to choose the easiest level.

Flyers Against Swimmers

When play begins, the screen is split into upper and lower color zones—sky-blue for the eight high-flying eagles and murky black for the eight gators swimming below. Each player

controls a group of eight sprites. The game is written to be played with two joysticks, but you can substitute keypresses to control the gators, normally controlled by joystick 1. Press the CTRL key to move left or the 2 key to move right. The space bar serves as a fire button.

Your sprites are in constant, realistic motion, flying or swimming up and down in their zones, rebounding when they hit the border between the two worlds. The flashing colored sprite in your group is the one controlled by your joystick. To switch control to a different sprite, push the joystick left or right.

The goal of the game is to expand your zone and shove your opponents off the screen. This is done by convincing your creatures to push against the sky/water border. Press the fire button just as your flashing eagle or gator bumps the border, and it "clamps" on and starts to shove. Whenever you clamp successfully, your opponent's screen flashes red, and all your creatures thrash momentarily with excitement.

Your zone expands whenever you have more sprites pushing than your opponent. The game ends when one side or the other takes over the entire screen. That sounds simple, but it takes precise timing to clamp just one sprite. If you press the fire button at the wrong time, your sprite moves away from the border, regardless of its initial direction. If you press the fire button when your flashing sprite's already stuck to the border, it bounces back to the farthest end of your zone.

Your opponent can also unclamp one of your creatures by clamping one of his or her own creatures directly across the border from yours. The ability to unclamp an opponent's creatures at any time adds an extra element of strategy and makes it possible to turn the tables on your opponent even when defeat seems almost certain. The game is designed to reward concentration and careful play, and to discourage wild tactics such as holding the fire button down constantly. At best, you'll prevent your creatures from approaching the border. At worst, you'll inadvertently unclamp those from your side who are already pushing.

When you win a game, the screen clears to your color, and a joyful, double-sized sprite from the winning side performs a victory dance across the screen. At this point you can pick a new skill level for the next round or quit by pressing the Q key. After you've exited, the game can be restarted by typing **SYS 16384** and pressing RETURN. To stop in midplay, press RUN/STOP-RESTORE.

Displaying 16 Sprites

Space doesn't permit a detailed explanation of every routine in this program, but you may be curious about how to display more than eight sprites at a time, something often mentioned, but rarely seen. To understand how it's done, you'll need to know a little about raster displays.

Your TV or monitor makes a picture by scanning its screen with a moving electron beam called a *raster*. This is done line by line, from top to bottom, 60 times a second. The visible portion of a TV picture is made up of 200 raster lines, numbered from 50 at the top through 249 at the bottom. The same numbering scheme is used for locating sprites vertically on the screen, so raster line 100, for example, matches up with sprite vertical location 100. This correspondence between raster lines and sprite positions greatly simplifies the task of creating more than the usual number of sprites.

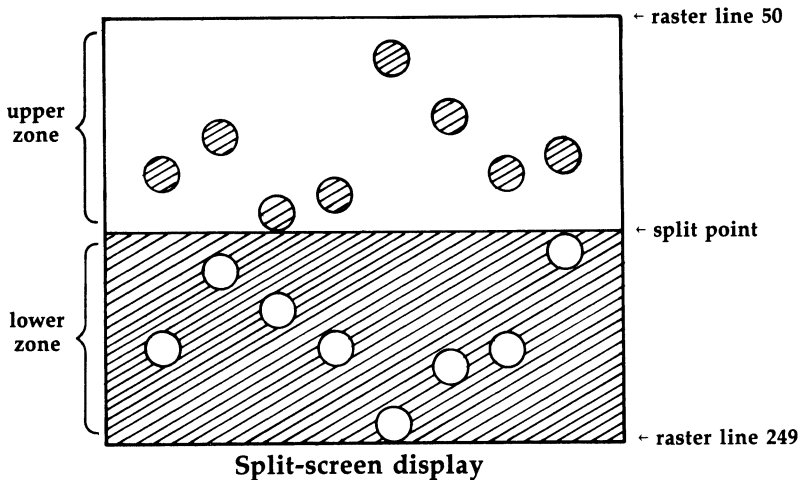
The Commodore 64 keeps track of the raster's vertical location in a special memory location (register) at 53266 (\$D012). BASIC is too slow to make much use of it, but machine language (ML) is fast enough to let us check this register repeatedly and monitor the position of the raster beam as it scans down the screen. This information is used to create a split screen as well as to double the usual number of sprites.

Split Screens

To make the two-color split screen, we first establish an arbitrary "split" point somewhere near the midpoint of the screen (say, at line 100), then repeatedly compare the raster's location to that value. When the raster value is less, we know it's above the split, so we color the screen cyan to create the sky zone. When the raster value is greater, that means it's dipped below the split point, so we color the screen black for the water zone. Remember, moving downward on the screen increases the raster (or sprite) vertical location value. By changing the border color along with the screen background color, we can make the split extend the full width of the screen for heightened realism.

The same sort of monitoring makes it possible to display 16 sprites. When the raster's above the split, we display all of the 64's 8 sprites at vertical locations in the upper color zone. When it drops below the split, we display the same 8 sprites

at locations in the lower zone. The two sets of sprites are never actually displayed at the same time. The program just changes their locations so fast that your eye sees 16 solid figures, without a trace of flicker.



Shadow Registers

If you stopped at this point, you'd have two identical groups of eight sprites—upper and lower—sitting motionless on a split screen. Nice, but boring.

To exploit the effect in a game, we want the two sprite groups to look different and move independently. Unfortunately, the Commodore 64 still has only eight sets of sprite control registers for handling the sprites' shapes, colors, and so on. Remember, we're just flipping the same group of eight sprites back and forth at very high speed between different vertical locations. Thus, top sprite 1 doesn't just look the same as bottom sprite 1—it *is* the same sprite.

The minute you change the shape or color of any sprite in the upper group, its lower twin immediately follows suit, and vice versa.

What we really need is 16 sets of sprite control registers. Commodore didn't design 16 sprites for the 64, but you can achieve much the same effect through software. Here's an outline of how it's done in this program.

First, we set aside some free memory to use as "shadow" registers in place of the actual sprite control registers. Since

none of the eagle or gator sprites move horizontally, we don't need to duplicate the 8 horizontal position registers. However, every sprite must be able to move freely up or down and change its shape and color at any time. Thus, for each group of 8 sprites we need 8 registers for vertical position, 8 for color, and 8 more to hold shape data pointers. Three sets of 8 make 24. Double that for the second set of 8 sprites, and you now have 48 shadow registers.

To make each sprite appear in the correct spot, we must transfer the contents of each shadow register into the corresponding control register at precisely the right instant. When the raster beam is in the upper portion of the screen, we'll take data from the eagles' set of shadow registers and store it in the actual sprite control registers. When the raster creeps down below the split, we'll fetch gator data instead and plug that into the control registers.

Of course, it's not necessary to do this on a two-color split screen. This particular program changes screen colors at the split point to accentuate the division between sides. But in another application, you might prefer to leave out the color-changing code. The split location is just an invisible reference point that tells you when it's time to flip the sprites from one zone into the other.

Coasting on the Interrupt

Expressed in this simple, schematic form, the job of displaying 16 sprites sounds quite simple. When the raster beam is above the midpoint, make the sprites appear in the upper area. When it's below the split, move them down into the lower area and give them new shapes and colors. The process is uncomplicated in theory. But don't bother trying to do this in BASIC, which is many times too slow to read the raster's position accurately. This program uses an interrupt-driven routine to manage the 16-sprite effect. Although interrupt handling goes beyond the scope of this article, the technique essentially wedges our custom sprite display routines into the computer's operating system along with its normal hardware-servicing routines, which do background tasks such as scanning the computer's keyboard. Conveniently, the interrupt-driven routine is executed 60 times per second, exactly the same frequency at which the screen is redrawn. As a result, the sprite display and color split are quite stable.

The advantage of driving an ML routine on the hardware interrupt is that it becomes completely automatic. Sixty times every second, the computer executes your custom routine along with its own housekeeping routines. An interrupt-driven routine seems to coast along all by itself, without slowing down other program events. In effect, we've enhanced the 64 by adding the ability to display 16 sprites, with the option of a two-color split screen. Because the custom routines operate in the background, the shadow sprite control registers seem to work exactly like the actual control registers. We can change the color, shape, or position of any of our 16 sprites by plugging different values into their shadow control registers. To move our border up or down, we just change the value for our split point, and everything else follows accordingly.

The price you pay for these new features is that the sprites can't leave their respective zones. If you try to move an eagle sprite down into the gator zone, the custom display routine automatically gives it a gator shape and color, and the sky is minus one eagle. The same thing happens in reverse if you move a gator into the upper zone. If you move any sprite so that it overlaps the border, you get a weird hybrid creature with an eagle-shaped upper body and a gator tail. It's easy to forget because the shapes look so real, but the 64 still has only 8 sprites built into hardware. Using similar techniques, it's possible to display as many as 64 sprites on the screen at once. All you need to do is divide the screen vertically into additional zones, add extra shadow registers, and modify the display routines to handle them. However, the same general rules apply. Each group of sprites is confined to its own vertical zone, and you may not have more than 8 sprites occupying the same screen raster line.

Eagles and Gators

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```

0801:18 08 00 00 9E 32 30 37 74
0809:36 3A 8F 22 14 14 14 14 04
0811:45 41 47 4C 45 53 00 00 39
0819:00 00 00 A0 00 A9 56 85 0C
0821:03 A9 08 85 04 A9 00 85 C2
0829:05 A9 40 85 06 A9 E3 85 AA
0831:FB A9 1C 85 FC A5 03 E5 EF
0839:FB A5 04 E5 FC B0 13 B1 12
0841:03 91 05 E6 03 D0 02 E6 8C

```

0849:04 E6 05 D0 02 E6 06 4C C6
0851:36 08 4C 00 40 A9 3B 85 AC
0859:FB A9 03 85 FC A0 BF A9 1E
0861:00 91 FB 88 D0 F9 A9 FF 9F
0869:85 FB A9 D3 85 FC A0 19 28
0871:A9 00 91 FB 88 D0 F9 A9 6D
0879:93 20 D2 FF A9 00 8D 20 3E
0881:D0 8D 21 D0 18 A0 0B A2 8A
0889:05 20 F0 FF A9 00 85 FB 96
0891:A9 45 85 FC 20 AD 42 18 9C
0899:A0 0C A2 08 20 F0 FF A9 40
08A1:14 85 FB A9 45 85 FC 20 91
08A9:AD 42 20 1E 43 18 A0 0A CC
08B1:A2 0E 20 F0 FF A9 26 85 22
08B9:FB A9 45 85 FC 20 AD 42 39
08C1:20 CD 42 18 A0 0D A2 11 AE
08C9:20 F0 FF A9 3B 85 FB A9 52
08D1:45 85 FC A0 00 20 BD 42 CD
08D9:AD 04 DC 8D 86 02 B1 FB D1
08E1:C9 7E F0 07 20 D2 FF C8 1A
08E9:4C 80 40 20 9F FF 20 E4 6C
08F1:FF C9 00 F0 CE C9 59 F0 C4
08F9:78 C9 4E F0 03 4C 6E 40 F7
0901:A9 93 20 D2 FF 20 74 42 A9
0909:20 6D 41 18 A0 11 A2 09 C7
0911:20 F0 FF A9 4C 85 FB A9 24
0919:45 85 FC 20 AD 42 18 A0 18
0921:14 A2 0B 20 F0 FF A9 5E 82
0929:85 FB A9 45 85 FC 20 AD 94
0931:42 18 A0 11 A2 0F 20 F0 12
0939:FF A9 68 85 FB A9 45 85 B1
0941:FC 20 AD 42 20 9F FF 20 53
0949:E4 FF C9 00 F0 F6 C9 51 4F
0951:D0 03 4C E2 FC C9 31 B0 66
0959:03 4C EF 40 C9 3A B0 E4 7F
0961:38 E9 30 8D 3F 03 A9 93 D5
0969:20 D2 FF 20 EA 4B 4C 0F 6E
0971:54 A9 93 20 D2 FF 20 74 D7
0979:42 18 A0 04 A2 00 20 F0 4D
0981:FF A9 7C 85 FB A9 45 85 7C
0989:FC A9 05 85 FD A9 47 85 28
0991:FE A0 00 B1 FB 20 D2 FF 6C
0999:A5 FB C9 FF 90 02 E6 FC 0E
09A1:E6 FB A5 FB C5 FD A5 FC 09
09A9:E5 FE 90 E7 20 EC 50 AD 02
09B1:67 03 C9 00 F0 09 AD 6A E2
09B9:03 C9 00 F0 02 D0 ED 4C 4A
09C1:AB 40 A9 40 85 FB A9 03 65
09C9:85 FC A9 E4 85 FD A0 08 CE
09D1:A5 FD 91 FB E6 FD E6 FD 23

```

09D9:88 C0 01 D0 F3 A9 EA 91 3B
09E1:FB A0 08 A9 48 85 FB A9 AF
09E9:0D 91 FB 88 D0 F9 A9 50 01
09F1:85 FB A9 F2 85 FD A0 08 97
09F9:A5 FD 91 FB E6 FD E6 FD 4B
0A01:88 C0 01 D0 F3 A9 F2 91 74
0A09:FB A9 59 85 FB A0 07 A9 23
0A11:00 91 FB 88 D0 F9 A9 5B AE
0A19:8D 6C 03 A9 4E 8D 6E 03 92
0A21:A9 03 8D 6D 03 8D 6F 03 83
0A29:A9 71 85 FB A9 32 85 FD FD
0A31:A0 08 A5 FD 91 FB 18 69 42
0A39:0A 85 FD 88 D0 F4 A9 79 23
0A41:85 FB A9 E6 85 FD A0 08 28
0A49:A5 FD 91 FB 38 E9 0A 85 A4
0A51:FD 88 D0 F4 A9 81 85 FB 4A
0A59:A0 10 A9 FF 91 FB 88 A9 2E
0A61:01 91 FB 88 D0 F4 A9 82 92
0A69:8D 92 03 A9 8A 8D 94 03 9A
0A71:A9 72 8D 96 03 A9 7A 8D 53
0A79:98 03 A9 03 8D 93 03 8D 4E
0A81:95 03 8D 97 03 8D 99 03 D0
0A89:A9 1E 8D 00 D0 A9 46 8D F2
0A91:02 D0 A9 6E 8D 04 D0 A9 BE
0A99:96 8D 06 D0 A9 BE 8D 08 95
0AA1:D0 A9 E6 8D 0A D0 A9 0F 34
0AA9:8D 0C D0 A9 37 8D 0E D0 19
0AB1:A9 C0 8D 10 D0 A9 00 8D 38
0AB9:17 D0 8D 1D D0 A9 01 8D CD
0AC1:18 D4 EA A9 FF 8D 15 D0 40
0AC9:60 78 A9 00 8D 0E DC A9 69
0AD1:8C 8D 14 03 A9 4A 8D 15 E8
0AD9:03 A9 0F 8D 88 4A A9 96 EB
0AE1:8D 89 4A A9 00 8D 8A 4A 98
0AE9:A9 03 8D 8B 4A AD 11 D0 F9
0AF1:29 7F 8D 11 D0 AD 89 4A D7
0AF9:8D 12 D0 A9 01 8D 1A D0 51
0B01:58 60 A0 00 B1 FB C9 7E FE
0B09:F0 07 20 D2 FF C8 4C AF F5
0B11:42 60 EE 01 D0 CE 03 D0 E6
0B19:CE 0D D0 EE 0F D0 20 CA A9
0B21:44 60 A9 1E 8D 00 D0 A9 40
0B29:46 8D 02 D0 A9 C0 8D 10 8E
0B31:D0 A9 0F 8D 0C D0 A9 37 03
0B39:8D 0E D0 A9 07 8D 27 D0 DB
0B41:8D 2E D0 A9 0D 8D 28 D0 1E
0B49:8D 2D D0 A9 C3 8D 15 D0 75
0B51:A9 FF 8D F8 07 8D FF 07 F2
0B59:A9 E4 8D F9 07 A9 EA 8D 11
0B61:FE 07 60 EA EA A0 00 B1 FE

```

```

0B69:FB 91 FD C8 C0 40 F0 02 19
0B71:D0 F5 60 A9 08 85 FB A9 0C
0B79:47 85 FC A9 C0 85 FD A9 90
0B81:3A 85 FE 20 10 43 A9 80 59
0B89:85 FD 20 10 43 A9 48 85 BD
0B91:FB A9 00 85 FD A9 3B 85 FA
0B99:FE 20 10 43 A9 40 85 FD C4
0BA1:A9 3A 85 FE 20 10 43 A9 2D
0BA9:88 85 FB A9 00 85 FD 20 B1
0BB1:10 43 A9 40 85 FD A9 3B 8C
0BB9:85 FE 20 10 43 A9 C8 85 2F
0BC1:FB A9 80 85 FD 20 10 43 7C
0BC9:A9 C0 85 FD A9 39 85 FE B1
0BD1:20 10 43 A9 08 85 FB A9 F6
0BD9:48 85 FC A9 80 85 FD 20 E5
0BE1:10 43 A9 C0 85 FD A9 3B C4
0BE9:85 FE 20 10 43 A9 48 85 5E
0BF1:FB A9 40 85 FD A9 39 85 5F
0BF9:FE 20 10 43 A9 00 85 FD 24
0C01:A9 3C 85 FE 20 10 43 A9 0F
0C09:88 85 FB A9 40 85 FD 20 15
0C11:10 43 A9 00 85 FD A9 39 E7
0C19:85 FE 20 10 43 A9 C8 85 90
0C21:FB A9 48 85 FC A9 40 85 97
0C29:FD A9 3E 85 FE 20 10 43 A6
0C31:A9 00 85 FD 20 10 43 A9 20
0C39:08 85 FB A9 49 85 FC A9 D4
0C41:80 85 FD 20 10 43 A9 C0 5E
0C49:85 FD A9 3D 85 FE 20 10 25
0C51:43 A9 48 85 FB A9 80 85 E3
0C59:FD 20 10 43 A9 C0 85 FD 08
0C61:A9 3E 85 FE 20 10 43 A9 EF
0C69:88 85 FB A9 00 85 FD A9 FC
0C71:3F 85 FE 20 10 43 A9 40 8D
0C79:85 FD A9 3D 85 FE 20 10 55
0C81:43 A9 C8 85 FB A9 00 85 23
0C89:FD 20 10 43 A9 40 85 FD 36
0C91:A9 3F 85 FE 20 10 43 A9 60
0C99:08 85 FB A9 4A 85 FC A9 3D
0CA1:C0 85 FD A9 3C 85 FE 20 EB
0CA9:10 43 A9 80 85 FD A9 3F 8E
0CB1:85 FE 20 10 43 A9 48 85 28
0CB9:FB A9 C0 85 FD 20 10 43 7E
0CC1:A9 80 85 FD A9 3C 85 FE A7
0CC9:20 10 43 A9 0F 8D 05 D4 86
0CD1:8D 0C D4 8D 13 D4 A9 F0 57
0CD9:8D 06 D4 8D 0D D4 8D 14 98
0CE1:D4 A9 3F 8D 00 D4 A9 18 4E
0CE9:8D 01 D4 A9 42 8D 07 D4 69
0CF1:A9 06 8D 08 D4 A9 0C 8D 85

```

0CF9:0E D4 A9 06 8D 0F D4 A9 DF
0D01:15 8D 04 D4 8D 0B D4 8D A6
0D09:12 D4 A9 3F 8D 18 D4 A9 AA
0D11:F7 8D 17 D4 A9 22 8D 16 C1
0D19:D4 A9 0F 8D 15 D4 60 AD 2D
0D21:3E 03 C9 2F F0 04 EE 3E FA
0D29:03 60 A9 00 8D 3E 03 EE 6C
0D31:3F 03 AD 3F 03 8D 0E D4 94
0D39:CE 40 03 AD 40 03 8D 16 45
0D41:D4 AD 3F 03 C9 35 90 06 93
0D49:A9 18 8D 01 D4 60 AD 3F C2
0D51:03 8D 01 D4 60 8E 08 9F AA
0D59:00 20 45 41 47 4C 45 53 81
0D61:20 26 20 47 41 54 4F 52 D9
0D69:53 20 92 7E 00 00 00 00 6F
0D71:00 00 00 00 00 00 00 00 8B
0D79:00 00 00 9F 57 41 4E 54 3E
0D81:20 49 4E 53 54 52 55 43 D6
0D89:54 49 4F 4E 53 20 3F 7E 07
0D91:50 52 45 53 53 20 12 59 DE
0D99:92 20 4F 52 20 12 4E 92 8C
0DA1:7E 90 50 49 43 4B 20 53 98
0DA9:4B 49 4C 4C 20 4C 45 56 1D
0DB1:45 4C 7E 28 20 31 20 63 3D
0DB9:20 39 20 29 7E 9E 50 52 2A
0DC1:45 53 53 20 12 51 92 20 DA
0DC9:20 54 4F 20 51 55 49 54 BB
0DD1:7E 90 12 77 11 9D 20 11 59
0DD9:9D 9D 00 00 20 20 20 92 7E
0DE1:63 69 11 9D 7B 91 91 50 99
0DE9:4C 55 47 20 4A 4F 59 53 FF
0DF1:54 49 43 4B 53 20 49 4E A1
0DF9:2E 2E 11 11 7B 91 9D 75 BC
0E01:63 63 63 63 12 20 20 20 BB
0E09:91 9D 9D 20 91 9D 69 8D 6E
0E11:11 11 11 00 20 12 50 4F 55
0E19:52 54 20 32 20 46 4F 52 A5
0E21:20 45 41 47 4C 45 53 92 EB
0E29:20 20 20 12 50 4F 52 54 3B
0E31:20 31 20 46 4F 52 20 47 5D
0E39:41 54 4F 52 53 92 8D 11 2B
0E41:00 20 43 4C 41 4D 50 20 92
0E49:59 4F 55 52 20 43 52 45 AD
0E51:41 54 55 52 45 53 20 54 FE
0E59:4F 20 54 48 45 20 42 4F B2
0E61:52 44 45 52 20 42 59 8D CF
0E69:11 20 50 52 45 53 53 49 AC
0E71:4E 47 20 12 46 49 52 45 EC
0E79:20 42 55 54 54 4F 4E 92 35
0E81:20 57 48 45 4E 20 54 48 C4

```

0E89:45 59 20 48 49 54 20 49 4C
0E91:54 69 11 9D 62 11 9D 9D 5E
0E99:6D 62 6E 11 9D 9D 2A 11 AC
0EA1:9D 7A 8D 91 91 20 42 4F D6
0EA9:52 44 45 52 20 4D 4F 56 F8
0EB1:45 53 20 57 48 45 4E 20 D2
0EB9:53 49 44 45 53 20 41 52 9E
0EC1:45 20 55 4E 45 51 55 41 73
0EC9:4C 8D 9F 11 11 20 20 20 DD
0ED1:4D 4F 56 45 20 52 49 47 AB
0ED9:48 54 20 4F 52 20 4C 45 19
0EE1:46 54 20 54 4F 20 43 4F 50
0EE9:4E 54 52 4F 4C 20 41 8D 74
0EF1:11 20 20 20 20 20 20 20 86
0EF9:20 20 44 49 46 46 45 52 73
0F01:45 4E 54 20 43 52 45 41 11
0F09:54 55 52 45 8D 11 11 20 38
0F11:20 05 57 49 4E 20 42 59 D0
0F19:20 50 55 53 48 49 4E 47 86
0F21:20 4F 50 50 4F 4E 45 4E BE
0F29:54 53 20 4F 46 46 20 53 1E
0F31:43 52 45 45 4E 8D 11 11 5E
0F39:9E 20 20 20 20 20 12 50 AA
0F41:52 45 53 53 20 46 49 52 78
0F49:45 20 42 55 54 54 4F 4E 90
0F51:20 54 4F 20 42 45 47 49 7F
0F59:4E 92 00 00 00 00 03 80 C9
0F61:00 0F C0 02 1F 80 06 37 B9
0F69:80 03 7D 80 01 FF A0 00 89
0F71:FF 30 00 7F E0 00 7F 00 99
0F79:00 7F E0 00 FF 60 01 FE 16
0F81:00 03 3E 00 06 3E 00 04 55
0F89:1E 00 00 1C 00 00 1E 00 B4
0F91:00 0E 08 00 0F 30 00 03 70
0F99:E0 00 01 80 00 00 0C 00 68
0FA1:00 1E 00 00 3E 00 04 3F 80
0FA9:00 0E 6D 20 03 FF 30 00 73
0FB1:FF 60 00 FF C0 00 7F 00 EC
0FB9:00 FF C0 00 FF 60 03 FE 76
0FC1:00 06 7C 00 00 7C 00 00 E2
0FC9:3C 00 00 3C 00 00 3C 00 42
0FD1:00 1C 00 00 1E 00 00 0F F6
0FD9:E0 00 03 80 FF 00 18 00 01
0FE1:00 3C 00 00 3C 00 04 7E 77
0FE9:20 0C 5A 30 06 FF 60 03 5D
0FF1:FF C0 00 FF 00 00 FF 00 40
0FF9:00 FF 00 03 FF C0 06 FF 57
1001:60 00 7E 00 00 7C 00 00 13
1009:7C 00 00 3C 00 00 3C 00 A3
1011:00 3C 00 00 3C 00 00 1E 40

```



```
1019:00 00 07 80 00 00 18 00 52
1021:00 3C 00 00 3C 00 04 7E B8
1029:20 0C 5A 30 06 FF 60 03 9E
1031:FF C0 00 FF 00 00 FF 00 81
1039:00 FF 00 03 FF C0 06 FF 98
1041:60 00 7E 00 00 7E 00 00 5B
1049:7E 00 00 3C 00 00 3C 00 E4
1051:00 3C 00 00 3C 00 00 18 7A
1059:00 00 18 00 F7 00 18 00 6C
1061:00 3C 00 00 3C 00 04 7E F8
1069:20 0C 5A 30 06 FF 60 03 DE
1071:FF C0 00 FF 00 00 FF 00 C1
1079:00 FF 00 03 FF C0 06 FF D8
1081:60 00 7E 00 00 3E 00 00 9A
1089:3E 00 00 3C 00 00 3C 00 05
1091:00 3C 00 00 3C 00 00 78 1B
1099:00 01 E0 00 00 00 30 00 76
10A1:00 78 00 00 7C 00 00 FC C0
10A9:20 04 B6 70 0C FF C0 06 A0
10B1:FF 00 03 FF 00 00 FE 00 30
10B9:03 FF 00 06 FF 00 00 7F 3B
10C1:C0 00 3E 60 00 3E 00 00 09
10C9:3C 00 00 3C 00 00 3C 00 44
10D1:00 38 00 00 78 00 07 F0 C2
10D9:00 01 C0 00 FF 01 C0 00 D7
10E1:03 F0 00 01 F8 40 01 EC 87
10E9:60 01 BE C0 05 FF 80 0C 93
10F1:FF 00 07 FE 00 00 FE 00 E0
10F9:07 FE 00 06 FF 00 00 7F 3D
1101:80 00 7C C0 00 7C 60 00 B1
1109:78 20 00 38 00 00 78 00 E3
1111:10 70 00 0C F0 00 07 C0 6E
1119:00 01 80 00 00 00 00 00 8B
1121:00 00 00 00 00 00 00 00 43
1129:00 0E 00 70 1F 3C F8 7F 31
1131:FF FE 7E 7E 7E FC 99 3F 25
1139:EF 18 F7 8F 3C F1 C7 E7 72
1141:E3 07 C3 E0 03 C3 C0 07 4D
1149:E7 E0 1E FF 78 0E 3C 30 FF
1151:00 00 00 00 00 00 00 00 73
1159:00 00 00 00 FF 00 00 00 7B
1161:00 00 00 00 00 00 00 00 83
1169:00 1E 00 78 7F BD FE FF 8B
1171:FF FF DE 7E 7B 8E DB 71 96
1179:0F 18 F0 03 18 C0 07 FF 49
1181:E0 07 81 E0 07 C3 E0 03 20
1189:FF C0 07 FF E0 3E 3C 7C B1
1191:1C 00 38 00 00 00 00 00 C8
1199:00 00 00 00 FF 00 00 00 BB
11A1:00 00 00 00 00 00 30 00 24
```

```
11A9:0C FE 00 7F FF 99 FF DF CF
11B1:FF FB 1E 3C 78 0E FF 70 C6
11B9:07 18 E0 07 18 E0 0F FF 54
11C1:F0 0F 18 F0 07 81 E0 07 39
11C9:FF E0 07 FF E0 1E 7E 78 F9
11D1:3E 00 7C 20 00 04 00 00 B4
11D9:00 00 00 00 00 00 00 00 FB
11E1:00 00 00 00 00 00 F8 00 F5
11E9:1F FE 00 7F 7F 81 FE 3F 92
11F1:DB FC 1F 3C F8 0E FF 70 59
11F9:07 99 E0 0F 18 F0 0F FF B5
1201:F0 0E 7E 70 07 18 E0 03 55
1209:C3 C0 0E FF E0 0E 7E 70 AD
1211:3E 00 7C 22 00 44 00 00 17
1219:00 00 00 00 FF 00 00 00 3D
1221:C0 00 03 E0 00 07 7C 00 29
1229:3E 7F 81 FE 3F 81 FC 1F 85
1231:DB F8 06 FF 60 03 3C C0 8A
1239:07 99 E0 0E 18 70 0F FF E4
1241:F0 0C FF 30 07 3C E0 03 D1
1249:81 C0 03 FF C0 0F 7E F0 EE
1251:18 00 18 3E 00 7C 32 00 BE
1259:4C 00 00 00 00 C0 00 03 A9
1261:C0 00 03 70 00 0E 7E 00 82
1269:7E 3F 81 FC 1F 81 F8 1F AC
1271:C3 F8 06 FF 60 03 7E C0 43
1279:07 99 E0 0E 18 70 0F DB 01
1281:F0 0C FF 30 06 3C 60 03 09
1289:81 C0 03 FF C0 0E 7E E0 1B
1291:1C 00 38 3F 00 FC 31 00 15
1299:8C 30 00 0C FF 1C 00 38 79
12A1:30 00 0C 38 00 1C 3E 00 CF
12A9:7C 3F 81 FC 1F C3 F8 0F E4
12B1:C3 F0 06 FF 60 03 7E C0 81
12B9:07 BD E0 0E 18 70 0B DB 42
12C1:D0 0C FF 30 06 3C 60 03 39
12C9:81 C0 03 FF C0 07 7E E0 3F
12D1:1C 00 38 3E 00 7C 33 00 47
12D9:CC 18 00 18 00 0F 7C 00 21
12E1:03 AC 12 D0 A2 02 CC 88 41
12E9:4A F0 07 88 CC 88 4A F0 E6
12F1:01 CA BD 89 4A 8D 21 D0 35
12F9:8D 20 D0 BD 87 4A 8D 12 75
1301:D0 AD 19 D0 8D 19 D0 E0 7E
1309:01 D0 06 20 C8 4A 4C C2 71
1311:4A 20 59 4B 20 EA FF 68 59
1319:A8 68 AA 68 40 AD 7A 03 3A
1321:8D 01 D0 AD 7B 03 8D 03 49
1329:D0 AD 7C 03 8D 05 D0 AD B2
1331:7D 03 8D 07 D0 AD 7E 03 36
```

1339:8D 09 D0 AD 7F 03 8D 0B 8B
1341:D0 AD 80 03 8D 0D D0 AD 6B
1349:81 03 8D 0F D0 AD 41 03 56
1351:8D F8 07 AD 42 03 8D F9 6B
1359:07 AD 43 03 8D FA 07 AD 1B
1361:44 03 8D FB 07 AD 45 03 58
1369:8D FC 07 AD 46 03 8D FD A8
1371:07 AD 47 03 8D FE 07 AD C3
1379:48 03 8D FF 07 AD 49 03 BA
1381:8D 27 D0 AD 4A 03 8D 28 CE
1389:D0 AD 4B 03 8D 29 D0 AD 7D
1391:4C 03 8D 2A D0 AD 4D 03 CD
1399:8D 2B D0 AD 4E 03 8D 2C 0C
13A1:D0 AD 4F 03 8D 2D D0 AD 26
13A9:50 03 8D 2E D0 60 AD 72 23
13B1:03 8D 01 D0 AD 73 03 8D B8
13B9:03 D0 AD 74 03 8D 05 D0 BB
13C1:AD 75 03 8D 07 D0 AD 76 A2
13C9:03 8D 09 D0 AD 77 03 8D E1
13D1:0B D0 AD 78 03 8D 0D D0 28
13D9:AD 79 03 8D 0F D0 AD 51 D6
13E1:03 8D F8 07 AD 52 03 8D C6
13E9:F9 07 AD 53 03 8D FA 07 05
13F1:AD 54 03 8D FB 07 AD 55 E9
13F9:03 8D FC 07 AD 56 03 8D 6F
1401:FD 07 AD 57 03 8D FE 07 68
1409:AD 58 03 8D FF 07 AD 59 28
1411:03 8D 27 D0 AD 5A 03 8D 7A
1419:28 D0 AD 5B 03 8D 29 D0 66
1421:AD 5C 03 8D 2A D0 AD 5D BD
1429:03 8D 2B D0 AD 5E 03 8D 23
1431:2C D0 AD 5F 03 8D 2D D0 C8
1439:AD 60 03 8D 2E D0 60 A0 9F
1441:00 A9 00 85 FB A9 D4 85 E1
1449:FC A9 00 91 FB C0 0F F0 65
1451:04 C8 4C F4 4B A9 0F 8D 33
1459:18 D4 A9 00 8D 17 D4 A9 14
1461:60 8D 05 D4 A9 19 8D 13 EA
1469:D4 A9 0F 8D 00 D4 8D 07 96
1471:D4 8D 0E D4 A9 0F 8D 01 1C
1479:D4 A9 4F 8D 08 D4 A9 4F 6F
1481:8D 0F D4 A9 00 8D 04 D4 7C
1489:A9 00 8D 0B D4 A9 00 8D C3
1491:12 D4 A9 00 8D 06 D4 8D E8
1499:0D D4 8D 14 D4 A9 60 8D 0C
14A1:0C D4 60 A9 60 8D 05 D4 C3
14A9:AD 89 4A 4A 4A 4A 85 FB 7B
14B1:A9 24 E5 FB 8D 01 D4 A9 F7
14B9:60 8D 05 D4 A9 20 8D 04 50
14C1:D4 A9 21 8D 04 D4 60 A9 99

```

14C9:60 8D 0C D4 AD 89 4A 4A C6
14D1:4A 4A 8D 08 D4 A9 80 8D BF
14D9:0B D4 A9 81 8D 0B D4 60 AC
14E1:A9 2A 8D 3C 03 A9 10 8D 4B
14E9:12 D4 A9 15 8D 12 D4 60 95
14F1:A9 04 8D 3C 03 A9 10 8D D1
14F9:12 D4 A9 15 8D 12 D4 60 A5
1501:A9 09 8D 8A 4A A5 FB 85 03
1509:FD A5 FC 85 FE 98 8D 3D 46
1511:03 A9 10 8D 12 D4 A9 15 4E
1519:8D 12 D4 A2 1F A9 00 8D 80
1521:0F D4 20 97 4D 8A 8D 3E 73
1529:03 0A 0A 0A 0A 8D 0F D4 B2
1531:A0 12 20 5B 4D AD 3E 03 8A
1539:AA CA D0 E6 AD 3D 03 A8 05
1541:A5 FD 85 FB A5 FE 85 FC 5F
1549:A9 E0 8D 0F D4 A9 85 8D 09
1551:12 D4 A9 00 8D 8A 4A 60 7A
1559:A9 09 8D 8B 4A A5 FB 85 6B
1561:FD A5 FC 85 FE 98 8D 3D 9E
1569:03 A9 10 8D 12 D4 A9 15 A6
1571:8D 12 D4 A2 18 A9 08 8D B0
1579:0F D4 20 75 4D 8A 8D 3E A9
1581:03 0A 0A EA EA 8D 08 D4 12
1589:A0 12 20 5B 4D AD 3E 03 E2
1591:AA CA D0 E6 AD 3D 03 A8 5D
1599:A5 FD 85 FB A5 FE 85 FC B7
15A1:A9 E0 8D 0F D4 A9 15 8D 80
15A9:12 D4 A9 03 8D 8B 4A 60 07
15B1:C0 00 F0 15 C0 01 D0 03 5A
15B9:4C 6F 4D 88 A2 C6 CA D0 AE
15C1:FD 88 D0 EC A2 C3 CA D0 80
15C9:FD 60 A9 41 85 FB A9 03 C6
15D1:85 FC A0 00 B1 FB C9 F1 15
15D9:F0 08 18 69 01 91 FB 4C AA
15E1:91 4D A9 E4 91 FB C8 C0 7A
15E9:08 D0 E9 60 A9 51 85 FB 29
15F1:A9 03 85 FC A0 00 B1 FB 96
15F9:C9 FF F0 08 18 69 01 91 A1
1601:FB 4C B3 4D A9 F2 91 FB C1
1609:C8 C0 08 D0 E9 A0 08 EE A8
1611:3C 03 AD 3C 03 8D 0F D4 D6
1619:88 D0 F4 60 AD 6C 03 18 9F
1621:69 29 85 FB AD 6D 03 85 6B
1629:FC AD 67 03 C9 00 D0 61 AD
1631:A0 00 B1 FB C9 00 F0 43 17
1639:A5 FB 38 E9 10 85 FB A0 0C
1641:00 B1 FB 18 69 19 CD 89 AF
1649:4A B0 03 4C 31 4E 20 AB 9A
1651:4C AD 89 4A 38 E9 17 91 0E

```

1659:FB A5 FB 18 69 10 85 FB 80
1661:A9 00 91 FB A5 FB 18 69 0B
1669:08 85 FB B1 FB C9 00 F0 8D
1671:03 4C 3C 4E A9 FF 91 FB 0B
1679:4C 3C 4E A5 FB 38 E9 10 A3
1681:85 FB A9 25 91 FB A5 FB BA
1689:18 69 10 85 FB A9 01 91 90
1691:FB 60 AD 6E 03 18 69 41 FC
1699:85 FB AD 6F 03 85 FC AD 0A
16A1:6A 03 C9 00 D0 61 A0 00 4A
16A9:B1 FB C9 00 F0 43 A5 FB C2
16B1:38 E9 10 85 FB A0 00 B1 E2
16B9:FB 38 E9 08 CD 89 4A 90 69
16C1:03 4C A7 4E AD 89 4A 18 9C
16C9:69 05 91 FB 20 03 4D A5 2B
16D1:FB 18 69 10 85 FB A9 00 9F
16D9:91 FB A5 FB 38 E9 08 85 41
16E1:FB B1 FB C9 00 F0 03 4C AA
16E9:B2 4E A9 01 91 FB 4C B2 10
16F1:4E A5 FB 38 E9 10 85 FB 48
16F9:A9 F7 91 FB A5 FB 18 69 A1
1701:10 85 FB A9 FF 91 FB 60 51
1709:AD 6C 03 85 FB AD 6D 03 56
1711:85 FC A0 00 AD 65 03 C9 28
1719:FF F0 06 C9 01 F0 20 D0 BD
1721:3B A9 00 91 FB A5 FB C9 A8
1729:59 F0 0A C6 FB A5 FB 8D E9
1731:6C 03 4C 07 4F A9 60 85 B7
1739:FB 8D 6C 03 4C 07 4F A0 44
1741:00 A9 00 91 FB A5 FB C9 2B
1749:60 F0 0A E6 FB A5 FB 8D 8F
1751:6C 03 4C 07 4F A9 59 85 C9
1759:FB 8D 6C 03 AD 70 03 C9 A5
1761:11 D0 05 A9 0C 8D 70 03 02
1769:EE 70 03 AD 70 03 91 FB 15
1771:AD 6E 03 85 FB AD 6F 03 43
1779:85 FC AD 64 03 C9 FF F0 D5
1781:06 C9 01 F0 22 D0 3D A0 C3
1789:00 A9 0D 91 FB A5 FB C9 15
1791:49 F0 0A C6 FB A5 FB 8D 4A
1799:6E 03 4C 6F 4F A9 50 85 87
17A1:FB 8D 6E 03 4C 6F 4F A0 8E
17A9:00 A9 0D 91 FB A5 FB C9 35
17B1:50 F0 0A E6 FB A5 FB 8D EF
17B9:6E 03 4C 6F 4F A9 49 85 99
17C1:FB 8D 6E 03 AD 71 03 C9 52
17C9:09 D0 05 A9 03 8D 71 03 20
17D1:EE 71 03 AD 71 03 91 FB C5
17D9:AD A8 03 F0 04 CE A8 03 28
17E1:60 A9 03 8D A8 03 A9 82 0B

```

17E9:85 FB A9 03 85 FC A0 00 A0
17F1:B1 FB C9 00 D0 1A A5 FB 67
17F9:38 E9 29 85 FB A9 06 91 60
1801:FB A5 FB 18 69 29 85 FB 90
1809:C0 07 F0 19 C8 4C 9B 4F 09
1811:A5 FB 38 E9 29 85 FB A9 B9
1819:00 91 FB A5 FB 18 69 29 C3
1821:85 FB 4C B3 4F A9 8A 85 93
1829:FB A9 03 85 FC A0 00 B1 96
1831:FB C9 00 D0 1B A5 FB 38 7E
1839:E9 41 85 FB A9 01 91 FB 8F
1841:A5 FB 18 69 41 85 FB C0 B5
1849:07 F0 04 C8 4C DA 4F 60 13
1851:A5 FB 38 E9 41 85 FB A9 BA
1859:0D 91 FB A5 FB 18 69 41 A2
1861:85 FB 4C F2 4F A9 82 85 B7
1869:FB A9 03 85 FC A9 72 85 B3
1871:FD A9 03 85 FE A0 00 B1 EF
1879:FB C9 FF F0 06 C9 01 F0 73
1881:26 D0 1B B1 FD 18 69 17 B1
1889:CD 89 4A B0 0A B1 FD 18 82
1891:69 01 91 FD 4C 49 50 20 11
1899:C2 50 A9 01 91 FB C0 07 89
18A1:F0 04 C8 4C 22 50 60 B1 ED
18A9:FD C9 1E 90 0A B1 FD 38 63
18B1:E9 01 91 FD 4C 49 50 A9 FA
18B9:FF 91 FB 4C 49 50 A9 8A FB
18C1:85 FB A9 03 85 FC A9 7A 07
18C9:85 FD A9 03 85 FE A0 00 0B
18D1:B1 FB C9 FF F0 06 C9 01 47
18D9:F0 1F D0 14 B1 FD C9 FB BA
18E1:B0 0A B1 FD 18 69 01 91 FC
18E9:FD 4C 9B 50 A9 01 91 FB 15
18F1:C0 07 F0 04 C8 4C 7B 50 61
18F9:60 B1 FD 38 E9 06 CD 89 96
1901:4A 90 0A B1 FD 38 E9 01 7E
1909:91 FD 4C 9B 50 20 D7 50 C9
1911:A9 FF 91 FB 4C 9B 50 A9 25
1919:10 8D 0C D4 A9 45 8D 08 0B
1921:D4 A9 80 8D 0B D4 A9 81 91
1929:8D 0B D4 60 A9 24 8D 05 83
1931:D4 A9 2F 8D 01 D4 A9 10 B5
1939:8D 04 D4 A9 11 8D 04 D4 04
1941:60 AD 00 DC A0 00 A2 00 27
1949:4A B0 01 88 4A B0 01 C8 55
1951:4A B0 01 CA 4A B0 01 E8 A1
1959:4A 8E 66 03 2A 29 01 8D D6
1961:68 03 AD 01 DC A0 00 A2 5A
1969:00 4A B0 01 88 4A B0 01 24
1971:C8 4A B0 01 CA 4A B0 01 A2

```

1979:E8 4A 8E 69 03 2A 29 01 2F
 1981:8D 6B 03 AD 68 03 C9 00 73
 1989:D0 11 AD 67 03 C9 00 D0 A4
 1991:12 AD AB 03 C9 FF F0 0B 19
 1999:EE AB 03 A9 01 8D 67 03 39
 19A1:4C 5C 51 20 9B 4C A9 00 9E
 19A9:8D AB 03 AD 68 03 8D 67 9A
 19B1:03 AD 6B 03 C9 00 D0 11 6F
 19B9:AD 6A 03 C9 00 D0 12 AD 6F
 19C1:AC 03 C9 FF F0 0B EE AC 82
 19C9:03 A9 01 8D 6A 03 4C 8A 63
 19D1:51 20 8B 4C A9 00 8D AC FF
 19D9:03 AD 6B 03 8D 6A 03 AD 60
 19E1:66 03 C9 00 F0 11 AD 65 CD
 19E9:03 C9 00 F0 12 AD AD 03 C4
 19F1:C9 01 F0 0B EE AD 03 A9 F5
 19F9:00 8D 65 03 4C B8 51 20 74
 1A01:72 4C A9 00 8D AD 03 AD 8D
 1A09:66 03 8D 65 03 AD 69 03 DD
 1A11:C9 00 F0 11 AD 64 03 C9 28
 1A19:00 F0 12 AD AE 03 C9 01 BC
 1A21:F0 0B EE AE 03 A9 00 8D A5
 1A29:64 03 4C E6 51 20 4E 4C 3C
 1A31:A9 00 8D AE 03 AD 69 03 7B
 1A39:8D 64 03 60 A9 82 85 FB 12
 1A41:A9 03 85 FC A9 00 8D 9A 8E
 1A49:03 A0 00 B1 FB C9 00 D0 1A
 1A51:03 EE 9A 03 C0 07 F0 04 4E
 1A59:C8 4C F6 51 A9 8A 85 FB 77
 1A61:A9 03 85 FC A9 00 8D 9B AF
 1A69:03 A0 00 B1 FB C9 00 D0 3A
 1A71:03 EE 9B 03 C0 07 F0 04 8E
 1A79:C8 4C 16 52 AD 9B 03 CD BC
 1A81:9A 03 90 6D F0 6A AD 89 C2
 1A89:4A C9 3E D0 03 4C 8B 53 DD
 1A91:38 E9 01 8D 89 4A A9 82 A0
 1A99:85 FB A9 03 85 FC A0 00 56
 1AA1:B1 FB C9 00 D0 15 A5 FB 09
 1AA9:38 E9 10 85 FB B1 FB 38 A5
 1AB1:E9 01 91 FB A5 FB 18 69 C3
 1AB9:10 85 FB C0 07 F0 04 C8 AF
 1AC1:4C 4B 52 A9 8A 85 FB A9 DF
 1AC9:03 85 FC A0 00 B1 FB C9 13
 1AD1:00 D0 15 A5 FB 38 E9 10 DB
 1AD9:85 FB B1 FB 38 E9 01 91 C2
 1AE1:FB A5 FB 18 69 10 85 FB 11
 1AE9:C0 07 F0 04 C8 4C 78 52 59
 1AF1:60 AD 89 4A C9 E8 D0 03 2E
 1AF9:4C 07 53 18 69 01 8D 89 F5
 1B01:4A A9 82 85 FB A9 03 85 81

```

1B09:FC A0 00 B1 FB C9 00 D0 D8
1B11:15 A5 FB 38 E9 10 85 FB D4
1B19:B1 FB 18 69 01 91 FB A5 AC
1B21:FB 18 69 10 85 FB C0 07 2E
1B29:F0 04 C8 4C B6 52 A9 8A 93
1B31:85 FB A9 03 85 FC A0 00 EF
1B39:B1 FB C9 00 D0 15 A5 FB A2
1B41:38 E9 10 85 FB B1 FB 18 1F
1B49:69 01 91 FB A5 FB 18 69 1D
1B51:10 85 FB C0 07 F0 04 C8 49
1B59:4C E3 52 60 A9 60 8D 0C F4
1B61:D4 8D 05 D4 A9 00 8D 5C 18
1B69:03 A9 20 8D 75 03 A9 FF 73
1B71:8D 85 03 A9 FA 8D 89 4A 36
1B79:A9 08 8D 15 D0 A9 08 8D 54
1B81:1D D0 8D 17 D0 A9 5C 8D 11
1B89:AF 03 A9 10 8D 04 D4 A9 5E
1B91:15 8D 04 D4 AD 3D 03 8D 79
1B99:0F D4 AD 75 03 8D 01 D4 BE
1BA1:20 97 4D 20 10 50 20 10 8B
1BA9:50 A0 40 20 5B 4D AD AF 55
1BB1:03 C9 01 F0 06 CE AF 03 D8
1BB9:4C 35 53 A9 00 8D 15 D0 99
1BC1:A9 FF 8D 3D 03 A0 FF 20 0D
1BC9:5B 4D AD 3D 03 D0 06 CE C0
1BD1:3D 03 4C 70 53 A9 00 8D C6
1BD9:17 D0 8D 1D D0 4C AB 40 A2
1BE1:A9 15 8D 13 D4 8D 05 D4 D0
1BE9:A9 0D 8D 4C 03 A9 E0 8D BC
1BF1:7D 03 A9 01 8D 8D 03 A9 3F
1BF9:1C 8D 89 4A A9 08 8D 15 15
1C01:D0 A9 08 8D 1D D0 8D 17 44
1C09:D0 A9 5C 8D AF 03 A9 10 65
1C11:8D 12 D4 A9 15 8D 12 D4 A1
1C19:AD 3D 03 8D 01 D4 AD 7D E4
1C21:03 8D 0F D4 20 75 4D 20 FE
1C29:69 50 20 69 50 A0 40 20 6A
1C31:5B 4D AD AF 03 C9 01 F0 4D
1C39:06 CE AF 03 4C B9 53 A9 E7
1C41:00 8D 15 D0 A9 FF 8D 3D 32
1C49:03 A0 FF 20 5B 4D AD 3D D5
1C51:03 D0 06 CE 3D 03 4C F4 70
1C59:53 A9 00 8D 17 D0 8D 1D B2
1C61:D0 4C AB 40 AC 3F 03 A9 A0
1C69:0A 8D 3F 03 CE 3F 03 88 24
1C71:D0 FA AD A0 03 C9 3C F0 39
1C79:06 EE A0 03 4C 3B 54 20 CC
1C81:EC 50 20 C7 4D 20 3D 4E 78
1C89:20 B3 4E A9 00 8D A0 03 9D
1C91:AD A1 03 C9 0F F0 06 EE 3D

```



```
1C99:A1 03 4C 53 54 20 10 50 B5
1CA1:20 69 50 A9 00 8D A1 03 65
1CA9:AD A2 03 C9 1E F0 06 EE 0E
1CB1:A2 03 4C 6B 54 20 75 4D 97
1CB9:20 97 4D A9 00 8D A2 03 AA
1CC1:AD A3 03 C9 FF F0 06 EE 75
1CC9:A3 03 4C 80 54 20 E7 51 6A
1CD1:A9 00 8D A3 03 AE 3F 03 1F
1CD9:A0 18 88 D0 FD CA D0 F8 3C
1CE1:4C 1D 54 FB 00 00 00 00 D1
```

The Idol of Orion

Richard Caravella

Delve into the darkest dungeon, battle the fiercest monsters, and probe the strangest mysteries in "The Idol of Orion," a graphics-based adventure game. Idol of Orion requires a steady hand and an even steadier nerve. Joystick needed.

You don't even need a reason to plunge into the Dungeons of Orion. The mere prospect of adventure is enough.

You enter the dungeon on level 1 with a bow, a quiver of silver arrows, a supply of blast powder, and all the courage you can muster to accomplish your quest. Each dungeon room takes you to the next round, and every other round takes you one level deeper into the cavernous pits.

Your goal is to reach level 20 (or beyond), and recover the ancient idol of Orion. Then you must return to level 1, where you can leave the dungeons behind and complete your journey.

Making Dank Dungeons

Type in the program. It's long, so make sure you use "The Automatic Proofreader," the typing-entry utility found in Appendix C. The Proofreader insures that you enter a correct copy the first time.

Save what you've typed to tape or disk, load it back into the computer, and type **RUN**.

The title screen appears, and after a slight delay, you'll see a list of symbols and what each represents in "The Idol of Orion." Take a few moments to study these.

When you're ready to begin your descent into the labyrinth, plug a joystick into port 2 and hit the fire button.

In the Dark

A moment or two to create the dungeon screen, and you're on your way. Push the joystick in the direction you wish to move.

In each screen you collect objects and avoid the evil creatures who want to put a deadly end to your trip. At the top of the screen your status is displayed and updated.

At the top of the screen you can see your current score (shown by the opened box), health status (H), number of arrows (↑↑), blast powder (B), level (L), and round of play (R) displayed.

You begin with a health status of 99—but every time you're attacked by a monster, you're injured, and that number declines. When it reaches 0, you become yet another lost adventurer. Obviously, the adventure ends.

There are objects scattered around the dungeon which you'll be interested in collecting. Arrows, treasure chests, and bottles of magic potions are yours for the taking. Each time you pick up a treasure chest, its contents are displayed on the screen and the value of the item is added to your score. It's important to collect as much treasure as possible, for without a 1000-point sacrifice, you won't be able to recover the idol of Orion.

Bottles of magic potion are also strewn about. Open a bottle (by moving your figure atop it) and the spell within is cast as it's displayed on the screen. The magic potions can transport you to another part of the dungeon or transport you to the next level, or even drop you off in an invisible dungeon, leaving you to find a way out of the maze of rooms in the dark. Other spells equip you with an endless supply of arrows (good for just the current room), a new supply of blast powder, or a potion to heal your wounds. One even protects you by stunning and immobilizing all the monsters on the screen.

Beware, though, because several potions can cause you injury.

Except for the blast-powder spell, all potions last until you advance to the next screen.

Blast powder can be used to blow your way through the normal brick walls of the dungeon. Just walk through a wall section—each time you pass through a section of brick, your supply of blast powder decreases by one. The only way to replenish your supply is to find a magic bottle filled with more blast powder. Keep in mind that blast powder doesn't work on solid or invisible bricks—only regular dungeon bricks.

Fight Back

Snakes, bats, giant spiders, and a demon dragon roam the dungeons waiting for an easy meal: you. Don't fear, for you're carrying a supply of arrows. To shoot an arrow, aim by mov-

ing the joystick in the direction of the target and press the fire button. Hit the creature and you'll add points to your score. Don't relax yet; another creature immediately replaces the one you've eliminated.

The deeper into the dungeon you travel, the more dangerous and faster the creatures become. Some leave unpleasant surprises in the dungeons to hinder your progress, so once you've collected all the arrows, treasures, and magic bottles on the screen, head for the trapdoor at the bottom of the screen. By going through the door, you travel down to the next section of the dungeon.

You'll find the idol of Orion somewhere beyond level 20. Once you recover the idol, you must fight your way back up to level 1 and escape the dungeon. Keep in mind that you must leave a 1000-point sacrifice or you can't take the idol.

The game ends when your health status falls to 0 (or below), or when you recover the idol and escape the dungeons. At that point, your score, bonus score, and total score are displayed, along with the highest score achieved. An option to play another game is offered.

Dare to descend into the dungeon!

The Idol of Orion

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

```
SQ 200 POKE53280,0:POKE53281,0:GOSUB1380
BE 210 POKE52,48:POKE56,48:CLR:POKE56334,PEEK(5633
4)AND254:POKE1,PEEK(1)AND251
EQ 220 FORI=0TO2047:POKE12288+I,PEEK(53248+I):NEXT
:POKE1,PEEK(1)OR4
JJ 230 POKE56334,PEEK(56334)OR1
KK 240 READA:IFA=-1THEN260
HF 250 FORI=0TO7:READB:POKE12288+A*8+I,B:NEXT:GOTO
240
QR 260 POKE53272,(PEEK(53272)AND240)OR12:GOSUB1430
AG 270 SM=1024:CM=55296:Q=54272:WE=54276:QW=54273:
W=54296
SF 280 FORJ=QTOW:NEXT:POKEW,15:POKE54277,64:POKE54
278,48
SC 290 DIMT$(35),V(35):FORI=0TO35:READT$(I):READV(
I):NEXT
FK 300 FORI=0TO9:READS$:S$(I)=S$:NEXT
CM 310 A$="{HOME}{UP}{39 SPACES}"
FG 320 FORF=1TO4:READMC:MC(F)=MC:READMP:MP(F)=MP:R
EADWP:WP(F)=WP:READPP:PP(F)=PP
JR 330 NEXT:FORF=1TO4:GOSUB1140:NEXT
```

```

XE 340 ID=0:BL=20:S=0:WD=99:L=1:RD=0:AR=5
HP 350 IFID=1ANDL=1THEN1260
FP 360 PRINT "{CLR}":FORI=1TO11:PRINT "{RVS}{PUR}{E}
      {R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}
      {R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}
      {R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}";
JE 370 PRINT "{RVS}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}
      {R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}
      {R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}{R}{E}
      {R}{E}{OFF}";:NEXT:X=10:Y=10
BA 380 AZ=1:BZ=-1:EZ=0:RD=RD+1:MF=0:IFID=1ANDRD=3T
      HENL=L-1:RD=1:GOTO400
GE 390 IFRD=3THENL=L+1:RD=1
BF 400 R=54272:N=INT(10*RND(1))+1:P=INT(10*RND(1))
      +1:BZ=BZ+1:IFBZ=25THEN450
FK 410 V=INT(1000*RND(1))+1064:IFV+N>2023THENV=V-N
      :GOTO410
GK 420 IFV+(40*P)>2023THENV=V-(V0*P):GOTO410
BQ 430 U=INT(3*RND(1))+28:POKEV,U:POKEV+R,INT(15*R
      ND(1))+1:FORJ=1TOP:FORK=1TON
PH 440 POKEV+AZ,32:V=V+AZ:NEXT:V=V+40-N:NEXTJ:GOTO
      400
JR 450 FORI=1064TO1944STEP40:POKEI,235:NEXT:FORI=1
      103TO1983STEP40:POKEI,243:NEXT
EK 460 FORI=1064TO1103:POKEI,160:NEXT:FORI=1984TO2
      023:POKEI,160:NEXT
BR 470 FORI=55336TO56256STEP40:POKEI,4:NEXT:FORI=5
      5375TO56255STEP40:POKEI,4:NEXT
SK 480 FORI=55336TO55375:POKEI,4:NEXT:FORI=56256TO
      56295:POKEI,4:NEXT
MQ 490 FORI=1429TO1439:POKEI,32:NEXT:POKE2003,36:P
      OKE56275,6
EG 500 IFL=>20THENPOKE1524,88:POKE55796,1
QG 510 JS=PEEK(56320)AND15:ZX=X:ZY=Y
BM 520 IF(JSAND8)=0THENX=X+1
HP 530 IF(JSAND4)=0THENX=X-1
HM 540 IF(JSAND2)=0THENY=Y+1
JF 550 IF(JSAND1)=0THENY=Y-1
CG 570 IFPEEK(SM+X+40*Y)=28THENGOSUB850:GOTO690
FH 580 IFPEEK(SM+X+40*Y)=29THENAR=AR+1:GOSUB870:GO
      TO690
KJ 590 IFPEEK(SM+X+40*Y)=30THEN880
KF 600 IFPEEK(SM+X+40*Y)=36THENGOSUB870:GOTO350
XJ 610 IFPEEK(SM+X+40*Y)=241ANDBL>0THENBL=BL-1:GOS
      UB870:GOTO690
DP 620 IFPEEK(SM+X+40*Y)=242ANDBL>0THENBL=BL-1:GOS
      UB870:GOTO690
HS 630 RN=RND(1)*1:IFMF=0ANDRN>.5ANDL<6THENF=1:GOS
      UB1160

```

```

ME 640 IFMF=0ANDRN>.7ANDL>2ANDL<=10THENF=2:GOSUB1160
PX 650 IFMF=0ANDRN>.8ANDL=>5THENF=3:GOSUB1160
FP 660 IFMF=0ANDRN>.5ANDL=>15THENF=4:GOSUB1160
KM 670 IFPEEK(SM+X+40*Y)=88THEN740
FM 680 IFPEEK(SM+X+40*Y)<>32THENX=ZX:Y=ZY
KS 690 POKESM+ZX+40*ZY,32:POKESM+X+40*Y,38:POKECM+X+40*Y,7
FF 700 PRINT "{HOME}{UP}{CYN}£"S;TAB(10)"H"WD;TAB(15)""]"AR;TAB(22)"B"BL;TAB(30)"L"LTAB(35)"R"RD
HP 710 IFWD<=0THEN1300
JK 720 IF(PEEK(56320)AND16)=0ANDAR>0THENGOSUB1050
AA 730 GOTO510
RQ 740 IFS>1000THEN770:PRINT$:GOSUB870
KP 750 PRINT$:FORI=1TO150:PRINT "{HOME}{UP}{GRN} YOU CAN'T RECOVER THE IDOL....":NEXT:PRINT$:
FR 760 FORI=1TO150:PRINT "{HOME}{UP}{GRN} UNTIL YOU HAVE 1000 POINTS":NEXT:PRINT$:GOTO690
PF 770 ID=1:S=S-1000:PRINT$:FORI=1TO150:POKEQW,59:POKEQ,95:POKEWE,17
AK 780 PRINT "{HOME}{UP}{GRN} CONGRATULATIONS, YOU'VE FOUND THE":NEXT:POKEWE,0
MR 790 PRINT$:FORI=1TO150:POKEQW,44:POKEQ,80:POKEWE,17
XJ 800 PRINT "{HOME}{UP}{GRN}{4 SPACES}IDOL OF ORION. YOU SACRIFICE":NEXT:POKEWE,0
KQ 810 PRINT$:FORI=1TO150:POKEQW,39:POKEQ,56:POKEWE,17
GA 820 PRINT "{HOME}{UP}{GRN} 1,000 POINTS.{2 SPACES}NOW, GET BACK UP TO":NEXT:POKEWE,0
HA 830 PRINT$:FORI=1TO150:POKEQW,59:POKEQ,95:POKEWE,17
XP 840 PRINT "{HOME}{UP}{GRN} LEVEL ONE AND ESCAPE {SPACE}THE DUNGEONS":NEXT:POKEWE,0:PRINT$:GOTO690
PJ 850 PRINT$:R=INT(20*RND(1)):FORI=1TO50:POKEQW,40:POKEQ,200:POKEWE,33
GG 860 PRINT "{HOME}{UP}{GRN}"TAB(8);T$(R)" = "V(R)"PTS.":NEXT:POKEWE,0:PRINT$:S=S+V(R):RETURN
JG 870 FORI=1TO5:POKEQW,52:POKEQ,22:POKEWE,33:NEXT:POKEWE,0:PRINT$:RETURN
RP 880 PRINT$:R=INT(10*RND(1)):FORI=1TO50:POKEQW,40:POKEQ,200:POKEWE,33
BA 890 PRINT "{HOME}{UP}{GRN}"TAB(12);S$(R):NEXT:POKEWE,0:PRINT$
XD 900 ONR+1GOTO910,920,930,950,970,980,990,1000,1020,1030

```

```

DA 910 GOTO690
AS 920 EZ=1:GOTO690
KJ 930 WD=WD+25:IFWD>=99THENWD=99
HC 940 GOTO690
QS 950 BL=BL+25:IFBL>=99THENBL=99
XF 960 GOTO690
AG 970 X=INT(37*RND(1))+2:Y=INT(20*RND(1))+2:GOTO6
    90
PD 980 GOTO350
AX 990 MF=1:GOTO690
QX 1000 WD=WD-25:IFWD<=0THEN1300
DB 1010 GOTO690
DP 1020 POKESM+X+40*Y,87:POKECM+X+40*Y,2:WD=WD-25:
    GOTO690
MG 1030 PRINT "{CLR}":FORI=1TO21:PRINT "
    {40 SHIFT-SPACE}";:NEXT
QP 1035 REM ** LISTING NOTE ** CHAR. IN LINE 1030
    {SPACE}IS SHIFTED SPACE **
RB 1040 GOTO380
BQ 1050 H=X:V=Y:PRINTA$:IFEZ=0THENAR=AR-1
GR 1060 IF(JSAND8)=0THENH=H+1
XR 1070 IF(JSAND4)=0THENH=H-1
AS 1080 IF(JSAND2)=0THENV=V+1
AP 1090 IF(JSAND1)=0THENV=V-1
DG 1100 FORI=1TO2:POKEQW,36:POKEQ,85:POKEWE,17:NEX
    T:POKEWE,0
FJ 1110 IFPEEK(SM+H+40*V)=MP(F)THENS=S+WP(F):GOSUB
    1140:RETURN
AB 1120 IFPEEK(SM+H+40*V)<>32THENRETURN
RJ 1130 POKESM+H+40*V,42:POKECM+H+40*V,10:POKESM+H
    +40*V,32:GOTO1060
AG 1140 M(F)=INT(37*RND(1))+2:N(F)=INT(20*RND(1))+
    3:RETURN
RG 1150 FORI=1TO20:POKEQW,51:POKEQ,97:POKEWE,33:NE
    XT:POKEWE,0:RETURN
QE 1160 POKESM+MM(F)+40*NN(F),PP(F)
XD 1170 POKESM+M(F)+40*N(F),MP(F):POKECM+M(F)+40*N
    (F),MC(F)
MQ 1180 MM(F)=M(F):NN(F)=N(F)
RP 1190 IFM(F)>XTHENM(F)=M(F)-1
ER 1200 IFM(F)<XTHENM(F)=M(F)+1
EE 1210 IFN(F)>YTHENN(F)=N(F)-1
AJ 1220 IFN(F)<YTHENN(F)=N(F)+1
KR 1230 FORI=1TO10:POKEQW,36:POKEQ,88:POKEWE,33:NE
    XT:POKEWE,0
AB 1240 IFM(F)=XANDN(F)=YTHENWD=WD-WP(F):GOTO 1245
XB 1242 GOTO 1250
FF 1245 GOSUB1140:GOSUB1150:PRINTA$:IFWD<=0THEN130
    0
ES 1250 RETURN

```

```

AH 1260 PRINT "{CLR}{5 DOWN}{CYN} CONGRATULATIONS,
        {SPACE}YOU'VE RECOVERED THE"
PX 1270 PRINT "{DOWN}IDOL OF ORION AND HAVE RETURNE
        D SAFELY"
XA 1280 PRINT "{DOWN}TO THE REAL WORLD.":FORI=1TO10
        00:NEXT:I=20
AJ 1300 FORI=1TO3:FORJ=1TO16:POKE53280,J:FORK=1TO5
        :POKEQW,52:POKEQ,228:POKEWE,33
KC 1310 NEXTK:POKEWE,0:NEXTJ:NEXTI:PRINT "{CLR}":PO
        KE53280,0:IFWD<=0THENWD=1
FK 1320 BS=(S/10)*L:TS=BS+S
KR 1330 IFTS>HSTHENHS=TS
ER 1340 PRINTTAB(12)" {CYN}{4 DOWN}FINAL SCORE:":S:
        PRINTTAB(12)" {DOWN}BONUS SCORE:":BS
RD 1350 PRINTTAB(12)" {DOWN}TOTAL SCORE:":TS:PRINTT
        AB(12)" {DOWN}HIGH SCORE:":HS
SA 1360 PRINTTAB(5)" {3 DOWN}{PUR}HIT FIRE BUTTON T
        O PLAY AGAIN"
XF 1370 WAIT56464,16,16:GOTO340
AQ 1380 PRINT "{CLR}{BLU}":FORI=1TO9:FORJ=1TO9:PRIN
        TTAB(J)" THE IDOL OF ORION{UP}"
CH 1390 R=(RND(1)*14)+1:POKE646,R:FORK=1TO50:NEXTK
        :NEXTJ:PRINT "{DOWN}":NEXTI:PRINT "{DOWN}"
FE 1400 FORI=1TO7:PRINTTAB(I)" {BLU} THE ULTIMATE A
        DVENTURE{UP}":FORJ=1TO50:NEXT:NEXT
PP 1425 RETURN
RB 1430 PRINT "{CLR}{2 DOWN}"TAB(9);"{YEL}{A}CCCCC
        CCCCCCCCCCCCCC[S]":B$="{YEL}{Q}CCCCCCCCCCC
        CCCCCCCC[W]"
FC 1440 PRINTTAB(9)" {YEL}B{2 SPACES}{CYN}X{BLU} =
        {SPACE}IDOL OF ORION {YEL}B":PRINTTAB(9);B
        $
XC 1450 PRINTTAB(9)" {YEL}B{3 SPACES}[7]&{GRN} = AD
        VENTURER{3 SPACES}{YEL}B":PRINTTAB(9);B$
RR 1460 PRINTTAB(9)" {YEL}B {BLU}[L]{PUR} = TREASURE
        CHEST {YEL}B":PRINTTAB(9);B$
AM 1470 PRINTTAB(9)" {YEL}B{2 SPACES}{RED}[6] = MA
        GIC BOTTLE{2 SPACES}{YEL}B":PRINTTAB(9);B$
FX 1480 PRINTTAB(9)" {YEL}B{5 SPACES}[1][7] = ARRO
        WS{5 SPACES}{YEL}B":PRINTTAB(9);B$
SM 1490 PRINTTAB(9)" {YEL}B{6 SPACES}{WHT}$ {BLU} =
        {SPACE}DOOR{6 SPACES}{YEL}B":PRINTTAB(9);B
        $
FM 1500 PRINTTAB(9)" {YEL}B {PUR}! {CYN}< {RED}#
        {GRN}[5] = MONSTERS {YEL}B":PRINTTAB(9);B
        $
HG 1510 PRINTTAB(9)" {YEL}B{2 SPACES}{CYN}R=ROUND
        {2 SPACES}{PUR}L=LEVEL{2 SPACES}{YEL}B":PR
        INTTAB(9);B$

```



```

XD 1520 PRINTTAB(9)"{YEL}B{2 SPACES}{WHT}H=HEALTH
      {SPACE}{7}B=BLAST{2 SPACES}{YEL}B"
GJ 1530 PRINTTAB(9)"{YEL}{Z}CCCCCCCCCCCCCCCCCCCC
      {X}"
PD 1535 PRINT"{DOWN}{GRN}{2 SPACES}PLUG JOYSTICK I
      N PORT 2 TO PLAY"
HM 1540 PRINTTAB(5)"{DOWN}{PUR}HIT FIRE BUTTON TO
      {SPACE}START PLAY":WAIT56464,16,16:RETURN
RQ 1550 DATA27,240,206,1,30,96,193,99,62,28,24,12,
      6,3,255,213,171,255
SC 1560 DATA29,6,15,102,246,102,102,96,96,30,60,24
      ,24,60,126,195,195,126
XQ 1570 DATA31,36,36,90,153,153,129,60,0,33,195,36
      ,60,36,255,36,66,129
EF 1580 DATA35,248,44,248,51,49,123,238,108,36,255
      ,129,129,129,129,129,129,255
KP 1590 DATA38,60,60,152,255,25,24,102,195,-1
XR 1600 DATAWORTHLESS TRASH,0,A TIN CUP,15,COPPER
      {SPACE}NUGGETS,25,SILVER COINS,50
PG 1605 DATASILVER BARS,55,OLD BONES,2,CROWN OF EM
      ERALDS,45,SHELLS,5
RB 1610 DATACOPPER COIN,5,SILVER CANDLESTICK,175,C
      RYSTAL BALL,50,GLOWING MOSS,5
DX 1615 DATASATIN PILLOW,15,ANCIENT TAPESTRY,80,MA
      GIC WAND,20,SUN DIAL,30
AX 1616 DATATURTLE SHELLS,10,OIL LAMP,15,CRYSTAL S
      HARD,55,DIAMOND DUST,45
EP 1620 DATAEMPTY TREASURE CHEST,0,ANCIENT SCROLL,
      35,SILVER ARROW HEADS,40
AC 1625 DATAIVORY COMB,60,IVORY SCULPTURE,70,EMBRO
      IDERED ROBE,95
FQ 1630 DATA AN EMERALD RING,75,A JWELED CROWN,10
      0,GOLD BARS,125,DIAMONDS,15
GP 1640 DATA MAGIC MUSHROOMS,10,SPUN GLASS VASE,25
      ,BRASS STATUE,20,WOODEN BEADS,5
SH 1650 DATAIRON CHAINS,5,GOLDEN GOBLET,150,WORHL
      ESS SPELL,ENDLESS ARROWS
RX 1660 DATAHEALING POTION,BLASTING SPELL,TRANSPOR
      T SPELL,DESCEND SPELL
XJ 1670 DATASTUN POTION,POISONED POTION,DEATH TRAP
      ,INVISIBLE DUNGEON
RB 1680 DATA13,27,5,32,2,31,10,241,14,33,15,160,5,
      35,25,96

```

1000000

1000000

1000000

1000000

1000000

1000000

1000000

1000000

1000000

1000000

Chapter 2

For School

Solarpix

Simon Edgeworth

Planets move in graceful orbits, comets hurtle toward the sun, and the moon turns in its eons-old patterns. This impressive series of astronomical simulations shows all this, and more.

Whether you're new to stargazing or a veteran astronomer, this easy-to-use demonstration of the planets and comets is fun to watch. "Solarpix" is a four-module simulation which includes:

- **Sun and Planets**, simulating the movement of the planets around the sun
- **Sun, Earth, and Moon**, showing how the phases of the moon are created by its orbit
- **Comets**, illustrating the path of Halley's and other comets
- **Facts**, listing the key properties of each planet, the sun, and the moon.

Type in Solarpix using "The Automatic Proofreader," which you'll find in Appendix C. Once you've entered and saved the program, type **RUN**.

Copernican Motions

In the first module, Sun and Planets, you can watch the orbits of up to five planets at once. The sun is stationary at the center of the screen. Select the innermost and outermost planets you'd like to see, and a year from 1 to 2000. Once the screen is set up, press G to put the planets in motion, the plus key (+) to increase the speed, the minus key (−) to reduce the speed, and S to stop. Elapsed time is displayed in years on the right side of the screen. When you're through, exit the module (and all others) by pressing the Z key.

Lunar Lore

The second module, Sun, Earth, and Moon, shows how the moon's orbit around the earth causes its different phases. The earth is shown orbiting the sun, while the moon orbits the earth. The distance between the earth and the moon has been magnified by 30 times to make it visible. You'll see the current appearance of the moon in the top right corner of the screen. Elapsed time in days is displayed to the right. You can control the motion by using the G, S, +, and - keys as you did in the first module.

The moon takes 27.3 days to orbit the earth. However, it's 29.6 days from one full moon to the next. If you watch the display carefully, you can see why this happens—the moon actually has to complete slightly more than one orbit around the earth to reach the same position relative to the sun.

Because the moon's orbit is slightly tilted, the moon, earth, and sun are normally never exactly aligned. On the rare occasions when they are, an eclipse occurs. If this happens at full moon, we see a lunar eclipse. If it happens at new moon, we see a solar eclipse.

Dirty Ice

The third module, Comets, shows the orbits of these dirty snowballs. Planets and comets move in elliptical orbits, which look like flattened circles. The extent to which the circle is flattened is called its eccentricity, and ranges from 0 (circular) up to almost 1. The eccentricities of the major planets vary widely—Venus's is almost 0, for instance, while Pluto's is approximately 0.25. A moving body—say, a comet—with an eccentricity of 1 or more would leave the solar system.

Enter a value between 0 and .96 and press RETURN to set a comet (real or imaginary) into motion. Notice that the speed of a comet varies as it moves around its orbit. It moves slowest when it's farthest from the sun, and fastest when it's closest to the sun. This is true of planets too, but it's far easier to see in comets' orbits.

Try entering these eccentricities—they represent actual comets.

Faye	.56
Finlay	.70
Enke	.85.
Halley's	.96R

(The R in Halley's stands for reverse, because the comet orbits in the opposite direction of the planets.)

To exit this module, you have to hit RETURN after you've pressed the Z key.

Facts and Figures

The last module in Solarpix provides some interesting facts about the solar system. From the menu you can select a planet, the sun, or the moon, then view all the facts about it; or you can select a characteristic, such as mass or temperature, and compare the values for the nine planets plus the sun and the moon. Because the distances from the planets to the sun are so hard to visualize, another item, "Drive Time," has been added. It shows the amount of time it would take to drive from a given planet to the sun while traveling at 55 miles per hour. For the moon, the drive time is given for travel from the moon to the earth.

Masses are given in units equivalent to the weight of the earth because the values in tons are too large to comprehend (the earth weighs about 6,588,000,000,000,000,000 tons). Densities are compared with water (notice that Saturn is *less* dense than water). Temperatures are average surface temperatures. Moons are listed in order of distance from their planet, starting with the closest. Moons listed as X have not yet been named.

For the amateur astronomer, here are a few technical details. In Solarpix, all orbits are assumed to be on the same plane. For Pluto, however, true elliptical motion is simulated. The planet screen is oriented with Pluto's aphelion on the right, and the positions of the planets are calculated from their actual positions on October 27, 1984. All the graphics screens show views from ecliptic north.

Solarpix

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

```

XA 10 POKE56,132:CLR
FG 20 POKE792,193:POKE648,132
RC 30 POKE56576,(PEEK(56576)AND252)OR1
XR 40 POKE53280,3:POKE53281,3:DIMS(255),C(255),M$(
9,17)
PA 50 BP=34808:V=53248:LC=49152:EY=1984.82
CB 60 PP=2*↑:P$(0)="AC":P$(1)="DE":SA=1.1
RA 70 PRINT" {CLR}{8 DOWN}{BLU}"SPC(14)"PLEASE WAIT
"
CB 80 SP$=CHR$(32):X$=SP$+"Z=EXIT"
KC 90 RE$=CHR$(18):RX$=CHR$(146)
XQ 100 T$(1)=RE$+"SUN & PLANETS"+RX$
FF 110 T$(2)=RE$+"SUN,EARTH & MOON"+RX$
RD 120 T$(3)=RE$+"COMETS"+RX$
PQ 130 T$(4)=RE$+"FACTS"+RX$
SH 140 FORJ=49152TO49240:READK:POKEJ,K:NEXT
AF 150 FORJ=34816TO35199:READK:POKEJ,K:NEXT
KK 160 FORJ=35520TO35775:POKEJ,0:NEXT
PX 170 FORL=1TO10:READJ,K:POKE35584+J,K:NEXT
SJ 180 FORJ=0TO7:READK:POKEBP+J,K:NEXT
BH 190 FORJ=0TO7:READH(J):NEXT
BC 200 FORJ=0TO10:READF$(J),U$(J)
HQ 210 L=0:H=10:IFJ>4THENL=1:H=9
EB 220 FORK=LTOH:READZ(K,J):NEXT:NEXT
DQ 230 FORJ=1TO9:D(J)=Z(J,5)
BB 240 P(J)=Z(J,6):NEXT
XE 250 FORJ=0TO10:READN$(J)
DG 260 IFJ<4ORJ=10THEN280
HF 270 FORK=1TOZ(J,10):READM$(J,K):NEXT
CE 280 NEXT:FORJ=1TO9:READR:R(J)=R/360:NEXT
SJ 290 FORN=0TO255:M=N*2*↑/256
RJ 300 S(N)=SIN(M):C(N)=COS(M):NEXT
RR 310 FORJ=32TO36:K=74-J
BA 320 POKE251,J:POKE252,2:POKE253,K
AK 330 POKE254,2:SYS49175:NEXT
MF 340 FORJ=0TO7:POKEV+39+J,H(J):NEXT
BH 350 POKEV+23,0:POKEV+29,0:POKEV+16,128
QQ 360 GOSUB550:POKEV+39,7:POKEV+40,3
HK 370 POKEV+32,3:POKEV+33,3:POKE646,6
EG 380 PRINT" {CLR}":SYSLC,4,14
KQ 390 PRINT"SOLARPIX":PRINT:PRINT
KC 400 FORJ=1TO4:PRINTTAB(8)J:T$(J):PRINT
GD 410 NEXT:PRINTTAB(9)"Q QUIT"
DR 420 GOSUB470:ONKGOTO630,1010,1280,1460
MB 430 IFK$<>"Q"THEN420
MM 440 POKE56576,PEEK(56576)OR3
MG 450 POKE648,4:POKE792,71:GOSUB550

```



```

KE 460 PRINT "{CLR}":POKE56,160:CLR:END
GE 470 POKE198,0
RR 480 GETK$:IFK$=" "THEN480
HF 490 K=VAL(K$):RETURN
BM 500 SYSLC,18,31:PRINT "G=GO"
BC 510 PRINTTAB(31)"S=STOP"
PH 520 PRINTTAB(31)" +=FASTER"
ME 530 PRINTTAB(31)" -=SLOWER"
RS 540 PRINTTAB(30)X$:RETURN
PX 550 POKEV+21,0:FORJ=0TO15
JK 560 POKEV+J,0:NEXT:RETURN
DH 570 SYSLC,23,30:PRINTX$
GA 580 SYSLC,2,0:RETURN
FR 590 GETK$:IFK$="S "THENFL=1
PH 600 IFK$="G"THENFL=0
QJ 610 IFK$="-"THENNM=M/2
XR 620 RETURN
JA 630 POKEV+32,11:POKEV+33,0:POKE646,7
KR 640 GOSUB550:PRINT "{CLR}"T$(1):GOSUB570
HG 650 PRINT "[7]PLEASE SELECT PLANETS (MAX5)"
FF 660 PRINT:FORJ=1TO9:PRINTJ,N$(J):NEXT
DG 670 PRINT:PRINT "INNER PLANET?";
CX 680 GOSUB470:IFK$="Z"THEN360
QF 690 L=K:IFL<1THEN680
EG 700 PRINTN$(L):PRINT "OUTER PLANET?";
RK 710 GOSUB470:IFK$="Z"THEN360
DE 720 H=K:IFH<LORH>9ORH-L>4THEN710
FR 730 PRINTN$(H):PRINT:X=1
MH 740 FORJ=1TOH-L+1:X=X+2↑J:NEXT
MK 750 FORJ=LTOH:F(J)=D(J)*90/D(H)
RR 760 E(J)=F(J)*SA:NEXT:M=P(L)/50:ML=M*9
SR 770 INPUT"STARTING YEAR (1-2000)":K$
FC 780 SY=VAL(K$):IFSY<1ORSY>2000THEN630
DJ 790 PRINT "{CLR} {YEL}"T$(1):K=1
AF 800 FORJ=LTOH:POKE646,H(K)
RF 810 PRINTTAB(31)N$(J):K=K+1:NEXT
DJ 820 POKE646,14:GOSUB500:SYSLC,7,31
AX 830 PRINT "EARTH":PRINTTAB(31)"YEAR"
PE 840 E=.25:T=SY-EY:FL=1
RS 850 POKEV,130:POKEV+1,148:POKEV+21,X
BH 860 FORJ=LTOH:A=T/P(J)+R(J)
BD 870 A=INT((A-INT(A))*256):IFJ=9?THEN890
AD 880 X(J)=C(A):Y(J)=-S(A):GOTO910
MB 890 Z=1+E*C(A):X(J)=E+(E+C(A))/Z
FK 900 Y(J)=(E*E-1)*S(A)/Z
XE 910 X(J)=INT(131+X(J)*E(J))
MQ 920 Y(J)=INT(149+Y(J)*F(J)):NEXT
JC 930 K=1:FORJ=LTOH:POKEV+2*K,X(J)
ER 940 POKEV+2*K+1,Y(J):K=K+1:NEXT
JB 950 T$=STR$(INT((T+EY)*10)/10)

```

```

BA 960 SYSLC,9,30:PRINTT$SP$SP$
CP 970 GOSUB590:IFK$="Z"THEN630
QJ 980 IFFLTHEN970
HK 990 IFK$="+"ANDM<MLTHENM=M*2
RJ 1000 T=T+M:GOTO860
BQ 1010 POKEV+32,14:POKEV+33,0:POKE646,14
SM 1020 PRINT"{CLR}"T$(2):PRINTTAB(31)"MOON"
AQ 1030 PRINTTAB(31)"PHASE":GOSUB500
BR 1040 SYSLC,24,1:PRINT"(EARTH-MOON DISTANCE MAGN
    IFIED BY 30)";
CP 1050 SYSLC,7,31:PRINT"EARTH"
EQ 1060 PRINTTAB(31)"DAY"
GP 1070 E=84:D=E*SA:F=D*30/390:G=F/SA
BF 1080 M=0.3/365.25:T=0:H=13.3685
EB 1090 POKEV+39,14:POKEV+40,12
SF 1100 POKEV+12,141:POKEV+13,147
HS 1110 POKEV+14,24:POKEV+15,80
PS 1120 POKEV+21,195:FL=1
DG 1130 A=T-INT(T):B=T*H:B=B-INT(B)
CQ 1140 C=INT(12*(B-A)+.5-7)
RX 1150 IFC<0THENC=C+12:GOTO1150
DG 1160 A=A*PP:B=B*PP
XB 1170 X=INT(142.5+COS(A)*D)
RH 1180 Y=INT(148.5-SIN(A)*E)
KJ 1190 W=INT(X+1+COS(B)*F)
GA 1200 Z=INT(Y+1-SIN(B)*G)
MD 1210 POKEV,X:POKEV+1,Y:POKEV+2,W
EP 1220 POKEV+3,Z:POKEBP+7,32+C
JR 1230 SYSLC,9,30:PRINTINT(T*365.25)
GR 1240 GOSUB590:IFK$="Z"THEN360
QX 1250 IFFLTHEN1240
PB 1260 IFK$="+"ANDM<.006THENM=M*2
GJ 1270 T=T+M:GOTO1130
GD 1280 POKEV+32,5:POKEV+33,0:POKE646,5
CR 1290 GOSUB550:PRINT"{CLR}"RE$T$(3):GOSUB570
SB 1300 INPUT"ECCENTRICITY(0-0.96)";K$
HJ 1310 E=VAL(K$):IFK$="Z"THEN360
KB 1320 IFE<0ORE>.96ORLEN(K$)=0THEN1280
BE 1330 FL=1:IFRIGHT$(K$,1)="R"THENFL=-1
BS 1340 PRINT"{CLR}"RE$T$(3):GOSUB570
FF 1350 PRINT"ECCENTRICITY="E:SYSLC,2,27
JS 1360 IFETHENPRINT"ACCELERATING"
HC 1370 M=0.01:T=0:F=65*SA:G=65*FL
QK 1380 POKEV,106:POKEV+1,148:POKEV+21,3
MG 1390 A=(T-INT(T))*PP:Z=1+E*COS(A)
RC 1400 X=E+(E+COS(A))/Z:Y=(E*E-1)*SIN(A)/Z
AG 1410 X=INT(107+X*F):Y=INT(149+Y*G)
FD 1420 J=INT(A/1):POKEV+2,X:POKEV+3,Y
JQ 1430 IFE>0.1THENSYSLC,2,27:PRINTP$(J)
FX 1440 GETK$:IFK$="Z"THEN1280

```

```

XQ 1450 T=T+M:GOTO1390
XA 1460 POKEV+32,3:POKEV+33,6:POKE646,3
CD 1470 PRINT "{CLR}"RE$T$(4)" MENU":SYSLC,4,0
MS 1480 GOSUB570:SYSLC,4,0
JK 1490 PRINT" S SUN":PRINT" M MOON"
SJ 1500 FORJ=1TO9:PRINTJ;N$(J):NEXT
FS 1510 SYSLC,4,0:FORJ=0TO10
BC 1520 PRINTTAB(20)CHR$(J+65)" F$(J):NEXT
CR 1530 GOSUB470:IFK$="Z"THEN360
QD 1540 IFK$="S"THENK=0:GOTO1590
GE 1550 IFK$="M"THENK=10:GOTO1590
EM 1560 IFK>0ANDK<10THEN1590
RS 1570 K=ASC(K$)-65:IFK<0ORK>10THEN1530
RG 1580 GOTO1720
FM 1590 POKEV+32,3:POKEV+33,11:POKE646,3
QG 1600 PRINT "{CLR}"RE$N$(K)RX$;
AG 1610 H=4:IFK=0ORK=10THENPRINT:GOTO1630
SB 1620 H=10:PRINT" PLANET #"K"FROM SUN"
PE 1630 PRINT:FORJ=0TOH
HJ 1640 PRINTF$(J)TAB(18)Z(K,J)TAB(28)U$(J)
GR 1650 NEXT:IFK<10THEN1680
FS 1660 PRINT"DIST. FROM EARTH"TAB(19)"243000 MILES
"
DH 1670 PRINTF$(9)TAB(19)"6 MONTHS"
SE 1680 PRINT:IFK<4ORK=10THEN1770
CQ 1690 FORJ=1TOZ(K,10):C=INT((J-1)/7)
PE 1700 L=13+J-C*7:SYSLC,L,C*13
BD 1710 PRINTM$(K,J):NEXT:GOTO1770
MR 1720 POKEV+32,3:POKEV+33,11:POKE646,3
BK 1730 PRINT "{CLR}"RE$F$(K)RX$TAB(19)U$(K)
CP 1740 L=0:H=10:IFK>4THENL=1:H=9
AR 1750 SYSLC,5,0:FORJ=LTOH
KD 1760 PRINTN$(J)TAB(18)Z(J,K):NEXT
RQ 1770 GOSUB570:GOSUB470:GOTO1460
XJ 1780 DATA 32,13,192,132,251,32,13
HG 1790 DATA 192,166,251,76,240,255,32
EM 1800 DATA 253,174,32,158,173,32,170
AH 1810 DATA 177,96,162,6,6,251,38
BA 1820 DATA 252,6,253,38,254,202,208
MH 1830 DATA 245,169,21,133,2,160,2
CC 1840 DATA 177,251,162,8,106,38,255
ME 1850 DATA 202,208,250,192,1,240,4
QX 1860 DATA 152,73,2,168,165,255,145
PK 1870 DATA 253,192,1,240,4,152,73
PH 1880 DATA 2,168,136,16,223,162,3
QR 1890 DATA 230,251,230,253,202,208,249
KS 1900 DATA 198,2,208,208,96
KS 1910 DATA 0,15,0,0,0,192,0,0,48
JG 1920 DATA 0,0,56,0,0,28,0,0,28
FP 1930 DATA 0,0,14,0,0,14,0,0,7

```

```

GC 1940 DATA 0,0,7,0,0,7,0,0,7
GF 1950 DATA 0,0,7,0,0,14,0,0,14
GR 1960 DATA 0,0,28,0,0,28,0,0,56
JX 1970 DATA 0,0,48,0,0,192,0,15,0,0
QM 1980 DATA 0,15,0,0,7,192,0,3,240
GR 1990 DATA 0,1,248,0,1,252,0,1,252
ER 2000 DATA 0,0,254,0,0,254,0,0,255
BJ 2010 DATA 0,0,255,0,0,255,0,0,255
MR 2020 DATA 0,0,255,0,0,254,0,0,254
SG 2030 DATA 0,1,252,0,1,252,0,1,248
KH 2040 DATA 0,3,240,0,7,192,0,15,0,0
HH 2050 DATA 0,15,0,0,15,192,0,15,240
RQ 2060 DATA 0,15,248,0,15,252,0,15,252
GF 2070 DATA 0,15,254,0,15,254,0,15,255
QB 2080 DATA 0,15,255,0,15,255,0,15,255
SE 2090 DATA 0,15,255,0,15,254,0,15,254
EF 2100 DATA 0,15,252,0,15,252,0,15,248
SJ 2110 DATA 0,15,240,0,15,192,0,15,0,0
BB 2120 DATA 0,15,0,0,31,192,0,63,240
SX 2130 DATA 0,127,248,0,127,252,0,127,252
MX 2140 DATA 0,255,254,0,255,254,0,255,255
DD 2150 DATA 0,255,255,0,255,255,0,255,255
QX 2160 DATA 0,255,255,0,255,254,0,255,254
FK 2170 DATA 0,127,252,0,127,252,0,127,248
XD 2180 DATA 0,63,240,0,31,192,0,15,0,0
FR 2190 DATA 0,31,0,0,255,192,3,255,240
HA 2200 DATA 3,255,248,7,255,252,7,255,252
HP 2210 DATA 15,255,254,15,255,254,31,255,255
RP 2220 DATA 31,255,255,31,255,255,31,255,255
HS 2230 DATA 31,255,255,15,255,254,15,255,254
XE 2240 DATA 7,255,252,7,255,252,3,255,248
CA 2250 DATA 3,255,240,0,255,192,0,31,0,0
KQ 2260 DATA 0,255,0,3,255,192,15,255,240
PB 2270 DATA 31,255,248,63,255,252,63,255,252
XR 2280 DATA 127,255,254,127,255,254,255,255,255
AF 2290 DATA 255,255,255,255,255,255,255,255,255
CB 2300 DATA 255,255,255,127,255,254,127,255,254
DG 2310 DATA 63,255,252,63,255,252,31,255,248
RR 2320 DATA 15,255,240,3,255,192,0,255,0,0
CK 2330 DATA 0,112,3,248,6,248,9,248,12,112
BR 2340 DATA 64,192,67,192
BR 2350 DATA 128,224,131,224,134,224
SE 2360 DATA 46,45,45,45,45,45,44,37
FR 2370 DATA 7,3,4,14,8,6,7,12
AK 2380 DATA DIAMETER,MILES
FH 2390 DATA 864000,3100,7700,7926
RC 2400 DATA 4200,88000,71000,32000
GA 2410 DATA 31000,1500,2160
GJ 2420 DATA MASS,(EARTH=1)
MK 2430 DATA 332830,0.055,0.815,1

```

HF 2440 DATA 0.107,318,95.2,14.6
 CM 2450 DATA 17.2,0.01,0.012
 SA 2460 DATA DENSITY,(WATER=1)
 SD 2470 DATA 1.4,5.4,5.2,5.5,3.9,1.3
 BD 2480 DATA 0.7,1.2,1.7,1.4,3.4
 EE 2490 DATA ROTATION PERIOD,EARTHDAYS
 QS 2500 DATA 25,58.9,243,1,1.03,0.404
 MJ 2510 DATA 0.444,0.72,0.77,6.39,27.3
 KK 2520 DATA TEMPERATURE,FAHRENHEIT
 AJ 2530 DATA 10000,620,900,72,-10,-240
 BM 2540 DATA -300,-340,-370,-400,-10
 FE 2550 DATA DISTANCE FROM SUN,MILL.MILES
 SP 2560 DATA 36,67.2,92.9,142,483
 BK 2570 DATA 887,1783,2795,3670
 GK 2580 DATA LENGTH OF YEAR,EARTHYEARS
 EX 2590 DATA 0.241,0.615,1,1.88,11.86
 PR 2600 DATA 29.46,84,164,248
 BR 2610 DATA ORBIT SPEED,MPH
 XJ 2620 DATA 107000,78000,66000,54000
 QD 2630 DATA 29000,22000,15000,12000,10000
 XG 2640 DATA AXIS TILT,DEGREES
 AJ 2650 DATA 0,179,23.5,25,3.1
 FD 2660 DATA 26.7,97.9,28.8,0
 XQ 2670 DATA DRIVETIME(55MPH),YEARS
 JA 2680 DATA 75,139,193,295,1002
 MM 2690 DATA 1840,3698,5797,7612
 RS 2700 DATA MOONS,-
 CE 2710 DATA 0,0,1,2,16,17,15,3,1
 XC 2720 DATA SUN,MERCURY,VENUS,EARTH,MARS,PHOBOS,D
 EIMOS
 DH 2730 DATA JUPITER,METIS,ADRASTREA,AMALTHEA,THEB
 E,IO,EUROPA,GANYMEDE,CALLISTO
 JB 2740 DATA LEDA,HIMALIA,LYSITHEA,ELARA,ANANKE,CA
 RME,PASIPHAE,SINOPE
 FR 2750 DATA SATURN,ATLAS,X,X,JANUS,EPIMETHEUS,MIM
 AS,ENCELADUS,TETHYS,TELESTO
 MA 2760 DATA CALYPSO,DIONE,X,RHEA,TITAN,HYPERION,I
 APETUS,PHOEBE
 SF 2770 DATA URANUS,MIRANDA,ARIEL,UMBRIEL,TITANIA,
 OBERON,X,X,X,X,X,X,X,X,X
 JK 2780 DATA NEPTUNE,TRITON,NEREID,X,PLUTO,CHARON,
 MOON
 EH 2790 DATA 198,255,351,285,249,180,204,227,173

Video Jigsaw

Fred Karg

This innovative program creates an onscreen jigsaw puzzle—your choice of several levels of complexity—from Doodle or Koala high-resolution picture files. Designed for everyone in the family, “Video Jigsaw” requires a disk drive and joystick.

Jigsaw puzzles have always provided a lot of fun for people of all ages. If you have a Commodore 64 or 128, a disk drive, and a joystick, you can try to solve a different kind of jigsaw puzzle.

“Video Jigsaw” puts high-resolution color puzzles on the screen from any picture created with either the popular *Doodle* or *KoalaPainter* (packaged with *KoalaPad*) drawing packages. After choosing a picture file, select a difficulty level and try to solve the puzzle by moving pieces on the screen with a joystick.

Start

After typing in the program, be sure to save a copy to disk. To start, load the program and type **RUN**. You’re first asked to insert a disk containing *Doodle* or *Koala* picture files. Put the disk in the drive and press any key. The program then locates any *Doodle* or *Koala* files in the directory and lists them. As instructed, use the cursor-up/-down key to highlight the name of the file you want to load; then press RETURN.

After the hi-res picture is loaded into memory, choose the number of puzzle pieces. First pick the width (2–12), then the length (2–20). This feature makes the puzzle adaptable to any difficulty level—from super easy to almost impossible. The larger the numbers in either range, the easier the puzzle is to solve. A choice of 12×20 , for example, gives 4 very large puzzle pieces, suitable for play by preschoolers. A choice of 2×2 gives 240 pieces and a headache for even the most determined puzzle enthusiast. All combinations between these extremes are possible. After choosing the width and length, you’re reminded to plug a joystick into port 2 and press any key to begin puzzle construction.

When the picture appears, you'll see a flashing arrow pointing to the upper left jigsaw piece. Select the piece you want to relocate by moving the arrow to it with the joystick and pressing RETURN. Then move the arrow to the area where you want to place the piece and press RETURN. If you need extra help, press f1, and you'll see the completed picture. Press f1 again to return to the puzzle. When the puzzle is solved correctly, you'll be asked if you wish to load another picture file and play again. Press Y or N, as prompted.

Any *Doodle* or *Koala* picture can be turned into a jigsaw, but those with a lot of detail are best. *Doodle's* famous "Middle Earth" picture makes an exceptional puzzle. Avoid pictures with large, featureless areas. One patch of clear blue sky looks just like another, but the program requires you to put the piece in exactly the right place. Like a regular jigsaw, it helps to do the border first.

Video Jigsaw has all the challenge, fun, and frustration of a real jigsaw puzzle, with none of the drawbacks. There are no pieces to scatter or pick up—and never a missing piece. Puzzles are always random and as limitless as your video art collection.

Video Jigsaw

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

```

SG 10 IF AA=1THEN90
SX 20 POKE53280,0:POKE53281,0:PRINT"{CLR}[1]"TAB(1
3)"{RVS} VIDEO JIGSAW "
QA 30 GOSUB1230:AA=1:PRINT"{CLR}{2 DOWN}{7 SPACES}
LOADING "N$(A)
MG 40 PRINT"{3 DOWN} PLEASE WAIT....{BLK}"
CR 50 IFLEFT$(N$(A),1)="D"THENZZ=1
DK 60 IFLEFT$(N$(A),1)="[1]"THENZZ=2
AD 70 POKE49990,ZZ
CB 80 LOAD N$(A),8,1
XQ 90 POKE55,0:POKE56,92:CLR:ZZ=PEEK(49990):DIM N$
(19)
BF 100 FORA=828TO873:READ D:POKEA,D:NEXT
DX 110 FORA=49879TO49973:READ D:POKEA,D:NEXT
SS 120 PRINT"{CLR}{BLU}":POKE781,8:POKE782,5:POKE7
83,0:SYS65520
GC 130 PRINT"{RVS}CHOOSE SIZE OF PUZZLE PIECES:"
DM 140 PRINT"{2 RIGHT}{DOWN}(SMALLER NUMBERS GIVE
{SPACE}HARDER PUZZLES)"
SK 150 POKE781,14:POKE782,2:POKE783,0:SYS65520
SA 160 PRINT"LENGTH ({RVS}FROM 2 TO 12{OFF})";:INP
UT L$

```

```
CC 170 L=VAL(L$):IF L<20 THEN 150
GH 180 POKE 781,18:POKE 782,2:POKE 783,0:SYS 65520
DG 190 PRINT "WIDTH{2 SPACES}({RVS}FROM 2 TO 20
      {OFF})";:INPUT W$
BM 200 W=VAL(W$):IF W<20 THEN 180
BH 210 PRINT "{CLR}{3 DOWN}{3 SPACES}ONCE PUZZLE AP
      PEARS YOU MAY SEE"
XB 220 PRINT "{DOWN}{3 SPACES}PICTURE AS IT SHOULD
      {SPACE}LOOK WHEN"
RA 230 PRINT "{DOWN}{3 SPACES}FINISHED BY PRESSING
      {SPACE}{RVS}F1"
MC 240 PRINT "{YEL}{2 DOWN}{3 SPACES}PLUG JOYSTICK
      {SPACE}INTO PORT 2"
FX 250 PRINT "{GRN}{2 DOWN}{3 SPACES}PUZZLE CONSTRU
      CTION WILL BLANK"
GP 260 PRINT "{3 SPACES}SCREEN FOR 1 TO 2 MINUTES
      {2 DOWN}"
MQ 270 PRINT "{2 SPACES}PRESS ANY KEY TO START CONS
      TRUCTION"
GR 280 GET K$:IF K$="" THEN PRINT "{UP}";:POKE 646,PEE
      K(56325):GOTO 270
FD 290 POKE 53265,PEEK(53265)AND 239
MS 300 IF ZZ=1 THEN GOSUB 1530
XB 310 IF ZZ=2 THEN GOSUB 1570:GOSUB 1530:POKE 53281,PEE
      K(34576)
PH 320 FOR A=32704 TO 32767:READ D:POKE A,D:NEXT
SK 330 V=53248:SID=54272:XX=24+(W*4-4):YY=50+(L*4
      )-4
MK 340 POKE SID+5,9:POKE SID+6,9
SG 350 MAX=XX+(INT(40/W)-1)*8*W:MYX=YY+(INT(25/L)-
      1)*8*L
EP 360 DIM S(1000):DIM C(1000)
RF 370 DEF FN HI(M)=INT(M/256):DEF FN LO(M)=M-256*
      FN HI(M)
FK 380 FOR N=0 TO INT(25/L)-1
XC 390 FOR N1=0 TO INT(40/W)-1
HC 400 B=B+1
RM 410 S(B)=24576+N1*(W*8)+N*(L*320)
CF 420 C(B)=55296+N1*(W)+N*(40*L)
GH 430 A$=A$+CHR$(B)
CF 440 NEXT N1:NEXT N
DE 450 D$=A$:B$=A$
SQ 460 FOR I=1 TO B:A=I+INT((B+1-I)*RND(0))
PE 470 D$=MID$(D$,A,1)+LEFT$(D$,A-1)+RIGHT$(D$,B-A
      )
PA 480 NEXT
DD 490 Z=INT(LEN(D$)/2)*2-2
SC 500 FOR I=1 TO Z STEP 2:X1=ASC(MID$(D$,I,1)):X2=AS
      C(MID$(D$,I+1,1))
AH 510 GOSUB 1020:GOSUB 760
```



```

CD 520 POKE53280,I: NEXT:D$=B$:B$=A$
MK 530 B=1
PF 540 POKEV+28,PEEK(V+28)OR1:POKEV+39,7:POKEV+37,
    2:POKEV+38,5:POKEV,X
MF 550 POKEV+1,Y:POKE24568,255:POKEV+21,1
BA 560 POKE53265,PEEK(53265)AND239
QR 570 POKE53265,59:POKE53272,120:POKE56576,PEEK(5
    6576)AND254
DF 580 IFZZ=2THENPOKE53270,216
BB 590 POKE53265,PEEK(53265)OR16
MA 600 POKESID+24,15
JQ 610 LE=L*8:WE=W*8
JS 620 X=XX:Y=YY
HX 630 GETK$:IFK$="{F1}"THENGOSUB1150
BM 640 TG=1-TG:POKEV+21,TG:J2=PEEK(56320)
RJ 650 FR=J2AND16:JV=15-(J2AND15)
HC 660 IF FR=0THENGOSUB950:GOTO630
RE 670 IFJV=1 AND Y>YY THENY=Y-LE:B=B-INT(40/W):GO
    TO710
SH 680 IFJV=2 AND Y<MYX THENY=Y+LE:B=B+INT(40/W):G
    OTO710
SP 690 IFJV=4 AND X>XX THENX=X-WE:B=B-1:GOTO710
AH 700 IFJV=8 AND X<MAX THENX=X+WE:B=B+1
SA 710 TR=(X>255)
GD 720 POKEV+16,TR*(-1)
KA 730 POKEV,X+(TR*255):POKEV+1,Y
GC 740 GOTO630
SP 750 GOSUB1020:IFB$=D$THEN EN=1
EA 760 FORN=0TOL-1:FOR N1=0TOW-1
QP 770 Y1=S(X1)+(N1*8)+(N*320):C1=C(X1)+N1+N*40:V1
    =C1-31744
QX 780 Y2=S(X2)+(N1*8)+(N*320):C2=C(X2)+N1+N*40:V2
    =C2-31744
AS 790 YL=FN LO(Y1):YH= FN HI(Y1)
RK 800 XL=FN LO(Y2):XH= FN HI(Y2)
BH 810 POKE165,YL:POKE166,YH:POKE167,XL:POKE168,XH
FG 820 SYS 828
KG 830 IFZZ=2 THEN POKE49160,PEEK(C1):POKEC1,PEEK(
    C2):POKEC2,PEEK(49160)
GM 840 POKE49161,PEEK(V1):POKEV1,PEEK(V2):POKEV2,P
    EEK(49161)
DH 850 NEXT:NEXT
PD 860 IFEN=1THEN1060
GQ 870 RETURN
AR 880 POKE53270,200:POKE53265,27:POKE53272,21:POK
    E56576,PEEK(56576)OR1:POKEV+21,0
HQ 890 POKE53280,0:POKE53281,0:PRINT"{CLR}{DOWN}
    {1}{3 SPACES}LOAD ANOTHER PICTURE PUZZLE AN
    D"

```

```
PS 900 PRINT "{2 DOWN}{9 SPACES}PLAY AGAIN {RVS}{Y/N}{OFF} ";
GB 910 GETA$:IFA$=""THEN910
EG 920 IFA$="N"THEN POKE55,0:POKE56,160:CLR:STOP
SS 930 IFA$="Y"THENPRINT:GOTO30
HB 940 GOTO910
CP 950 FL=FL+1
EM 960 IFFL=2THENX2=B:IFX2=X1THEN FL=1:RETURN
QR 970 IF FL=2 THEN FL=0:GOSUB990:GOSUB750:RETURN
DF 980 X1=B:GOSUB1000:RETURN
KC 990 HF=110:LF=0:GOSUB1010:RETURN
EJ 1000 HF=70:LF=0:GOSUB1010:RETURN
HP 1010 POKESID+1,HF:POKESID,LF:POKESID+4,17:POKESID+4,16:RETURN
KF 1020 IFX1>X2THENX3=X1:X1=X2:X2=X3
PF 1030 X1$=MID$(B$,X1,1):X2$=MID$(B$,X2,1)
BD 1040 B$=LEFT$(B$,X1-1)+X2$+MID$(B$,X1+1,X2-X1-1)+X1$+RIGHT$(B$,LEN(B$)-X2)
AD 1050 RETURN
HS 1060 FOR R=0TO2
JC 1070 POKESID+5,85:POKESID+6,85:POKESID+12,85:POKESID+13,85
DM 1080 POKESID+4,33:POKESID+11,17
JP 1090 FORN=0TO5:FORX=0TO3:V(X)=PEEK(850+(N*4)+X):NEXT
AQ 1100 POKESID+1,V(0):POKESID,V(1):POKESID+8,V(2):POKESID+7,V(3):POKE53280,V(2)
JH 1110 IFH1=50THEN FOR T=1TO200:NEXT
BP 1120 FORT=1TO125:NEXT:NEXT
CJ 1130 POKESID+4,32:POKESID+11,16:FORW=1TO500:NEXT:NEXT
JJ 1140 GOTO880
AP 1150 IFZZ=2THENGOSUB1620
JD 1160 POKE53272,8:POKE56576,(PEEK(56576)AND252)OR1
HD 1170 POKEV+21,0
EG 1180 GETK$:IFK$<>"{F1}"THEN1180
DG 1190 IFZZ=2THENGOSUB1620
QS 1200 POKE53272,120:POKE56576,(PEEK(56576)AND252)OR2
RE 1210 POKEV+21,1
RR 1220 RETURN
GM 1230 PRINT "{3 DOWN} PLEASE INSERT DISK CONTAINING {RVS}DOODLE{OFF}":N=1
BF 1240 PRINT TAB(8);"OR {RVS}KOALA{OFF} PICTURE FILES"
KR 1250 PRINT "{2 DOWN}{7 SPACES}PRESS ANY KEY WHEN READY"
MX 1260 GETK$:IF K$=""THEN1260
```

```

MX 1270 PRINT "{CLR}{DOWN}"TAB(12);"{RVS} PICTURE F
      ILES {2 DOWN}"
RR 1280 OPEN1,8,0,"$"
JR 1290 GET#1,B$:IFST<>0THEN1400
QM 1300 IFN>19THEN1400
EQ 1310 IFB$<>CHR$(34)THEN1290
RG 1320 N$=""
HG 1330 GET#1,B$:IFB$<>CHR$(34)THEN N$=N$+B$:GOTO1
      330
CQ 1340 GET#1,B$:IFB$=CHR$(32)THEN1340
JD 1350 IFLEFT$(N$,1)=" "THEN N=N-1:GOTO1380
JF 1360 IFLEFT$(N$,2)<>"DD"ANDLEFT$(N$,1)<>"[1]" T
      HENN=N-1:GOTO1380
RM 1370 N$(N)=N$:PRINTTAB(12)N$(N)
FC 1380 GET#1,B$:IFB$<>" "THEN1380
FX 1390 IF ST=0THEN N=N+1:GOTO1290
DJ 1400 CLOSE1
KG 1410 IFN=1THENPRINT "{6 SPACES}{RVS}NO PICTURES
      {SPACE}ON THIS DISK":FORI=1TO2000:NEXT:GOT
      O1230
PC 1420 PRINT "{DOWN}{2 SPACES}(USE CURSOR UP/DOWN
      {SPACE}TO HIGHLIGHT)"
QF 1430 PRINT "{DOWN}{4 SPACES}{RVS}PRESS RETURN TO
      LOAD PICTURE{DOWN}"
MR 1440 A=1:C1=0:C=125:GOSUB1510
SX 1450 K=PEEK(197):SH=PEEK(653):IFK<>7ANDK<>1THEN
      1450
CA 1460 IFK=7ANDSH=0ANDA<(N-1)THENC1=1:GOSUB1500
GF 1470 IFK=7AND SH=1ANDA>1THENC1=-1:GOSUB1500
KC 1480 IFK=1THENRETURN
HA 1490 GOTO1450
JR 1500 FORI=0TO24:POKE55296+I+(A)*40+C,8:NEXT
KE 1510 A=A+C1:FORI=0TO24:POKE55296+I+(A)*40+C,1:N
      EXT
SS 1520 RETURN
EQ 1530 POKE49872,0:POKE49873,92:POKE49874,0:POKE4
      9875,96:POKE49876,0
DA 1540 POKE49877,128:POKE49878,0:SYS49879
XX 1550 POKE49872,0:POKE49873,96:POKE49874,0:POKE4
      9875,128:POKE49876,0
JB 1560 POKE49877,160:POKE49878,0:SYS49879:RETURN
KP 1570 POKE49872,64:POKE49873,127:POKE49874,39:PO
      KE49875,131:POKE49876,0
QC 1580 POKE49877,92:POKE49878,0:SYS49879
FM 1590 POKE49872,40:POKE49873,131:POKE49874,15:PO
      KE49875,135:POKE49876,0
CH 1600 POKE49877,216:POKE49878,0:SYS49879
RR 1610 POKE49877,196:POKE49878,0:SYS49879:RETURN
MG 1620 POKE49872,0:POKE49873,216:POKE49874,0:POKE
      49875,220:POKE49876,0

```

```

KB 1630 POKE49877,196:POKE49878,1:SYS49879:RETURN
GJ 1640 DATA 160,0,177,165,153,0,192,177
QH 1650 DATA 167,145,165,185,0,192,145,167
EG 1660 DATA 200,192,8,208,237,96,25,30
MA 1670 DATA 18,209,33,135,25,30,42,62
SK 1680 DATA 31,165,50,60,37,162,42,62
AP 1690 DATA 31,165,50,60,37,162
RS 1700 DATA 173,210,194,205,208,194,173
SR 1710 DATA 211,194,237,209,194,144,80,173
JF 1720 DATA 208,194,133,251,173,209,194
SE 1730 DATA 133,252,173,212,194,133,253
HX 1740 DATA 173,213,194,133,254,173,210
FM 1750 DATA 194,133,247,173,211,194,133
CE 1760 DATA 248,173,214,194,133,249,160,0
ER 1770 DATA 177,251,166,249,240,8,133,250
SS 1780 DATA 177,253,145,251,165,250,145
KS 1790 DATA 253,230,251,208,2,230,252,230
AP 1800 DATA 253,208,2,230,254,165,252,197
EB 1810 DATA 248,208,222,165,247,197,251,48
RC 1820 DATA 3,76,10,195,96
QC 1830 DATA 170,192,0,187,0,0,169,0
DM 1840 DATA 0,183,64,0,193,144,0,0
AF 1850 DATA 116,0,0,25,0,0,7,64
BK 1860 DATA 0,1,144,0,0,116,0,0
MK 1870 DATA 25,0,0,7,0,0,0,0
EX 1880 DATA 0,0,0,0,0,0,0,0
HS 1890 DATA 0,0,0,0,0,0,0,0
RR 1900 DATA 0,0,0,0,0,0,0,197

```

Number Construction Kit

Gerald W. Rightmer

You've got the tools, materials, and a goal in this construction project. All you have to do is think hard and work fast. For children, an excellent program for practicing math skills. For adults, a real brainteaser.

Your bid has just been accepted for a new construction project—to build a three-digit number between 100 and 999. Fortunately you have your number construction kit, which includes everything you need for the task.

Your tools are the four basic arithmetic operators: $+$, $-$, $*$, and $/$. Your building materials are single digit integers—three, four, or five of them, depending on the level of play you choose. And, as with all construction projects, there's a deadline—one, two, or three minutes, depending on your level of expertise.

Numbers 'R Us

"Number Construction Kit" is written entirely in BASIC. Type in the program and save a copy to tape or disk.

To play, load the program and type **RUN**. You're first asked for the number of players. Press 1 or 2. Then you're asked to choose one of three difficulty levels: (1) Beginner, (2) Intermediate, or (3) Advanced. Press the number key which corresponds to your choice. At the beginner level, you're given five numbers for materials and three minutes to complete the construction project. At the intermediate level, there are four numbers to build with and two minutes to finish. And, if you're up to it, the advanced level offers only one minute and three numbers to work with.

After selecting the number of players and a difficulty level, you're ready to begin. The project total (your goal) and materials (the numbers you're given to work with) are presented and you're prompted to press any key to begin. As soon as the game begins, the timer starts counting down. First select one of the numbers from the materials listed. Then

choose one of the operators: the plus or minus sign (on the top row of the keyboard), the multiplication sign (on the second row beneath the minus sign), or the division sign (at the bottom row next to the right SHIFT key).

Begin building toward your project goal without hesitation. If you reach the project goal before time runs out, press RETURN. Each game consists of five rounds. In the two-player game, one round consists of one turn for each player. After five rounds, the game ends, and you're asked if you'd like to play again. At this point, you can change the number of players and the difficulty level.

Staying in Bounds

Like any construction project, this one has constraints. First, there are only 15 spaces with which to work. This means that you can use only eight numbers and seven operators to complete the project. You're allowed to move across the computation line using the cursor keys, and change numbers or operators simply by typing over the existing characters. A running total is displayed to the right of the equation, allowing you to instantly monitor your progress. Since the target number is positive and greater than 100, the program doesn't allow numbers less than 0 in the target number. Numbers greater than 9999 cannot be created during the construction process. Error messages will appear, and the game will pause if the running total is greater than 9999 or less than 0.

Scoring is based on how close you are to the target, the difficulty level, and how quickly you respond. If your total at the point when time expires is greater than 200 from the proposed project total, no score is awarded for the round. A round is terminated either by the clock or by pressing the RETURN key.

The program randomly generates numbers for the project goal and materials, excluding duplicates and zeros. The only keys that work are the designated materials (numbers) keys, the four operators, left- and right-cursor keys, DELETE, and RETURN. The DELETE key deletes only from the end of the line and removes only the character under the cursor. This means that no gaps can be created in the middle of a computation line (this would result in a meaningless arithmetic expression).

Number Construction Kit is a challenge to any age, and it can be an entertaining way to reinforce basic arithmetic skills in children. Since immediate feedback is given throughout the game, a child can learn much by trial and error.

For those simply looking to win the game, a few strategy tips might be helpful. The first step is to study the project number and the materials before pressing a key to start construction. For larger, more difficult numbers, it's usually easier to get within range of the project as fast as possible and to fine-tune your answer with the remaining time and space. Remember, the materials may be used as often as necessary—you have an unlimited supply.

Number Construction Kit

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

```

AF 10 REM{2 SPACES}*NUMBER CONSTRUCTION KIT*
DG 20 POKE53280,.:POKE53281,.:CC=211:Q=214:Y=198:C
    0=55856:SC=1622:B$=CHR$(150)
JB 30 Z=200:F$=CHR$(156):G$=CHR$(5):H$=CHR$(98):I$
    =CHR$(99):J$=CHR$(147)
GB 40 K$=CHR$(174):L$=CHR$(173):M$=CHR$(189):N$=CH
    R$(176):O$=CHR$(32):Q$=CHR$(125)
XQ 50 U$=CHR$(157):W$="{4 SPACES}":P=1:X=32
EQ 60 PRINTJ$B$SPC(7)N$;:FORI=.TO22:PRINTI$;:NEXT:
    PRINTK$
DX 70 PRINTSPC(7)H$G$"NUMBER CONSTRUCTION KIT"B$H$
    :PRINTSPC(7)L$;:FORI=.TO22
SK 80 PRINTI$;:NEXT:PRINTM$:PRINT:PRINT:PRINTF$"TO
    OLS:"G$" + - * /"SPC(19)B$"TIME"
DH 90 PRINT:PRINTF$"MATERIALS: "G$"0 0 0 0 0"SPC(1
    3)"0:00"
AM 100 PRINT:PRINTSPC(9)F$N$I$I$I$I$K$:PRINT"PROJECT
    : "H$G$"000"F$H$
MP 110 PRINTSPC(9)L$I$I$I$I$M$:PRINT:PRINT:PRINTSPC(
    X)B$N$I$I$I$I$I$K$
SB 120 PRINTG$"0 ";:FORI=.TO6:PRINT"+ 0 ";:NEXT:PR
    INTCHR$(146)B$="H$G$"0000"B$H$
XA 130 FORI=.TO14:PRINTCHR$(163)" ";:NEXT:PRINTSPC
    (2)L$I$I$I$I$I$M$:FORI=.TO3:PRINT
RR 140 NEXT:PRINTSPC(24)"ROUND "G$"0"B$:PRINT:PRIN
    TSPC(24)"PLAYER 1: "G$"0000"B$
SM 150 PRINTSPC(24)"PLAYER 2: "G$"0000";
EE 160 POKEQ,18:PRINT:PRINT"NUMBER OF PLAYERS? ";
XC 170 GETP$:IFP$=""THEN170
DE 180 T=ASC(P$)-48:IF(T<>1ANDT<>2)ORP$=""THEN170
MQ 190 PRINT:PRINT:PRINT"1 - BEGINNER":PRINT"2 - I
    NTERMEDIATE":PRINT"3 - ADVANCED"

```

```

BC 200 POKEQ,18:PRINT:PRINT"LEVEL OF DIFFICULTY? "
;
XP 210 GETD$:IFD$<>"1"ANDD$<>"2"ANDD$<>"3"THEN210
CF 220 GOSUB610:D=1:L(1)=.:L(2)=.:U=.:M=3-(D$<"3")
-(D$<"2"):LE=4-VAL(D$)
BK 230 POKEQ,20:PRINT:PRINTSPC(29)D:R$="000000":K=
98+(D$>"1")+(D$>"2")
AJ 240 POKEQ,6:PRINT:PRINTSPC(X)K-95U$":00":N=1:G=
.:FORI=1TO5:M(I)=.:NEXT
FA 250 R=INT(RND(.)*9)+1:IFR=M(1)ORR=M(2)ORR=M(3)O
RR=M(4)ORR=M(5)THEN250
SA 260 FORI=NT05:M(I)=R:NEXT:POKEQ,6:PRINT:IFN<MTH
ENN=N+1:GOTO250
MS 270 POKEQ,6:PRINT:PRINTSPC(11)W$W$ " ":POKEQ,6:P
RINT:PRINTSPC(10);
FQ 280 N=1:O=1:G=.:T(1)=.:FORI=1TOM
DC 290 PRINTM(I);:NEXT:V=INT(RND(.)*900)+100:POKEQ
,9:PRINT:PRINTSPC(9)V;:POKEQ,9
JP 300 PRINT:PRINTSPC(9)F$Q$G$
DX 310 POKEQ,18:PRINT:PRINT"PLAYER"P:PRINT:PRINT"P
RESS ANY KEY TO"
HE 320 PRINT"BEGIN CONSTRUCTION!":POKEQ,14:PRINT:F
ORI=.TO28:PRINT0$;:NEXT
PR 330 POKEQ,14:PRINT:POKECC,33:PRINTW$
CK 340 WAIT,1:GETA$:POKEQ,20:PRINT:FORI=.TO15:PRI
NT0$;
EJ 350 POKEC0+I,7:NEXT:TI$="000000"
HA 360 POKEQ,14:PRINT:PRINTSPC(33)W$:FORI=1TO8:B(I
)=T(I):NEXT:L=G
BS 370 FORI=1TO7:B(I+1)=(O(I)=-3)*B(I+1)-(O(I)<>-3
)*B(I+1):NEXT:M%=.:I=.:F=.
RB 380 I=I+1:IF(O(I)=-6ORO(I)=-1)ANDI<L/2THEN440
KA 390 H=M%:IFH+B(I)>9999THENE$="TOTAL TOO LARGE -
PLEASE CORRECT!":F=1:GOTO450
QD 400 IFH+B(I)<.THENE$="NEGATIVE TOTAL - PLEASE C
ORRECT!":F=1:GOTO450
PH 410 M%=M%+B(I):IFI<L/2THEN380
XG 420 POKEQ,14:PRINT:PRINTSPC(X)M%;
FK 430 POKEQ,14:PRINT:PRINTSPC(X)B$Q$G$:N%=M%:GOTO
480
MP 440 B(I+1)=-(B(I)*B(I+1))*(O(I)=-6)-(B(I)/B(I+1
))*(O(I)=-1):GOTO380
BA 450 POKEQ,16:PRINT:PRINTE$:POKEB,X:N=N+(N<15):G
=G+(N=GANDG<>15):O=-O
QK 460 IFN=G+1ANDF=1THENPOKEQ,14:PRINT:POKEQ,14:PR
INT:PRINTSPC(X)N%
PF 470 IFN=G+1ANDF=1THENF=.:PRINTCHR$(145)SPC(X)B$
Q$G$:WAIT,1
FH 480 IFF=.THENPOKEQ,16:PRINT:FORI=.TOX:PRINT0$;:
NEXT:PRINT

```



```

JS 490 B=SC+N*2:A=PEEK(B):POKEB,A+128:IFNOT((N=15)
AND(O<.) ) THENO=-O
XE 500 GETC$:IFASC(MID$(TI$,4,1))<K-47THENGOSUB580
:IFC$=" " THEN500
HP 510 C$=C$+"?":C=ASC(C$)-48:IFASC(MID$(TI$,4,1))
=K-47ORC=-35 THEN620
RC 520 S=(N+1)/2:IF(C=-6ORC=-5ORC=-3ORC=-1) ANDO> .A
NDN<16 THENO(S)=C:GOTO570
KS 530 IF(C=M(1)ORC=M(2)ORC=M(3)ORC=M(4)ORC=M(5)) A
NDO< .ANDN<16 THEN(T(S)=C:GOTO570
CX 540 IFC=-28ANDN=GANDN> . THENPOKEB,X:G=G-1:O=-O:T
(S)=-T(S)*(N>1):GOTO360
BK 550 IFF=1ORNOT((C=-19AND(N<G+1)ANDN<15)OR(C=109
ANDN>1)) THEN500
GS 560 IF(C=-19AND(N<G+1)ANDN<15)OR(C=109ANDN>1) TH
ENPOKEB,A:N=N-(C=-19OR1):GOTO490
GS 570 POKEB,C+48:G=G-(N>G):N=N-(N<15):GOTO360
GX 580 POKEQ,6:PRINT:T$=TI$:IFT$=R$ THENRETURN
PC 590 R$=T$:V$=CHR$(K-ASC(MID$(T$,4,1))):POKECC,3
3:IFV$=J$ THENRETURN
CC 600 PRINTV$: "CHR$(101-ASC(MID$(T$,5,1)))CHR$(1
05-ASC(MID$(T$,6,1))):RETURN
DX 610 POKEQ,18:PRINT:FORI=.TO4:FORJ=.TO21:PRINTOS
;:NEXT:PRINT:NEXT:RETURN
RX 620 POKEB,A:E=V-M$:E=E+2*E*(E<.) :E=-E*(E<=Z)-Z*
(E>Z):U=Z-E
GR 630 TR=LE*60-(60*VAL(MID$(T$,4,1))+VAL(RIGHT$(T
$,2))):U=INT((Z-E)*(4-LE)*TR/10)
AB 640 L(P)=L(P)+U:POKEQ,21+P:PRINT:PRINTSPC(34)W$
CHR$(145)
CX 650 PRINTSPC(33)L(P);:D=D+INT(P/T):P=1-(T=2ANDP
=1):IFD<6 THEN230
PK 660 GOSUB610:POKEQ,18:PRINT:PRINT"GOOD GAME!":P
RINT:PRINT"PLAY AGAIN? (Y/N) "
PE 670 GETC$:IFC$<>"Y"ANDC$<>"N" THEN670
PX 680 PRINTJ$:IFC$="Y" THENRUN

```

Pirate Cove

Barbara Schulak

Children can enter a swashbuckling world of pirates as they attempt to find Treasure Island and the treasure chest in this miniadventure. While providing entertainment and challenge, it also sharpens map-reading and direction skills.

In "Pirate Cove," your child plays the role of a buccaneer who's found a map showing the locations of ten islands. Starting from Pirate Cove, he or she must navigate around the territory, trying to find Treasure Island. Along the way, the child may find Spice Island, Mermaid's Lagoon, or Skeleton Island, among others. Also, the child may search sunken ships or fight the Sea Monster. The game is over when Treasure Island has been found and the treasure chest successfully opened.

Playing the game is simple. In fact, older children may be able to play the game without instructions. After typing in the program, save a copy to tape or disk. To play, load the program and type **RUN**.

Land Ho!

At each land location on the map you're offered five options. Two of these options involve searching the area. The third choice is to read the map, the fourth to sail on, and the last choice to list what you have in your possession (inventory). Make your selection by pressing the CRSR UP key to move the red bar to your choice, and then pressing the RETURN key. If you choose to sail on, the program asks which direction you want to go (north, south, east, or west). Again, use the red bar to make this choice.

If the Read Map option is chosen, the screen displays the navigational map. Islands are shown by green, water by blue, and your current location by yellow. Choosing the List Inventory option lets you see what items you've gathered on your journey. (A few of these can be quite helpful.) When you're sailing (the High Seas option), you may choose either to read the map or to sail on.

Each time the program is run, the location of the islands changes so that the game may be played more than once. More advanced players might want to play the game without reading the map, charting their way on paper instead.

Programmers might note that sprites (expanded horizontally) have been used to form a highlighting bar for the choice selection. It's essential to make the choosing options as easy and foolproof as possible for children. This method seemed a good choice despite lengthening the program a bit. Sprites (expanded both horizontally and vertically) were also used for the various pictures.

Pirate Cove

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

```
KX 10 X=RND(-TI):S=54272:V=53248
DP 20 DIM T(25),R$(25),R1$(25),I$(8)
DC 30 DIM Q$(2,8),Q1$(2,25)
CX 40 DIM A$(2,8),A1$(2,25)
EB 50 DIM D(125),LO(25),LC(25),L(25)
XE 60 POKE53281,0:POKE53280,0:PRINT "{CLR}"
AM 70 FORI=0TO24:POKES+I,0:NEXT
XR 80 PRINT "{5 DOWN}"TAB(14)"[8]PIRATE COVE":GOSUB
1470:GOSUB1470
QB 90 PRINT "{9 DOWN}"{7 RIGHT}A MINI-ADVENTURE FOR
{SPACE}KIDS"
KQ 100 PRINTTAB(13)"[7]{3 DOWN}[PLEASE WAIT]"
GQ 110 FORI=12288TO12607:READA:POKEI,A:NEXT
HB 120 POKE2040,193
AA 130 FORI=2041TO2043:POKEI,197:NEXT
XX 140 FORI=V+40TOV+42:POKEI,2:NEXT
AH 150 POKEV+39,10:POKEV+37,7:POKEV+38,5
BB 160 POKEV+29,15:POKEV+23,1
EH 170 POKEV+28,1:POKEV+27,14
AS 180 POKEV+16,1:POKEV,0:POKEV+1,75
EX 190 X=61:Y=81:POKEV+21,1
DF 200 POKEV+2,X:POKEV+4,X+48:POKEV+6,X+96
HK 210 FORI=12608TO12608+23:POKEI,255:NEXT
CH 220 FORI=12608+24TO12608+63:POKEI,0:NEXT
RR 230 FORI=1TO10:READR$(I):NEXT
JC 240 FORI=1TO8:READQ$(1,I):READA$(1,I):NEXT
KS 250 FORI=1TO8:READQ$(2,I):READA$(2,I):NEXT
CF 260 FORI=1TO100:READD(I):NEXT
GH 270 FORI=1TO25:READLO(I):NEXT
DE 280 FORI=11TO25:R$(I)="HIGH SEAS":NEXT
JM 290 FORI=1TO10:LC(I)=1:NEXT
RS 300 FORI=11TO25:LC(I)=2:NEXT
CG 310 F1=0:F2=0:Q=1
```

```

XS 320 FORI=1TO8:I$(I)="":NEXT
MF 330 FORI=1TO25:T(I)=0:NEXT
QH 340 FORI=1TO25
SB 350 RN=INT(RND(1)*25)+1
HS 360 IFT(RN)=1THEN350
CG 370 T(RN)=1:R1$(RN)=R$(I):L(RN)=LC(I)
FK 380 NEXT
RP 390 Z=1:FORI=1TO25
PA 400 IFR1$(I)="TREASURE ISLAND"THEN480
FJ 410 IFR1$(I)="SEA MONSTER'S LAIR"THEN480
PM 420 IFR1$(I)="HIGH SEAS"THEN480
XH 430 Q1$(1,I)=Q$(1,Z)
DF 440 Q1$(2,I)=Q$(2,Z)
GH 450 A1$(1,I)=A$(1,Z)
MM 460 A1$(2,I)=A$(2,Z)
JH 470 Z=Z+1
PA 480 NEXT
FQ 490 FORI=1TO25:IFR1$(I)="PIRATE COVE"THENR=I:RR
=R
QA 500 NEXT
PH 510 TM=R:IFR1$(R)="TREASURE ISLAND"THEN1040
QG 520 PRINT "{CLR}{8}CURRENT LOCATION: {GRN}";
DJ 530 GOSUB1470
CA 540 PRINTR1$(R){8}
KQ 550 PRINT "{HOME}{2 DOWN}"
SG 560 IFR1$(R)="HIGH SEAS"THEN960
GE 570 IFR1$(R)="SEA MONSTER'S LAIR"THEN1270
DX 580 POKE2040,193:POKEV+39,10:POKEV+37,7:POKEV+3
8,5
AF 590 Y=81:K=1:D=5
HJ 600 POKEV+3,Y:POKEV+5,Y:POKEV+7,Y:POKEV+21,15
MF 610 L=LEN(Q1$(1,R)):L2=LEN(Q1$(2,R))
XX 620 PRINTTAB(14-L/2){8}{DOWN}"Q1$(1,R)
ES 630 PRINTTAB(14-L2/2)Q1$(2,R)
BG 640 PRINTTAB(10)"READ MAP"
KP 650 PRINTTAB(10)"SAIL ON"
FG 660 PRINTTAB(7)"LIST INVENTORY"
FH 670 GOSUB1790
XG 680 ONKGOTO860,860,1520,690,1910
AS 690 PRINT "{CLR}{3 DOWN}{8}SAIL: {YEL}";
DP 700 Y=81:D=4:K=1
MB 710 POKEV+3,Y:POKEV+21,3
HK 720 PRINT "{DOWN}{8}TAB(5)"NORTH"
EP 730 PRINTTAB(5)"SOUTH"
ER 740 PRINTTAB(5)"EAST"
GG 750 PRINTTAB(5)"WEST"
QM 760 GOSUB1790
SK 770 IFK=1THENR=R-5
CJ 780 IFK=2THENR=R+5
CF 790 IFK=3THENR=R+1

```

```

DP 800  IFK=4THENR=R-1
PQ 810  IFD((TM-1)*4)+K)=1THENPOKEV+21,1:GOTO510
CE 820  POKE2040,196:POKEV+39,6:POKEV+37,1:POKEV+38
      ,7:POKEV+21,1
MH 830  PRINT"{4 DOWN}STORM AHEAD-GO ANOTHER WAY":P
      RINT"{8 DOWN}":GOSUB1890:GOSUB1470
FD 840  GOSUB1860:R=TM:GOTO690
GQ 850  -----REM SEARCH-----
EG 860  PRINT"{4 DOWN}{PUR}YOU FIND "A$(K,R)
HP 870  G$=A$(K,R)
FQ 880  IFG$="A KEY"THENF1=1
ES 890  IFG$="A CUTLASS"THENF2=1
BX 900  IFLEFT$(G$,7)="NOTHING"THEN940
MQ 910  FORI=1TO8:IFG$=I$(I)THEN940
KP 920  NEXT
QE 930  I$(Q)=G$:Q=Q+1
PS 940  PRINTTAB(7)"{6 DOWN}":GOSUB1860:GOTO510
RF 950  -----REM HIGH SEAS-----
FQ 960  POKE2040,192:POKEV+39,6:POKEV+37,1:POKEV+38
      ,2
FC 970  K=1:D=2:Y=81
PB 980  POKEV+3,Y:POKEV+5,Y:POKEV+21,7
JQ 990  PRINTTAB(7)"{DOWN}[8]READ MAP"
QD 1000  PRINTTAB(7)"SAIL ON"
QM 1010  GOSUB1790
DE 1020  ONKGOTO1520,690
BQ 1030  -----REM TREASURE ISLAND-----
DF 1040  POKE2040,193:POKEV+39,10:POKEV+37,7:POKEV+
      38,5
QH 1050  PRINT"{CLR}[8]{9 DOWN}YOU HAVE LANDED AT T
      REASURE ISLAND":GOSUB1890:GOSUB1470
AP 1060  PRINT"{DOWN}AND FOUND THE TREASURE CHEST"
BS 1070  PRINT"{2 DOWN}[6]PRESS ANY KEY TO SEE IF I
      T WILL OPEN."
ME 1080  GETK$:IFK$=""THEN1080
KE 1090  IF F1=1THEN1150
DA 1100  PRINT"{CLR}{9 DOWN}[8]YOU DON'T HAVE THE K
      EY"
BJ 1110  PRINT"{DOWN}YOU MUST CONTINUE ON YOUR JOUR
      NEY,"
QQ 1120  PRINT"{DOWN}FIND THE KEY AND RETURN TO
KA 1130  PRINT"{DOWN}TREASURE ISLAND.{3 DOWN}"
PQ 1140  GOSUB1860:GOTO690
EX 1150  POKE2040,194:POKEV+39,6:POKEV+37,1:POKEV+3
      8,2:GOSUB1890
FS 1160  PRINT"{CLR}[8]YOUR KEY OPENS THE CHEST."
BM 1170  PRINT"YOU WIN"
RK 1180  PRINT"{4 DOWN}[6]OTHER TREASURES:[8]{DOWN}
      "
SR 1190  FORI=1TO8:PRINTI$(I):NEXT

```

```
GG 1200 PRINT "{2 DOWN}{7}PLAY AGAIN [Y/N]
CP 1210 GETK$:IFK$=""THEN1210
BP 1220 IFK$="Y"THENPOKEV+21,0:GOTO1250
MG 1230 IFK$="N"THENSYS2048:END
EQ 1240 GOTO1210
KP 1250 PRINTTAB(9)" {3 DOWN}{7}[PLEASE WAIT A MOMEN
NT]":GOTO310
ES 1260 -----REM MONSTER-----
CJ 1270 POKEV+39,6:POKEV+37,2:POKEV+38,1:POKE2040,
195
XP 1280 Y=81:D=2:K=1:POKEV+3,Y:POKEV+5,Y:POKEV+21,
7:GOSUB1890
KG 1290 PRINTTAB(6)" {DOWN}SAIL AWAY"
CG 1300 PRINTTAB(8)"FIGHT"
RX 1310 GOSUB1790
DX 1320 IFK=2THEN1390
MK 1330 RN=RND(1):IFRN>.8THENTM=R:GOTO690
RJ 1340 PRINT "{3 DOWN}NOT FAST ENOUGH"
JJ 1350 PRINT "THE MONSTER DAMAGED YOUR SHIP"
QM 1360 PRINT "YOU MUST GO BACK TO PIRATE COVE AND
JE 1370 PRINT "START OVER AGAIN. {4 DOWN}"
GJ 1380 R=RR:GOSUB1860:GOTO510
SJ 1390 IFF2=0THEN1430
SB 1400 PRINT "{2 DOWN}YOU DEFEATED THE SEA MONSTER
"
JE 1410 PRINT "YOU MAY CONTINUE ON YOUR JOURNEY.
{4 DOWN}"
SE 1420 TM=R:GOSUB1860:POKEV+21,1:GOTO690
HK 1430 PRINT "{2 DOWN}YOU CANNOT FIGHT THE MONSTER
WITHOUT"
JX 1440 PRINT "A CUTLASS"
PX 1450 GOTO1360
RF 1460 -----REM SOUND SUB-----
JE 1470 POKES+24,15:POKES+5,9:POKES+4,17
GD 1480 POKES+1,6:POKES,108:FORT=1TO1000:NEXT
AQ 1490 POKES+4,16:FORT=1TO25:NEXT
EX 1500 RETURN
RM 1510 -----REM DRAW MAP-----
XK 1520 POKEV+21,0
FR 1530 PRINT "{CLR}{2 DOWN}"TAB(7)" {8}NAVIGATION M
AP {2 DOWN}"
RB 1540 X$="O{Y}P"
XQ 1550 Y$="H{ }N}"
HC 1560 Z$="L{P}@"
CK 1570 FORI=1TO5:PRINTTAB(7);
DF 1580 FORJ=1TO5:PRINTX$;:NEXT:PRINT
AE 1590 PRINTTAB(7);
SM 1600 FORJ=1TO5:PRINTY$;:NEXT:PRINT
CE 1610 PRINTTAB(7);
AK 1620 FORJ=1TO5:PRINTZ$;:NEXT:PRINT
```

```

AK 1630 NEXT
SQ 1640 FORI=1TO25
FM 1650 IFL(I)=1THENPOKELO(I),81:POKELO(I)+S,5
QG 1660 IFL(I)=2THENPOKELO(I),45:POKELO(I)+S,6
EM 1670 NEXT
JP 1680 POKELO(R),81:POKELO(R)+S,7
KQ 1690 PRINT "{HOME}{6 DOWN}"TAB(27)"{GRN}Q[8] = L
    AND"
RK 1700 PRINTTAB(27)"{BLU}-[8] = WATER"
BE 1710 PRINTTAB(27)"{YEL}Q[8] = CURRENT"
SB 1720 PRINTTAB(31)"LOCATION"
FM 1730 PRINTTAB(33)"{3 DOWN}{YEL}N"
KP 1740 PRINTTAB(33)"-"
JK 1750 PRINTTAB(31)"W-Q-E"
HM 1760 PRINTTAB(33)"-"
EQ 1770 PRINTTAB(33)"S"
HK 1780 PRINT "{4 DOWN}":GOSUB1860:GOTO510
HQ 1790 GETK$:IFK$=""THEN1790
GR 1800 IFK$=CHR$(13)THENRETURN
PD 1810 IFK$<>"{DOWN}"THEN1790
EB 1820 K=K+1:Y=Y+8
JM 1830 IFK>DTHENK=1:Y=81
DH 1840 POKEV+3,Y:POKEV+5,Y:POKEV+7,Y
ER 1850 GOTO1790
RX 1860 PRINTTAB(6)"[7][PRESS ANY KEY TO CONTINUE]
    "
MK 1870 GETK$:IFK$=""THEN1870
QG 1880 RETURN
GQ 1890 FORI=1TO11:POKE53280,I
RM 1900 FORI=1TO25:NEXT:NEXT:POKE53280,0:RETURN
EJ 1910 PRINT "{2 DOWN}INVENTORY:{DOWN}"
AA 1920 FORI=1TO8:PRINTI$(I):NEXT
SG 1930 PRINT "{DOWN}":GOSUB1860:POKEV+21,1:GOTO520
AB 1940 DATA 0,0,0,1,192,0,1,192
AQ 1950 DATA 0,4,192,0,4,192,0,20
XR 1960 DATA 192,0,20,192,0,20,193,192
BK 1970 DATA 20,197,192,4,197,192,4,197
QM 1980 DATA 192,1,197,192,1,193,192,240
JS 1990 DATA 192,207,63,255,252,15,255,252
CX 2000 DATA 3,255,252,170,187,190,42,174
DC 2010 DATA 234,10,170,168,0,170,128,255
FG 2020 DATA0,0,0,0,0,0,60,0,0
EG 2030 DATA192,3,195,0,0,51,240,0
QD 2040 DATA15,12,3,248,192,12,8,48
RB 2050 DATA0,50,48,0,194,0,0,192
CB 2060 DATA128,0,192,128,0,0,128,0
GD 2070 DATA0,128,0,2,0,0,86,64
PM 2080 DATA5,85,84,21,85,80,1,85
JF 2090 DATA64,0,21,0,0,0,0,0
DB 2100 DATA5,0,0,1,64,0,0,80

```

QD 2110 DATA0,0,20,0,0,5,0,0
 JG 2120 DATA1,64,0,0,80,21,85,80
 EK 2130 DATA16,0,16,18,170,16,18,34
 ME 2140 DATA16,208,32,16,16,32,28,16
 RR 2150 DATA32,16,208,32,19,21,85,80
 HF 2160 DATA0,0,204,48,12,48,0,195
 AJ 2170 DATA12,12,12,48,0,0,192,0
 GD 2180 DATA0,16,0,0,32,0,0,8
 XA 2190 DATA0,0,8,0,0,8,0,72
 XR 2200 DATA32,33,34,32,136,0,170,0
 DD 2210 DATA0,158,0,10,182,160,160,158
 MC 2220 DATA10,128,182,2,64,170,1,2
 DE 2230 DATA8,128,8,8,32,8,32,32
 QM 2240 DATA16,32,4,0,32,0,0,8
 QF 2250 DATA0,0,4,0,0,0,0,0
 RB 2260 DATA 1,84,0,4,1,64,4,0
 HF 2270 DATA 20,16,0,1,16,80,1,17
 BM 2280 DATA 0,1,64,0,65,64,0,68
 EJ 2290 DATA 16,1,80,5,84,128,128,48
 XQ 2300 DATA 8,8,50,0,0,192,200,32
 KS 2310 DATA 200,192,0,192,194,35,195,32
 GQ 2320 DATA 3,35,0,3,3,8,3,12
 PX 2330 DATA 0,0,12,0,0,12,0,72
 ME 2340 DATA TREASURE ISLAND, SEA MONSTER'S LAIR,
 {SPACE}BLUEBEARD'S BAY
 SE 2350 DATA MERMAID'S LAGOON, CAPT. HOOK'S SWAMP,
 SPICE ISLAND
 SG 2360 DATA JEWEL ISLAND, SKELETON INLET, BARBARY
 COAST,PIRATE COVE
 AP 2370 DATA SIFT THROUGH SAND, A FEW PIECES OF EI
 GHT
 AX 2380 DATA SEARCH ISLAND, NOTHING OF INTEREST.
 GJ 2390 DATA SEARCH OLD GALLEON, SOME SILVER JEWEL
 RY
 AE 2400 DATA SEARCH ISLAND, EXOTIC SPICES
 FP 2410 DATA SEARCH HARBOR, A CUTLASS
 QE 2420 DATA SEARCH SWAMP, NOTHING BUT SNAKES
 PK 2430 DATA SEARCH JUNGLE, A BEAUTIFUL PARROT
 XS 2440 DATA SAIL ALONG COAST, NOTHING OF INTEREST
 .
 GF 2450 DATA SEARCH ALONG DOCK, NOTHING OF INTERES
 T.
 DX 2460 DATA SEARCH NATIVE HUTS, A KEY
 XX 2470 DATA SEARCH STREAM,NOTHING BUT ALLIGATORS
 BX 2480 DATA SEARCH ALONG CLIFFS, NOTHING OF INTER
 EST.
 SX 2490 DATA SEARCH WOODS, NOTHING OF INTEREST.
 XC 2500 DATA SEARCH WRECKED SHIP, SOME GEMS
 AA 2510 DATA SEARCH OLD SHIP, NOTHING BUT AN OLD C
 OMPASS.

AD 2520 DATA COMB BEACHES, A FEW DOUBLOONS
ED 2530 DATA 0,1,1,0,0,1,1,1,0,1,1,1,0,1,1,1,0,1,0
,1
XM 2540 DATA 1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0
,1
MF 2550 DATA 1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0
,1
KE 2560 DATA 1,1,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0
,1
XQ 2570 DATA 1,0,1,0,1,0,1,1,1,0,1,1,1,0,1,1,1,0,0
,1
BE 2580 DATA 1272,1275,1278,1281,1284
GA 2590 DATA 1392,1395,1398,1401,1404
PK 2600 DATA 1512,1515,1518,1521,1524
JF 2610 DATA 1632,1635,1638,1641,1644
MB 2620 DATA 1752,1755,1758,1761,1764

Math Attack

Dan Dickerman

School children of any age can sharpen their math skills with this clever educational program. It offers addition, subtraction, and multiplication challenges at four levels of difficulty.

Many of today's educational programs don't hold a child's attention. Others do—but they're better as games than as educational tools. My goal in writing "Math Attack" was to present a riveting game—with lots of color and sound—in which math problems played the central role. That way, players would receive painless drills in the basic math skills. I designed the math problems as sprites, so they could move smoothly about the screen to play an active role in the game.

Math Attack is easy to learn and easy to play. The premise is fairly simple. Sets of math problems have begun to descend upon an almost helpless city, and the only weapons for its defense are the correct answers. In testing this game, I found that many of the young testers quickly overcame their initial fear of math problems once they saw how easily a descending swarm of them could be eliminated.

The Player's Choices

After typing in the game and saving a copy, load it and type **RUN**. The screen is cleared and the city assembled. The player is then asked which type of problem he or she wishes to tackle: addition, subtraction, or multiplication. To respond, press either the $+$, $-$, or $\times$ key. (The letter x is used to represent multiplication because it's similar to the symbol children use in school.) Division was avoided to lend a more common appearance to the problems and to avoid other complications such as fractions and long decimals.

Next, the player chooses a starting level of difficulty (selected by pressing either 1, 2, or 3) in which upper limits are set on the sizes of the numbers in the incoming problems. This also provides the player with slower starting speeds at higher levels—a kind of reward for choosing a more difficult problem to begin with.

Falling Problems

The onslaught begins. Three math problems appear in the sky, descending menacingly toward the city. The player enters answers from the keyboard, and launches them via the RETURN key. The — key should be used before the answer if a negative value is required, and typos may be erased using the INST/DEL key. A correct answer destroys an attacking problem and lifts the sprite (math problem) beyond the top of the screen, where it soon reappears as a new problem. An incorrect answer sends the whole set of problems closer to the city, or, in some cases, increases the rate of their descent. If a problem reaches the city, it explodes into a cloud which retreats to the top of the screen. It then regenerates as another problem to fall from the skies.

After 20 problems have been correctly answered, the player is informed that the level has been completed, and that the new limits on the problems' sizes have been increased by a factor of ten. In all, there are four basic levels, with combinations of one-, two-, and three-digit numbers, the highest level having problems comprised of two- and three-digit numbers. This final level cannot be accessed directly at the start of the game. It can only be reached by progressing from the lower levels.

The game ends either when the fourth level has been successfully completed (that is, 20 problems on this level have been correctly answered) or after three problems have reached the city and exploded on its surface. At the end of the game, the player is presented with the number of correct and incorrect answers given during the course of the game, and the number of math problems reaching the city (which will always be three, unless the player completes the fourth level). It's up to the parent or teacher to decide whether the student may use a calculator.

When played by children just learning to solve simple math problems, Math Attack provides an exciting and effective drill of the basics which increases in difficulty as the players' skill grows. For us older kids, it will stump the best of us on its higher levels. It's almost impossible to keep the city intact through the fourth level in multiplication without using a calculator.

Math Attack

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

```

EP 10 FOR I=54272TO54296:POKEI,0:NEXT
AH 20 POKE51,127:POKE52,62:POKE55,127:POKE56,62:CL
R:GOTO40
RB 30 PRINT "{HOME}";:FORC=1TO21:PRINTC$:NEXT:PRINT
 "{HOME}":RETURN
AP 40 C$="{39 SPACES}":PRINTCHR$(142);CHR$(8)
ER 50 PRINT "{CLR}{21 DOWN}{BLU}";:POKE53280,0:POKE
53281,0:V=53248:POKE54281,75
QE 60 POKE54296,15:F=54273:AD=F+4:SR=F+5:W=F+3:F2=
F+7:W2=W+7:POKEW+8,0:POKEW+9,240
MM 70 D$="{HOME}{24 DOWN}":TB$=D$+"{20 RIGHT}"
BB 80 PRINT "{3 SPACES}{P}{O}{2 SPACES}{I} {O}
{2 SPACES}{O} {I}{2 SPACES}{O}{P}{O}{I} {I}
{O}{P}{O}{3 SPACES}{O}{O}{P} {P}{O}{I}
{2 SPACES}{O}"
FM 90 PRINT "{RVS}{RIGHT} {T}{2 SPACES}{U}{Y} {U}
{Y}{U} {T} {U}{T}{4 SPACES}{T}{4 SPACES}{U}
{Y}{T}{3 SPACES}{U}{3 SPACES}{Y}{U} "
MH 100 PRINT "{RVS}{RIGHT}{38 SPACES}":BC=PEEK(V+31
)
MM 110 PRINT "{HOME}{2 DOWN}{RIGHT}{PUR}TYPE: {RVS}
+{OFF} (ADDITION)"
XD 120 PRINT "{2 DOWN}{7 RIGHT}{RVS}-{OFF} (SUBTRAC
TION)"
DR 130 PRINT "{2 DOWN}{7 RIGHT}{RVS}X{OFF} (MULTIPL
ICATION)"
QK 140 GETT$:IFT$<>"+ANDT$<>"-ANDT$<>"X"THEN140
GM 150 GOSUB30:DIMN(10,7):POKE56334,PEEK(56334)AND
254:POKE1,PEEK(1)AND251
EK 160 CS=53632:FORX=0TO9:FORY=0TO7:N(X,Y)=PEEK(CS
):CS=CS+1:NEXTY,X
QH 170 SN=-53440*(T$="X")-53592*(T$="+")-53608*(T$
="-")
HF 180 FORY=0TO7:N(10,Y)=PEEK(SN):SN=SN+1:NEXT
CA 190 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1
SR 200 PRINT "{2 DOWN}{2 RIGHT}{YEL}LEVEL:
{3 SPACES}{RVS}1{6 RIGHT}2{6 RIGHT}3"
DX 210 PRINT "{2 DOWN}(LIMITS){3 SPACES}9{5 SPACES}
99{5 SPACES}99"
JF 220 PRINT "{9 SPACES}"T$" 9{4 SPACES}"T$" 9
{4 SPACES}"T$"99"
FA 230 PRINT "{9 SPACES}{3 T}{4 SPACES}{3 T}
{4 SPACES}{3 T}"
CM 240 GETL$:IFL$<>"1"ANDL$<>"2"ANDL$<>"3"THEN240
KD 250 L=VAL(L$):L1=-9*(L=1)-99*(L=2)-99*(L=3):L2=
-9*(L=1)-9*(L=2)-99*(L=3)

```

```

RM 260 GOSUB30:POKEV,100:POKEV+2,175:POKEV+4,250:P
      OKEV+1,0:POKEV+3,0:POKEV+5,0
SP 270 FORX=16000TO16191:READD:POKEX,D:NEXT:POKEV+
      39,5:POKEV+40,3:POKEV+41,8
MR 280 DATA0,0,0,0,0,0,0,0,0
GQ 290 DATA0,0,0,0,0,0,0,0,0
EQ 300 DATA0,0,0,16,0,0,74,0
BF 310 DATA2,82,64,2,74,64,2,82
RE 320 DATA64,17,36,136,9,36,144,41
DS 330 DATA36,148,36,145,36,18,137,72
FC 340 DATA73,82,146,36,137,36,18,82
JG 350 DATA72,170,137,85,0,84,0,0
KS 360 DATA0,170,0,5,85,64,21,85
QR 370 DATA80,42,170,168,85,85,84,85
HE 380 DATA85,84,85,85,84,42,170,168
PB 390 DATA21,85,80,5,85,64,0,170
FG 400 DATA0,0,0,0,0,0,0,0,0
BF 410 DATA0,0,0,0,0,0,0,0,0
RF 420 DATA0,0,0,0,0,0,0,0,0
KE 430 DATA0,0,0,0,0,0,0,0,0
EQ 440 DATA96,0,1,96,64,2,16,35
RP 450 DATA4,15,144,24,10,126,48,4
SP 460 DATA173,96,4,30,18,70,35,192
PR 470 DATA53,65,72,36,129,40,20,129
HF 480 DATA72,28,129,192,36,2,66,124
JF 490 DATA102,98,162,120,80,53,144,152
DK 500 DATA1,73,48,2,6,80,4,2
EA 510 DATA200,8,1,4,144,0,0,0
JJ 520 FORX=16192TO16384:POKEX,0:NEXT
ME 530 POKE2040,253:POKE2041,254:POKE2042,255
GB 540 FORSN=0TO2:GOSUB1080:NEXT:POKEV+21,7:POKEV+
      31,0:GOSUB30:POKE198,0
QF 550 Y(0)=0:Y(1)=30:Y(2)=10:PRINTD$"ANSWERS: {P}"
      ;TAB(19);"BOMBS YOU MISSED: 0";
RX 560 FORX=0TO2:Y(X)=Y(X)+1:NEXT
BS 570 POKEV+1,Y(0):POKEV+3,Y(1):POKEV+5,Y(2):BC=P
      EEK(V+31)
PC 580 POKEW2,17:POKEF2,Y(0):POKEF2,Y(1):POKEF2,Y(
      2):POKEW2,0
SC 590 FORD=0TO(L-1)*100:NEXT
AM 600 GETN$:IF(N$>"0"ANDN$<="9")ORN$="-"ORN$=CHR
      $(20)ORN$=CHR$(13)THEN700
AS 610 BC=PEEK(V+31)AND7:IFBC=0THEN560
CC 620 IF(BCAND1)=1THENSN=0:GOSUB660
AG 630 IF(BCAND2)=2THENSN=1:GOSUB660
JQ 640 IF(BCAND4)=4THENSN=2:GOSUB660
XX 650 GOTO560
EX 660 BH=BH+1:POKEV+1+SN*2,221:POKE2040+SN,250:PO
      KEW,0:POKEAD,12:POKEF,1

```

```

RB 670 POKEW,129:FORD=1TO100:NEXT:POKEW,0:POKEW,12
      9:POKE2040+SN,251:FORD=1TO100
PB 680 NEXT:GOSUB1080:PRINTTB$;TAB(36)BH;:Y(SN)=30
      :IFBH>2THEN890
EJ 690 POKE2040+SN,253+SN:RETURN
MD 700 IFN$=CHR$(20)ANDTB>0THENTB=TB-1:PRINT"
      {2 LEFT}{P}{LEFT}";:AN$=LEFT$(AN$,LEN(AN$)-
      1)
XD 710 IFN$=CHR$(20)THEN610
JG 720 IFN$<>"- "THEN750
XQ 730 IFTB=0THENTB=1:AN$="- ":PRINTLEFT$(TB$,33)"-
      {P}{LEFT}";
ED 740 GOTO610
JD 750 IFN$=CHR$(13)THEN780
DG 760 IFTB<4THENPRINTLEFT$(TB$,33+TB);N$;"{P}
      {LEFT}";:TB=TB+1:AN$=AN$+N$
JF 770 GOTO610
KD 780 AN=VAL(AN$):PRINTD$;"{8 RIGHT}{P}{6 SPACES}
      ";:AN$="":TB=0
XP 790 CF=0:FORXX=0TO2:IFA(XX)=ANANDY(XX)<221THENG
      OSUB860
AA 800 NEXT:IFTC>=20THEN1000
PJ 810 IFCF>0THEN850
QB 820 WT=WT+1:FORX=0TO2:Y(X)=Y(X)+(3-L)*10:IFY(X)
      >200THENY(X)=200
RM 830 NEXT:L=L-1:POKEW,0:POKEAD,0:POKESR,240
AP 840 POKEW,17:FORD=255TO100STEP-2:POKEF,D:NEXT:P
      OKEW,0:POKESR,0
FJ 850 GOTO610
BH 860 TC=TC+1:AT=AT+1:POKE2040+XX,252:SN=XX:POKEW
      ,0:POKEAD,144:POKEF,250
EG 870 POKEW,129:FORD=1TO100:NEXT:POKE2040+XX,253+
      XX:GOSUB1080:Y(XX)=30:CF=1
AR 880 BC=PEEK(V+31):RETURN
HD 890 POKEV+1,0:POKEV+3,0:POKEV+5,0
DB 900 PRINT"{CLR}{3 DOWN}{5 RIGHT}{RED}MATH PROBL
      EMS INVADE THE CITY":GOTO930
MH 910 PRINT"{CLR}{3 DOWN}{5 RIGHT}{RED}YOU'VE BEA
      TEN THE MATH PROBLEMS"
RK 920 PRINT"{2 DOWN}{3 RIGHT}{9 RIGHT}AT THEIR O
      WN GAME"
CA 930 PRINT"{2 DOWN}{3 RIGHT}{6}GAME RESULTS:":PR
      INT"{DOWN}{4 RIGHT}{GRN}RIGHT ANSWERS:"AT
MH 940 PRINT"{DOWN}{6 RIGHT}{PUR}WRONG ANSWERS:"WT
XQ 950 PRINT"{DOWN}{8 RIGHT}{1}BOMBS REACHING THE
      {SPACE}CITY:"BH
GP 960 PRINT"{3 DOWN}{7}{3 RIGHT}DO YOU WANT ANOTH
      ER CHANCE? (Y/N)?"
HX 970 GETC$:IFC$<>"Y"ANDC$<>"N"THEN970

```

```

MG 980 IFC$="Y"THENRUN
RP 990 PRINT "{CLR}{7}" ; :POKE53280,14:POKE53281,6:P
      OKEV+21,0:END
BS 1000 FORX=0TO2:Y(X)=0:POKEV+1+2*X,0:NEXT:IFL1+L
      2>1000THEN910
HK 1010 PRINT "{HOME}{3 DOWN}{6 RIGHT}{PUR}"TC"CORR
      ECT ANSWERS":TC=0
MM 1020 PRINT "{3}{DOWN}{3 RIGHT}YOU MAY GO ON . .
      {SPACE} . ."
GX 1030 PRINT "{DOWN}{4 RIGHT}{17 SPACES}TO THE NEX
      T LEVEL"
RD 1040 L1$=STR$(L1):L2$=STR$(L2):IFL2<L1THENL2$=L
      2$+"9":GOTO1060
GK 1050 L1$=L1$+"9"
QR 1060 L1=VAL(L1$):L2=VAL(L2$):PRINT "{2 DOWN}
      {4 RIGHT}{YEL}YOUR NEW LIMITS ARE: "L1"AND
      "L2
EA 1070 GOTO530
CM 1080 U$=STR$(INT(RND(0)*L1)):B$=STR$(INT(RND(0)
      *L2)):B$=RIGHT$(B$,2)
HP 1090 IFT$="X"THENA(SN)=VAL(U$)*VAL(B$)
KK 1100 IFT$="+"THENA(SN)=VAL(U$)+VAL(B$)
JD 1110 IFT$="-"THENA(SN)=VAL(U$)-VAL(B$)
BE 1120 B$=T$+B$:U$=RIGHT$(U$,3):SY=SN*2+1+V:IFLEN
      (U$)=2THENU$=" "+U$
QK 1130 SS=(SN+253)*64:POKEW2,0:POKEW2,65:FORY=0TO
      2:C1$=MID$(U$,Y+1,1)
XG 1140 FORX=0TO7:POKESS+X*3+Y,N(VAL(C1$),X):IFC1$
      =" "THENPOKESS+X*3+Y,0
AA 1150 POKESY,ABS(PEEK(SY)-3):POKEF2,100-10*X:NEX
      TX,Y
XM 1160 FORY=0TO2:C1$=MID$(B$,Y+1,1):FORX=8TO15
AH 1170 POKESS+X*3+Y,N(VAL(C1$),X-8):POKEF2,10*X:P
      OKESY,ABS(PEEK(SY)-3)
FD 1180 IFC1$=" "THENPOKESS+X*3+Y,0
XJ 1190 IFC1$=T$THENPOKESS+X*3+Y,N(10,X-8)
KP 1200 NEXTX,Y:POKEW2,0
XE 1210 POKESS+48,255:POKESS+49,255:POKESS+50,255
XB 1220 FORX=SS+51TOSS+62:POKEX,0:NEXT:RETURN

```




Chapter 3

For Home





The Construction Set

Fred Karg

This unique program has the fun elements of a game, but it's not a game. A creative toy? Maybe. If that sounds too vague, it could also be called a tool for computer-aided design. Whatever you call it, you'll find it's a lot of fun for children—and adults. Joystick required.

All our lives we build things, often just for the fun of it. Toddlers play with building blocks. Grade-schoolers create elaborate structures with Tinker Toys, Lego blocks, and Erector sets. Teenagers put together model cars and planes. Adults have basement workshops for making things out of wood. Even the process of writing a computer program can be described as building something from parts.

“The Construction Set” is unlike most other computer programs. It can hold up to four different sets of building blocks which can be combined easily in any way you like. No one wins or loses and there's no high score. But it's great fun to create your own interesting picture.

Special Typing Instructions

The program listings include a BASIC program and three programs in MLX format. Type in Program 1 (The Construction Set Main Program) first. Datassette owners should change **DI=8** to **DI=1** in line 160.

The three MLX listings (Programs 2–4) are not really machine language programs—they're character sets for three different sets of building blocks. Follow these directions to create each of the character sets.

1. Load and run the “MLX” entry program (found in Appendix D). If you haven't used it before, be sure to read MLX's directions carefully.
2. Answer the address prompts with the following values:

Starting Address: 7000

Ending Address: 77FF

4. Type in Program 2, 3, or 4. When you've finished entering all the data be sure to save a copy. For Program 1 to work properly, you must use the following names for the character set files: HOUSE for the data from Program 2, CREATURE for the data from Program 3, and SHAPES for the data from Program 4.
5. Repeat these steps for each of the character sets you wish to use.

You don't have to type in all of the character sets, but you need at least one for The Construction Set to work properly.

Line 170 of the main program keeps track of how many character sets will be loaded and what their names are. Currently, it's set up for 3. If you enter only one or two sets of building blocks, change the 3 to a 2 or 1 and follow it with a comma and the names you used to save the character set files.

Tape users should be especially careful with line 170. Save all of the character sets to the same tape, one after another. Remember which one was saved first and make sure that filename is first in line 170. For example, if you typed in and saved the first two sets, and saved them as HOUSE and MONSTERS, line 170 of the main program should read:

170 DATA 2,"HOUSE","MONSTERS"

Remember to modify line 160 for tape as well.

Putting Together a Picture

When you first load and run The Construction Set, you'll see the menu of commands onscreen while the program loads the character sets.

Any time you want to check the menu, press f7 and the list of commands will appear (press f7 to go back to the work area).

The work screen is divided into two parts. At the bottom you should see several building blocks. A pointing-hand symbol floats above these blocks. Use the cursor keys (left and right) to move the hand back and forth to select the block you want to place on the screen. Press the plus and minus keys to see more building blocks in the same set. Some blocks are full-size; others are shorter.

The top part of the screen is the workspace. At first the screen is blank, except for a window (a hollow rectangle). With

a joystick in port 2, move the window around the screen. Press the joystick fire button to place a building block on the screen in the same position as the rectangle. Whichever block the hand points to at the bottom of the screen is the one stamped on the screen.

Tap the INST/DEL key to erase whatever's within the window.

The process of creating a design is fairly easy. First move to the set of blocks you want with the plus and minus keys; then use the cursor keys to move the hand to the pattern you want. After selecting a block, position the window and press the fire button. Patterns can be repeated as often as you like.

If you forget the commands, press f7 to see the menu. If you want to start over, clear the screen by pressing the SHIFT-CLR/HOME.

When you've finished, you can experiment with different colors for different parts of the screen. Press the left-arrow key (right above the CTRL key) and then one of the following keys:

- (←) f1 **Change border color**
- (←) f3 **Change screen color**
- (←) f5 **Change character colors**

Four Character Sets, Four Screens

The Construction Set has room for four different character sets. Each defines a set of building blocks. The three accompanying the character sets are "House" (Program 2) for making brick buildings, "Creature" (Program 3) for putting together people (and monsters), and "Shapes" (Program 4) for abstract patterns. These custom character sets were created with "Ultrafont +" from the July 1984 issue of *COMPUTE!'s Gazette*. You can create your own Construction Set building blocks if you have a copy of Ultrafont +. You can invent building blocks for electronic circuits, quilts, landscapes, flags, or a variety of other patterns.

Each set of building blocks has its own screen. You can travel between the workspaces of each of the character sets by pressing f1 (remember that if you press the left-arrow key first, f1 changes the border color). The pictures you've created are preserved when you switch back and forth. Press f1 four times, and you'll come back to the screen where you started, with the picture intact.

An interesting modification for children is to load the same set of building blocks into all four workspaces. For example, you could change line 170 to **170 DATA 4, "CREATURE", "CREATURE", "CREATURE", "CREATURE"** (tape users will have to use MLX to save the Creature font four times in a row on a single tape). Then have the child make up a story and build four pictures to illustrate the plot. When the characters have been put together, use f1 to step through the pictures as the child tells the story.

Program 1. The Construction Set—Main Program

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

```

DD 100 POKE53280,1:POKE53281,1
CQ 110 PRINT"{CLR}{10 DOWN}{9 SPACES}{20 @}"
DJ 120 PRINT"{9 SPACES}{RVS}THE CONSTRUCTION SET":
    A$="{2 SPACES}{RED}{8 @}"
HP 130 PRINT"{7 DOWN}{12 SPACES}PLEASE WAIT..."
JX 140 POKE55,0:POKE56,64:CLR:BA=16384:READA:FORI=
    1TOA:READX$:NEXT:FORI=1TO352
HX 150 READX$:S=S+X:NEXT:IFS<>34546THENPRINT"ERROR
    {SPACE}IN DATA STATEMENTS.":STOP
XF 160 RESTORE:DI=8:READNS:REM SET DI=1 FOR TAPE
PP 170 DATA 3,"HOUSE","CREATURE","SHAPES"
FJ 180 PRINT"{UP}{14 SPACES}LOADING...{2 SPACES}":
    FORN=0TONS-1:CHSET=BA+6144+N*2048:READX$
JE 190 GOSUB1200:NEXT:PRINT"{CLR}{RVS}{GRN}":FORN=
    1TO40:PRINT"{Y}";:NEXT
GE 200 PRINTA$:PRINT"{2 SPACES}{RVS}{RED}
    {3 SPACES}F1{3 SPACES}{OFF}{BLU}{9 SPACES}F
    OR DIFFERENT PICTURE";
QJ 210 PRINTA$:PRINT"{2 SPACES}{RVS}{RED}
    {3 SPACES}F7{3 SPACES}{OFF}{BLU}{9 SPACES}F
    OR INSTRUCTIONS"
MK 220 PRINTA$:PRINT"{2 SPACES}{RVS}{RED}+ AND -
    {OFF}{BLU}{9 SPACES}FOR EXTRA PIECES"
BR 230 PRINTA$:PRINT"{2 SPACES}{RVS}{RED} CURSOR
    {BLU}{OFF}{9 SPACES}SELECTS PIECE"
KG 240 PRINT"{2 SPACES}{RED}{10 @}":PRINT"
    {2 SPACES}{RVS}{RED} JOYSTICK {BLU}{OFF}
    {7 SPACES}POSITIONS PIECE"
AC 250 PRINT"{2 SPACES}{RED}{13 @}":PRINT"
    {2 SPACES}{RVS}{RED} FIRE BUTTON {OFF}{BLU}
    {4 SPACES}STAMPS PIECE"
HF 260 PRINT"{2 SPACES}{RED}{10 @}":PRINT"
    {2 SPACES}{RVS}{RED} INST DEL {OFF}{BLU}
    {7 SPACES}ERASES PIECE"
AS 270 PRINT"{2 SPACES}{RED}{16 @}":PRINT"

```

```

      {2 SPACES}{RVS}{RED} SHIFT CLR HOME {OFF}
      {BLU} ERASES SCREEN"
GH 280 PRINT"{2 SPACES}{RED}[38 @]";
EK 290 PRINT"{2 SPACES}{RVS}{RED}PUSH ← AND THEN F
      1, F3, F5{12 SPACES}"
MA 300 PRINT"{2 SPACES}{BLU}TO CHANGE BORDER, SCRE
      EN, & PICTURE{5 SPACES}COLOR. F7 RESUMES PR
      OGRAM"
BP 310 PRINT"{RVS}{GRN}":FORN=1TO40:PRINT"[Y]";NE
      XT
HR 320 PRINT"[1]{9 SPACES}USE PORT 2 FOR JOYSTICK.
      ";
ME 330 V=53248:SB=15360+BA:SC=BA:H1=33:CB=1:MM=254
      :MN=-1
MQ 340 SE=8:FORSP=0TO2:FORN=0TO63:READD:POKESB+N+(
      64*SP),D:NEXT:NEXT
BP 350 FORA=53088TO53247:READI:POKEA,I:NEXT:SYS530
      88:POKE17400,241:POKE17401,240
AB 360 POKE17402,242:POKEV+39,1:POKEV+3,205:POKEV+
      2,24:POKEV+40,10:POKEV+28,1
DS 370 POKEV+37,5:POKEV+38,1:POKEV+28,PEEK(V+28)OR
      4:POKEV+41,1
BA 380 DIMA$(80,3):X=1:FORC=1TO20
KD 390 CO=CO+1:FORR=1TO3:A$(C,R)=STR$(X):X=X+32:NE
      XT:X=X-93:IFCO=10THENCO=0:X=X+66
XX 400 NEXTC:CO=0:FORC=21TO50:CO=CO+1:A$(C,1)=STR$(
      X):A$(C,2)="256":A$(C,3)="256"
QK 410 X=X+3:IFCO=10THENCO=0:X=X+2
SB 420 NEXT:POKE53265,PEEK(53265)AND239:POKE56578,
      PEEK(56578)OR3
GQ 430 POKE56576,(PEEK(56576)AND252)OR2:A=4:SK=-16
      :POKE53270,PEEK(53270)OR16:PY=1
RF 440 FORN=1TONS:GOSUB690:POKE53281,15:PRINT"
      {CLR}":POKE53281,0:NEXT:PY=0:A=4:SK=-16
PR 450 POKE53282,2:POKE53283,1:GOSUB690:POKEV+21,3
      :POKEV,24:POKEV+1,50
JA 460 POKEV+27,0:POKE53265,PEEK(53265)OR16:X=24:Y
      =50
QE 470 J2=PEEK(56320):FR=J2AND16:JV=15-(J2AND15):I
      FJV=1THENY=Y-SE:IFY<50THENY=50
MC 480 IFJV=2THENY=Y+SE:IFY>194THENY=194
KQ 490 IFJV=4THENX=X-SE:IFX<24THENX=24
SD 500 IFJV=8THENX=X+SE:IFX>312THENX=312
HE 510 TR=(X>255):POKEV+16,(PEEK(V+16)AND(MM))ORTR
      *(MN):POKEV+SP*2,X+(TR*255)
XB 520 POKEV+(SP*2)+1,Y+2:IFFR=0THEN640
GP 530 GETKH$:IFKH$=""THEN470
XM 540 IFVAL(KH$)<4ANDVAL(KH$)>0THENSE=8*VAL(KH$):
      GOTO470

```

```

BR 550 IFKH$="-"THENCB=CB-9:IFCB<0THENCB=37
FF 560 IFKH$="+"THENCB=CB+9:IFCB>37THENCB=1
FG 570 IFKH$="+"ORKH$="-"THENGOSUB730:GOTO530
RB 580 IFKH$="{RIGHT}"ORKH$="{LEFT}"THEN GOSUB780:
      GOTO470
BD 590 IFASC(KH$)=20THEN640
JS 600 IFKH$="{CLR}"THENPOKE53281,15:PRINT"{CLR}":
      POKE53281,0:GOSUB730:GOTO470
RR 610 IFKH$="{F1}"THENGOSUB690
XP 620 IFKH$="{F7}"THENGOSUB1260
GR 630 GOTO470
MF 640 X1=(X-24)/8:Y1=((Y-50)/8)*40:S=SC+X1+Y1:FOR
      R=1TO3:FORC=0TO2
FP 650 P=VAL(AS$(CB+CX,R))+C:IFKH$<>"THEN:IFASC(KH
      $)=20THENP=32
QM 660 IFP>255THENP=32
BQ 670 POKES+C+(R-1)*40,P:NEXT:IFVAL(AS$(CB+CX,1))+
      C>185THENR=3
KK 680 NEXT:FR=1:GOTO470
CK 690 A=A+2:SK=SK+16:IFA>12OR(A-6)/2=NSTHENA=6:SK
      =0
MR 700 SC=BA+1024*SK/16:POKE53272,(PEEK(53272)AND1
      5)ORSK
CR 710 POKE53272,(PEEK(53272)AND240)ORA:POKE648,SC
      /256:IFPY=1THENRETURN
KG 720 POKESC+1016,241:POKESC+1017,240:POKESC+1018
      ,242:GOSUB730:RETURN
MQ 730 VA=0:CO=0:FORC=1TO9:CO=CO+1:FORR=1TO3:FORN=
      1TO3:P=VAL(AS$(CB+C-1,R))-1+N
DS 740 IFP>255THENP=32
JM 750 POKESC+880+VA+N+(C-1)*4+(R-1)*40,P:NEXT:IFV
      AL(AS$(CB+C-1,R))-1+N>185THENNN=3
KS 760 NEXT:IFCO=9THENCO=0:VA=VA+120
EX 770 NEXT:GOSUB800:RETURN
SC 780 IFKH$="{LEFT}"THEN H1=H1-32:CX=CX-1:IFCX<0T
      HENCX=8:H1=288
BB 790 IFKH$="{RIGHT}"THEN H1=H1+32:CX=CX+1:IFCX>8
      THENCX=0:H1=40
XM 800 IFVAL(AS$(CB+CX,1))+C>185THENSP=2:SV=-16:POK
      EV+21,6:MM=251:MN=-4
HQ 810 IFVAL(AS$(CB+CX,1))+C<189THENSP=0:SV=0:POKEV
      +21,3:MM=254:MN=-1
EM 820 TZ=(H1>255):POKEV+16,(PEEK(V+16)AND(253))OR
      TZ*(-2):POKEV+2,H1+(TZ*255)
GP 830 RETURN
EH 840 DATA 3,252,0,7,254,0,15,254,0,31
EP 850 DATA 110,0,31,247,0,29,255,0,55,253
DS 860 DATA 128,61,191,128,59,223,128,50,238,128
EB 870 DATA 115,187,128,99,187,128,3,185,0,3,144,0
JK 880 DATA2,128,0,3,128,0,3,128,0,2

```



```

PD 890 DATA128,0,1,0,0,0,0,0,0,0
QA 900 DATA0,0,85,85,85,127,255,253,112,0
RC 910 DATA13,112,0,13,112,0,13,112,0,13
GE 920 DATA112,0,13,112,0,13,112,0,13,112
SS 930 DATA0,13,112,0,13,112,0,13
SG 940 DATA112,0,13,112,0,13,112,0,13,112
PK 950 DATA0,13,112,0,13,112,0,13,112,0
FH 960 DATA13,127,255,253,85,85,85,247,85,85
EB 970 DATA85,106,170,169,96,0,9,96,0,9
JP 980 DATA 96,0,9,96,0,9,96,0,9,106
DK 990 DATA170,169,85,85,85,0,0,0,0,0
FQ 1000 DATA0,0,0,0,0,0,0,0,0,0
GR 1010 DATA 0,0,0,0,0,0,0,0,0,0
DS 1020 DATA0,0,0,0,0,0,0,0,0,0
DG 1030 DATA 0,0,120,173,40,3,141,248,207,173
QA 1040 DATA41,3,141,249,207,173,250,207,141,40
EM 1050 DATA3,173,251,207,141,41,3,88,96,165
CM 1060 DATA145,201,253,208,116,120,169,0,133,198
XJ 1070 DATA32,159,255,32,228,255,201,0,240,246
XQ 1080 DATA201,133,240,15,201,134,240,23,201,135
SK 1090 DATA240,31,201,136,240,84,76,130,207,173
AK 1100 DATA32,208,24,105,1,141,32,208,76,130
RJ 1110 DATA 207,173,33,208,24,105,1,141,33,208
BH 1120 DATA 76,130,207,165,243,141,252,207,165,24
4
PH 1130 DATA 141,253,207,169,0,133,243,169,216,133
RS 1140 DATA 244,173,134,2,24,105,1,141,134,2
CF 1150 DATA 162,4,160,0,145,243,136,208,251,202
GX 1160 DATA 240,5,230,244,76,216,207,173,252,207
SP 1170 DATA 133,243,173,253,207,133,244,76,130,20
7
CS 1180 DATA 88,108,248,207,237,246,123,207,2,0
GK 1190 DATA 255,255
AA 1200 FORI=1TOLN(X$):POKE678+I,ASC(MID$(X$,I)):
NEXT
JM 1210 POKE781,167:POKE782,2:POKE780,LEN(X$):SYS6
5469
CX 1220 POKE780,1:POKE781,DI:POKE782,0:SYS65466
MK 1230 POKE780,0:POKE781,0:POKE782,CHSET/256:SYS6
5493
MB 1240 IF (PEEK (783)AND1)OR(191ANDST)THENPRINT"LOA
D ERROR":STOP
RJ 1250 CLOSE1:RETURN
FG 1260 POKE56578,PEEK(56578)OR3:POKE 56576,(PEEK(
56576)AND252)OR3
FR 1270 POKE53270,PEEK(53270)AND239:POKE53272,(PEE
K(53272)AND15)OR16
EX 1280 POKE53272,(PEEK(53272)AND240)OR4:XX=PEEK(V
+21.):POKEV+21,0:POKE53281,0
EH 1290 GETK$:IFK$=""THEN1290

```

```

JA 1300 POKE56578,PEEK(56578)OR3:POKE56576,(PEEK(5
6576)AND252)OR2
XF 1310 POKE53270,PEEK(53270)OR16:POKE53272,(PEEK(
53272)AND15)ORSK
BE 1320 POKE53272,(PEEK(53272)AND240)ORA:POKE53281
,0:POKEV+21,XX:RETURN

```

Program 2. The Construction Set—House Character Set

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```

7000:00 00 00 00 00 00 00 00 E0
7008:00 00 00 03 03 00 00 00 31
7010:3C 3C 3C FF FF 3C 3C 3C 4B
7018:00 00 00 C0 C0 00 00 00 0B
7020:51 55 51 45 51 45 54 45 0B
7028:14 55 45 14 55 15 51 15 09
7030:51 45 51 45 11 55 45 11 03
7038:11 45 6A 60 60 60 6A 60 FF
7040:45 05 AA 28 28 28 AA 28 3C
7048:11 45 A9 09 09 09 A9 09 91
7050:11 54 55 45 55 15 5A 5B DC
7058:10 55 15 55 69 BE EB AA 57
7060:51 54 15 55 11 55 A5 E5 06
7068:15 14 55 51 55 15 55 54 96
7070:04 55 15 55 55 51 55 45 80
7078:41 55 14 55 55 14 55 15 E1
7080:11 54 55 15 5A 18 58 58 37
7088:51 51 55 55 AA 28 28 28 D4
7090:45 44 55 51 A5 25 25 24 15
7098:00 00 00 00 00 00 00 00 79
70A0:00 00 00 00 00 00 00 00 81
70A8:00 00 00 00 03 0F 33 FF 44
70B0:28 14 14 14 FF FF FF 3F AD
70B8:00 00 00 00 FF 3F FF FF 96
70C0:00 00 00 00 FF 3F FF F3 92
70C8:00 00 00 00 F0 3C FF FF 22
70D0:00 00 00 00 00 00 00 C0 72
70D8:00 00 00 00 00 00 00 00 B9
70E0:54 45 55 16 5A 5B 58 1B 55
70E8:55 61 A9 8A CE CF 00 CF 30
70F0:45 55 54 55 91 95 95 85 3D
70F8:00 00 00 00 00 00 00 00 D9
7100:00 00 00 00 00 00 00 00 E2
7108:00 00 00 00 00 01 06 1B 16
7110:3C 3C 3C 3C 55 AA 33 CF F6
7118:00 00 00 00 00 40 90 E4 02
7120:55 51 55 45 51 54 45 51 B8
7128:44 51 15 54 15 54 45 54 42
7130:55 51 45 55 11 55 45 55 CD

```

```

7138:60 60 6A 55 55 6A 60 60 7B
7140:28 28 AA 55 55 AA 28 28 B9
7148:09 09 A9 55 55 A9 09 09 E8
7150:5B 5B 5B 5B 58 5B 5B 1B
7158:FF FF F7 F7 37 37 FF FF 50
7160:E5 25 E5 E5 E5 E5 E5 13
7168:15 54 55 15 5A 18 58 58 23
7170:55 51 55 55 AA 28 28 28 C0
7178:55 44 55 51 A5 25 25 24 07
7180:18 58 58 18 5A 55 15 55 B9
7188:28 28 28 28 AA 55 15 55 3B
7190:25 25 25 25 A4 55 14 55 3E
7198:00 00 00 00 03 0F AA 56 7B
71A0:03 0F 3C FF 33 FF AA 95 D4
71A8:FC CF FC FF 33 FF AA 55 E1
71B0:FC FF F0 FF FF 3F AA 45 C7
71B8:FC FF F3 FF CF FF AA 54 C0
71C0:FC FF F3 FF CF FF AA 54 C8
71C8:FC FF F3 FF CF FF AA 54 D0
71D0:F0 FC F3 FF CF FF AA 54 12
71D8:00 00 00 C0 30 FC AA 69 FB
71E0:5B 5B 59 49 59 59 1A 59 C5
71E8:CF CF 65 65 65 65 AA 65 26
71F0:95 91 95 95 94 95 85 95 AA
71F8:00 00 00 00 00 00 00 00 DB
7200:00 00 00 00 00 00 00 01 E5
7208:5B 5B 5B 5B 58 5B 5A 55 CC
7210:C3 C3 00 00 00 F3 AA 55 42
7218:E5 E5 E5 E5 25 E5 A5 55 E5
7220:55 51 45 54 45 50 55 55 5D
7228:05 51 45 54 45 55 45 55 31
7230:51 55 11 55 45 51 51 55 F1
7238:60 60 60 60 60 60 6A 55 26
7240:28 28 28 28 28 28 AA 55 57
7248:09 09 09 09 09 09 A9 55 BA
7250:5B 5B 5B 5B 59 5A 6A AA 8E
7258:FF FF FF FF AA AA AA 3D
7260:E5 25 E5 E5 65 A5 A9 AA 5C
7268:18 58 58 18 5A 55 15 55 A3
7270:28 28 28 28 AA 55 15 55 25
7278:25 25 25 25 A4 55 14 55 28
7280:50 55 15 55 55 51 55 45 BA
7288:54 55 15 55 55 51 55 45 C4
7290:55 55 15 55 55 51 55 45 4D
7298:56 55 55 55 54 55 55 55 F5
72A0:94 95 95 94 95 95 91 95 EC
72A8:55 51 55 55 55 45 54 55 4A
72B0:55 51 55 55 55 45 54 55 52
72B8:55 51 55 55 55 45 54 55 5A
72C0:55 51 55 55 55 45 54 55 62

```

```

72C8:55 51 55 55 55 45 54 55 6A
72D0:55 51 55 55 55 45 54 55 72
72D8:69 65 65 65 65 65 65 65 BF
72E0:59 59 59 59 5A 6A 55 55 06
72E8:65 65 65 65 AA AA 55 55 DC
72F0:95 95 95 95 95 A5 55 55 55
72F8:00 00 00 00 00 00 00 00 DD
7300:00 00 00 00 00 00 00 00 E6
7308:00 00 00 00 00 00 00 00 EE
7310:00 00 00 00 00 00 00 00 F6
7318:02 01 09 05 26 1B 9B 56 DC
7320:80 40 60 50 98 E4 E6 95 24
7328:00 00 00 00 00 00 00 00 0F
7330:00 00 00 00 00 00 00 00 17
7338:00 00 00 00 00 00 00 00 1F
7340:02 03 0B 0F 2F 3F BE F9 29
7348:AA FF FF FF FF FF BF 6F 73
7350:AA FF FF FF FF FF FF FF 8C
7358:AA FF FF FF FF FF FA E5 70
7360:AA FF FF FF FF FF FF BF 5C
7368:AA FF FF FF FF FF FE F9 9C
7370:80 C0 E0 F0 F8 FC BE 6F 9B
7378:00 00 00 00 00 00 00 00 5F
7380:95 95 95 95 95 95 95 95 67
7388:55 50 50 50 55 55 55 55 3D
7390:56 56 56 56 56 56 56 56 77
7398:00 00 00 00 00 00 00 00 7F
73A0:00 00 00 00 00 00 00 00 87
73A8:82 69 55 55 55 55 55 55 2B
73B0:00 00 00 00 00 00 00 00 97
73B8:A8 2F 2F 2F 20 20 20 20 7A
73C0:00 00 C0 F0 FC 0F 03 00 F8
73C8:15 56 5A 6A 5B 5B 5B 5B 1C
73D0:AA AA AA AA FF FF C3 C3 03
73D8:56 95 A5 A9 E5 E5 E5 A5 D7
73E0:A8 54 54 54 54 54 54 54 F1
73E8:02 01 01 01 01 01 01 01 50
73F0:A8 54 54 54 54 54 54 54 02
73F8:00 00 00 00 00 00 00 00 DF
7400:00 00 00 00 00 00 00 00 E8
7408:00 00 00 00 00 00 00 00 F0
7410:02 01 09 01 25 15 95 56 6A
7418:55 55 56 56 5B 6F BF D5 1F
7420:55 55 95 95 E5 F9 FE 57 81
7428:80 40 60 50 58 54 56 95 C8
7430:00 00 00 00 00 00 00 00 19
7438:02 03 0B 0F 2F 3F BF AA D5
7440:F9 F9 F9 FA EA FF FF AA 95
7448:6F 6F 6F AF AB FF FF AA B5
7450:FF FF FF FF FF FF FF AA E3

```

7458:E5 E5 E5 EA AA FF FF AA 19
7460:BF BF BF BF AF FF FF AA 35
7468:F9 F9 F9 FA EA FF FF AA BD
7470:6F 6F 6F AF AB FF FF AA DD
7478:80 C0 E0 F0 F8 FC FE AA 61
7480:95 95 95 95 95 95 95 95 69
7488:05 05 05 55 55 55 55 50 26
7490:56 56 56 56 56 56 56 56 79
7498:00 00 00 00 00 00 00 00 81
74A0:00 00 00 00 00 00 00 00 89
74A8:45 45 45 55 55 55 55 51 7F
74B0:00 00 02 01 01 01 01 01 F8
74B8:20 20 22 A9 A9 65 55 55 7B
74C0:00 00 00 00 00 00 00 00 A9
74C8:5B 5B 5B 5B 5B 5B 5B 5B A5
74D0:C3 C3 C3 FF FF 3F 3F FF 80
74D8:E5 E5 E5 E5 E5 E5 E5 E5 C1
74E0:54 56 55 55 54 54 54 54 7A
74E8:01 A9 55 55 55 55 55 55 BC
74F0:54 54 54 54 54 54 54 14 99
74F8:00 00 00 00 00 00 00 00 E1
7500:00 00 00 00 00 00 00 00 EA
7508:02 01 09 06 2B AF AF AA D7
7510:5B 6D BD FD FD FF FF AA 36
7518:6A 68 68 60 60 55 FF AA 68
7520:A9 29 29 09 09 55 FF AA 28
7528:E5 79 7E 7F 7F FF FF AA D2
7530:80 40 60 90 E8 F8 FE AA 54
7538:55 55 56 59 65 95 55 AA 5A
7540:65 99 56 55 55 55 55 AA B9
7548:55 55 55 95 65 59 56 AA 1F
7550:55 55 56 59 65 95 55 AA 72
7558:65 99 56 55 55 55 55 AA D1
7560:55 55 55 95 65 59 56 AA 37
7568:55 55 56 59 65 95 55 AA 8A
7570:65 99 56 55 55 55 55 AA E9
7578:55 55 55 95 65 59 56 AA 4F
7580:95 95 95 95 95 95 95 95 6B
7588:50 50 50 55 55 55 55 55 0F
7590:56 56 56 56 56 56 56 56 7B
7598:A0 50 50 55 55 55 55 55 47
75A0:A0 50 50 55 55 55 55 55 4F
75A8:51 51 51 55 55 55 55 55 10
75B0:01 01 01 01 01 01 01 01 9B
75B8:55 55 55 55 55 55 55 55 A3
75C0:00 00 00 00 00 00 00 00 AB
75C8:5B 5B 5B 5B 5A 5A 69 A5 0E
75D0:FF FF FF FF 96 55 69 AA 43
75D8:E5 A5 E5 E5 A5 A5 69 5A 2C
75E0:44 45 45 45 45 45 55 55 7B

```

75E8:55 55 55 55 55 55 55 55 D3
75F0:14 14 14 14 54 54 54 54 9F
75F8:00 00 00 00 00 00 00 00 E3
7600:00 00 00 00 00 00 00 00 EC
7608:88 88 AA 88 88 AA 88 88 C1
7610:88 88 AA 88 88 AA 88 88 C9
7618:88 88 AA 88 88 AA 88 88 D1
7620:BC BF 8F BF BF 8F BF 8F 94
7628:00 C3 EB EB EB 00 EB EB 65
7630:3E FE F2 FE FE F2 FE FE 0B
7638:AA BF BF BF 82 80 80 AA 20
7640:AA EB EB E8 28 28 28 AA 66
7648:AA FE FE 22 02 02 02 AA 13
7650:55 55 56 59 65 95 95 AA F4
7658:69 96 55 7D FF 7D 55 AA 6D
7660:55 55 95 65 59 56 56 AA D1
7668:AA 95 95 95 B4 9C 90 AA FF
7670:AA 55 41 00 00 00 00 AA DA
7678:AA 56 56 56 1E 36 06 AA 01
7680:02 01 01 A9 55 55 55 55 69
7688:A0 50 50 5A 55 55 55 55 89
7690:2A 15 15 95 55 55 55 55 D3
7698:55 15 55 54 55 15 55 55 64
76A0:55 15 55 54 55 15 55 55 6C
76A8:55 15 55 54 55 15 55 55 74
76B0:AA AA A8 AA 2A A8 AA AA 51
76B8:AA AA A8 AA 2A A8 AA AA 59
76C0:AA 2A A8 AA AA AA 8A AA 0D
76C8:FC FF FF FF 3F FF FF F3 22
76D0:FF FF FF FF 3F FF FF F3 AB
76D8:FF F3 FF FF 3F FF FF F3 B0
76E0:55 56 5B 6F 6F BF BF BF 2A
76E8:A9 FE FF FF FF FF FF FF 6A
76F0:54 54 94 E4 E4 F8 F8 F8 F3
76F8:00 00 00 00 00 00 00 00 E5
7700:00 00 00 00 00 00 00 00 EE
7708:00 00 03 0F 3F FF AA 55 EC
7710:3C FF FF FF FF FF AA 55 C7
7718:00 00 C0 F0 FC FF AA 55 C0
7720:A0 A0 AA A5 A5 A6 A8 A0 F0
7728:00 00 AA 69 82 00 00 00 17
7730:0A 0A AA 5A 5A 9A 2A 0A 3D
7738:A0 A0 A0 A0 A0 A0 A0 A0 27
7740:00 00 00 00 00 00 00 00 2F
7748:0A 0A 0A 0A 0A 0A 0A 0A 37
7750:00 00 00 00 00 00 00 00 3F
7758:55 55 55 55 55 55 55 55 47
7760:40 40 40 40 40 40 40 40 4F
7768:AA 80 80 80 80 80 80 AA 96
7770:AA 00 00 00 00 00 00 AA 5F

```

```

7778:AA 02 02 02 02 02 02 02 AA 64
7780:00 00 00 00 0A AA A5 55 0B
7788:00 00 0A AA A5 55 51 55 DD
7790:0A AA A5 51 55 45 55 55 B8
7798:A0 AA 5A 55 15 51 55 55 11
77A0:00 00 A0 AA 5A 55 15 55 F5
77A8:00 00 00 00 A0 AA 5A 15 11
77B0:00 00 00 02 01 0B 27 BD FF
77B8:28 BE 7D D7 FF 7D D7 FF 3E
77C0:00 00 00 80 40 E0 D8 7E 6D
77C8:02 00 00 00 00 00 00 00 B8
77D0:A0 BF BD BD 83 80 80 80 33
77D8:00 00 F0 F0 F0 00 00 00 7C
77E0:BF BF BF 6F 6F 5B 56 55 79
77E8:FF FF FF FF FF FF FE A9 7F
77F0:F8 F8 F8 E4 E4 94 54 54 7E

```

Program 3. The Construction Set—Creature Character Set

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```

7000:00 00 00 00 00 00 00 00 E0
7008:00 00 00 03 03 0F 05 0A 81
7010:00 30 FC FF FF FF 55 AA F1
7018:00 00 00 00 00 C0 40 80 FC
7020:00 00 00 01 05 3F 55 0A EA
7028:14 55 55 69 55 FF 55 AA A9
7030:00 00 00 40 50 FC 55 A0 D6
7038:00 03 0F 0F 3F 3E 2A EA DE
7040:FF FF FF FF FF FF BE AA 49
7048:00 C0 F0 F0 FC BC A8 AB 5E
7050:0C 23 00 00 01 05 15 15 5B
7058:00 00 D7 69 55 55 55 55 CA
7060:30 C8 00 00 40 50 54 54 CB
7068:35 D5 50 4A C8 C0 F0 FF 33
7070:FF 7D 55 3C 3C 3C EA EA B2
7078:5C 57 05 A3 23 03 0F FF 7B
7080:AA 80 80 83 8F 8F 8F 8F 88
7088:AA 00 00 FF FF FF FF FF BE
7090:AA 02 02 C2 F2 F2 F2 F2 EF
7098:00 00 00 02 02 09 09 09 E8
70A0:2A AA AA AA AA 08 08 59 20
70A8:00 80 80 A0 A0 18 18 58 B1
70B0:00 03 0D 0D 0D 0A 0A 3F A8
70B8:D5 55 55 55 55 AA AA FF 84
70C0:55 C9 7A 7A 70 A0 A0 FC F9
70C8:00 03 0F 3F 3F 30 30 30 8B
70D0:FF FF FF FF 3F 3F 0C 0C CC
70D8:F0 FC FF FF CF 03 03 03 05

```

```
70E0:00 00 01 05 15 55 55 14 EE
70E8:15 55 55 55 55 55 55 14 68
70F0:00 40 50 54 54 55 55 14 E7
70F8:00 00 00 00 00 00 00 00 D9
7100:00 00 00 00 00 00 00 00 E2
7108:0B 0A 20 28 2A 22 08 0A 6D
7110:EF FE 64 64 9A 9A 98 FE EC
7118:C0 80 20 A0 A0 20 80 80 90
7120:25 AA A0 A1 2A 2A 0A 0A 86
7128:EB 7D 28 69 AA BE 82 AA FB
7130:58 AA 0A 4A A8 A8 A8 A8 B1
7138:E9 F9 FA FB FE FF 3E 3A 5C
7140:AA AA BE FF FF 96 AA A9 54
7148:6B 6F AF EF BF FF BC AC D5
7150:15 57 56 55 15 17 05 01 C3
7158:55 D7 96 7D FF 00 FF 7D 04
7160:54 D5 95 55 54 D0 50 40 B1
7168:FF FE FE FF 3D 3D 35 17 4B
7170:AA AA 9D 55 55 69 6A AA 2C
7178:BF AF AF 7F 5C 5C 54 B4 C6
7180:8F 8F 8F 8F 83 80 88 AA D3
7188:FF FF FF FF FF 82 00 AA 20
7190:F2 F2 F2 F2 C2 02 22 AA 44
7198:02 02 00 00 00 00 00 00 FC
71A0:99 A2 A2 AA 80 2A 2A 08 01
71A8:A1 A1 80 80 80 00 00 00 E0
71B0:3F 2A FF BF AD AF AB A9 E6
71B8:BF AF AE 6E 7D FF D7 AA 6A
71C0:BC AC BF 7E 7A FA FA 6A 2D
71C8:30 3C 0F 0F 0F 0F 0F 0F 87
71D0:0C 0F 0F 3F F0 FC CF 03 71
71D8:03 0F 0C CC FC FC FC 3C 61
71E0:14 05 05 05 05 01 01 01 2F
71E8:D7 04 55 51 51 96 FF 55 B2
71F0:14 50 50 50 50 40 40 40 45
71F8:00 00 00 00 00 00 00 00 DB
7200:00 00 00 00 00 00 00 00 E4
7208:0F 32 02 03 03 0D 3F 3F 7B
7210:DF 56 AA AB EF FD 57 FF B0
7218:C0 F0 00 00 00 C0 FC FC 93
7220:02 02 03 03 0D 35 15 15 93
7228:2A 80 AA EA FF 4C 55 75 97
7230:20 A0 B0 F0 D0 5C 54 55 68
7238:0E 03 00 03 3F 3F FF FF 0C
7240:96 AA FF FF FF FF FF FF 1B
7248:B0 C0 00 C0 FC FC FF FF 9D
7250:02 05 17 55 57 5F 7F 7F 66
7258:55 96 69 D7 D7 7D FF FF EC
7260:80 50 D4 55 D5 F5 FD FD 0A
7268:00 00 03 0F 3F F3 FB EB 4C
```


7270:3C FF FF FF FF FF FF FF 73
7278:00 00 C0 F0 FC CF EF EB 77
7280:00 00 00 00 00 00 03 0F 7A
7288:28 28 28 28 28 EB EF FF E3
7290:00 00 00 00 00 00 C0 F0 E7
7298:00 08 22 20 22 20 20 20 B7
72A0:AA 00 08 80 2A 80 2A 80 0C
72A8:80 28 22 82 22 82 20 82 22
72B0:AB 2A 2A 2A 0A 0A 16 16 98
72B8:55 D7 FF BE AA AA AA AA 2A
72C0:E8 A8 A8 A8 A0 A0 90 94 21
72C8:03 00 00 00 00 02 09 09 52
72D0:C0 FC 7F 1F 95 56 58 58 46
72D8:0C 30 F0 40 80 00 00 00 F5
72E0:22 22 21 22 22 22 22 22 A5
72E8:55 96 AA AA 69 69 55 69 23
72F0:88 88 48 88 88 88 88 88 CD
72F8:00 00 00 00 00 00 00 00 DD
7300:00 00 00 00 00 00 00 00 E6
7308:F3 F3 3C 3C 0F 0D 0F 02 FD
7310:DF FF DF FF 7D DF FF AB F9
7318:FF CF 3C 3C F0 70 F0 C0 2A
7320:55 51 55 15 17 0E 06 01 FF
7328:55 5D 55 75 55 5D 55 55 33
7330:55 55 45 45 54 7C A0 40 2A
7338:FF F3 F3 F0 FC 3C 3E 0E 0D
7340:FF FF FF FF FF FF FF FF 27
7348:FF FF CF CF CF 3C BC B0 BF
7350:7F 5F 17 05 02 00 00 03 15
7358:FF FF FF FF BF 3F 7F D5 0F
7360:FD F5 D4 D0 A0 00 40 F0 E1
7368:B0 A0 28 0A 82 A0 A4 A6 FB
7370:FF FF FF BE AA 28 00 82 BB
7378:0E 0A 28 A0 82 0A 1A 9A 03
7380:3C 3C 0F 03 00 00 00 00 A6
7388:AA AA AA EB AF BE BA AA 1C
7390:3C 3C F0 C0 00 00 00 00 CE
7398:00 08 08 08 00 00 0A 0A 21
73A0:08 A2 08 A2 08 A2 88 00 3B
73A8:02 82 00 82 00 80 A8 28 D4
73B0:55 55 55 51 51 51 51 51 1B
73B8:AA 69 55 55 69 55 69 55 18
73C0:55 55 55 55 45 45 45 45 B6
73C8:25 09 09 02 02 00 00 00 D5
73D0:58 58 58 5A 56 95 95 25 04
73D8:00 00 00 00 80 80 80 60 27
73E0:21 22 22 22 22 22 22 57 7C
73E8:69 69 69 AA 96 AA AA AA 16
73F0:48 88 88 88 88 88 88 C4 F3
73F8:00 00 00 00 00 00 00 00 DF

```

7400:00 00 00 00 00 00 00 00 E8
7408:0F 0F 0F 0F 0F 0F 0F 0F F0
7410:FE FF CF CF CF CF 03 03 36
7418:A0 F8 F8 F2 C0 C0 C0 C0 29
7420:01 00 05 15 35 D5 D5 D5 FD
7428:55 28 55 55 55 55 55 41 B1
7430:00 00 50 54 5C 57 57 57 AE
7438:0E 02 03 0F 0F 3F 3F 3F 2D
7440:BF FF FF FF FF FF C3 03 93
7448:A0 C0 F0 F0 F0 FC FC FC 51
7450:0F 3F 0F 0F 03 00 00 00 7B
7458:FF FF FF FF FF FF FF 3F 80
7460:FC FF FC FC F0 C0 C0 00 43
7468:A6 A6 26 06 82 A0 A4 A6 F9
7470:A6 A6 A6 A6 A6 24 00 82 DD
7478:9A 9A 98 90 82 0A 1A 9A 7C
7480:00 0A 09 09 0A 09 09 09 2D
7488:AA AA 55 55 AA 55 55 55 1C
7490:00 A0 60 60 A0 60 60 60 5B
7498:02 00 02 08 08 08 08 08 BB
74A0:00 A2 08 00 00 00 00 00 33
74A8:20 80 20 08 08 08 08 08 BE
74B0:A5 F5 35 05 00 05 05 05 04
74B8:69 55 69 41 00 55 55 51 3E
74C0:5A 5F 5C 50 00 50 50 50 71
74C8:00 00 00 00 01 04 11 41 2D
74D0:09 02 00 00 14 54 69 80 04
74D8:60 58 96 96 96 98 98 58 E4
74E0:17 13 07 03 03 03 03 03 58
74E8:FF FF C3 C3 C3 C3 C3 C3 FE
74F0:D5 D4 C4 C0 C0 C0 C0 C0 E9
74F8:00 00 00 00 00 00 00 00 E1
7500:00 00 00 00 00 00 00 00 EA
7508:0F 3C 3C 3C 3C 3C 3C 00 20
7510:03 03 03 03 00 00 00 00 CD
7518:C0 C0 C0 C0 F0 F0 F0 00 E4
7520:05 05 01 01 05 05 05 05 4A
7528:41 41 41 41 00 00 00 00 40
7530:5F 7F 7F 7F 53 50 50 50 5F
7538:FC F0 F0 F0 3C 3C 3C 3F 95
7540:00 00 00 00 00 00 03 03 34
7548:FF 3F 0F 3F FC F0 F0 C0 27
7550:00 00 00 00 03 0F 0F 02 AF
7558:3F 3F FF FF FF F3 F3 80 EA
7560:00 00 C0 C0 F0 FC FC A0 85
7568:A6 A6 26 06 80 A0 0A 00 10
7570:A6 A6 A6 A6 A6 00 02 A8 79
7578:9A 9A 98 80 02 28 80 00 24
7580:0A 09 09 0A 00 02 02 00 80
7588:AA 55 55 AA 80 22 22 80 C4

```

```

7590:A0 60 60 A0 80 20 20 80 3F
7598:08 00 02 02 02 02 02 00 04
75A0:00 00 00 00 00 00 00 80 0C
75A8:08 00 20 20 20 20 20 80 DF
75B0:01 01 01 01 01 01 01 04 9E
75B8:40 40 40 40 40 40 01 01 E5
75C0:50 50 50 50 50 50 40 00 3B
75C8:14 55 58 64 61 18 16 05 01
75D0:56 81 06 12 49 A5 16 68 9E
75D8:58 58 50 60 60 80 00 00 1B
75E0:03 03 03 03 01 01 01 05 B1
75E8:C3 C3 C3 C3 41 41 41 55 42
75F0:C0 C0 C0 C0 40 40 40 50 64
75F8:00 00 00 00 00 00 00 00 E3
7600:00 00 00 00 00 00 00 00 EC
7608:00 00 00 00 02 02 02 2A 3B
7610:00 00 00 AA AA AA 96 AA 7F
7618:00 00 00 00 80 80 80 A8 B4
7620:00 00 00 00 00 00 FF FF 0D
7628:3F FF FF FF FB EA FF FF 40
7630:00 C0 C0 C0 C0 C0 FF FF 7A
7638:00 00 00 03 0D 0D 0A 0A 10
7640:0D 35 57 57 55 55 AA AA 61
7648:C0 F0 0A 0A C0 70 A0 A0 5D
7650:00 00 00 01 01 07 55 55 71
7658:14 55 55 55 54 55 FF 55 F1
7660:00 00 00 40 40 D0 55 55 96
7668:04 05 05 05 05 05 05 05 D4
7670:41 41 41 41 41 41 41 41 5D
7678:10 50 50 50 50 50 50 50 45
7680:0F 03 03 03 03 03 03 03 73
7688:C0 00 00 00 00 00 00 00 D5
7690:FC 30 30 30 30 30 30 30 E3
7698:A0 F0 F0 E0 F0 FB FF CF 85
76A0:00 00 00 00 00 00 C3 C3 D8
76A8:0A 0F 0F 0B 0F EF FF F3 1D
76B0:FF FF FF FF FF FF FF FF 9D
76B8:FF FF F7 FF FF F7 FF FF 84
76C0:FF FF FF FF FF FF FF FF AD
76C8:55 55 55 55 55 55 55 55 B5
76D0:59 55 55 59 55 59 55 55 10
76D8:55 55 55 55 55 55 55 55 C5
76E0:AA 80 AA 80 AA 80 AA AA F7
76E8:AA 00 AA 00 AA 00 AA AA D5
76F0:AA 02 AA 02 AA 02 AA AA 86
76F8:00 00 00 00 00 00 00 00 E5
7700:00 00 00 00 00 00 00 00 EE
7708:00 02 02 02 02 02 02 0A FD
7710:00 AA 99 99 99 99 99 AA 87
7718:00 80 80 80 80 80 80 A0 E6

```

```

7720:82 28 0A 22 82 00 00 02 D3
7728:20 08 02 20 80 82 88 A0 2B
7730:20 80 88 22 A8 00 00 00 C7
7738:00 00 00 00 00 00 00 02 29
7740:AA AA AA AA AA AA AA 2F
7748:00 00 00 00 00 00 00 80 B7
7750:0A 0B 0B 0B 0B 0A 0B 2A D9
7758:AA AE EE FA FE FE AE AA D1
7760:00 00 00 00 00 00 00 80 CF
7768:0C 0F 0F 0F 0E 0F FC FC 97
7770:C3 C3 C3 E3 8A C3 C3 97
7778:30 F0 F0 B0 F0 F0 3F 3F ED
7780:00 03 03 03 03 0F 33 C3 3F
7788:00 00 00 00 00 C0 33 33 14
7790:00 30 30 30 30 F0 3C 33 85
7798:00 00 00 00 03 04 0C 08 CF
77A0:0F 3F 3D FF CC 44 CC 88 28
77A8:00 C0 C0 F0 CC 44 CC 88 88
77B0:0C 3F 3E 02 00 00 00 00 5D
77B8:00 00 00 00 80 20 22 08 78
77C0:0F 3F 3B 08 20 80 00 00 F1
77C8:15 05 05 05 05 05 05 04 BE
77D0:41 41 41 41 41 41 41 41 BF
77D8:50 50 50 50 50 50 50 10 87
77E0:00 00 00 00 03 0D 01 00 1E
77E8:C0 C0 00 C0 F0 DC D0 C0 D1
77F0:0C 0C 00 0C 3F DD 1D 0C 61

```

Program 4. The Construction Set—Shapes Character Set

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```

7000:00 00 00 00 00 00 00 00 E0
7008:AA AA A0 A0 A0 A0 A0 A0 70
7010:AA AA 00 00 00 00 00 00 F0
7018:AA AA 0A 0A 0A 0A 0A 0A 71
7020:AA AA AA AA AA AA AA AA 01
7028:AA AA AA AA AA AA AA AA 09
7030:AA AA AA AA AA AA AA AA 11
7038:55 55 55 55 55 55 55 55 19
7040:55 55 55 55 55 55 55 55 21
7048:55 55 55 55 55 55 55 55 29
7050:FF FF FF FF FF FF FF FF 31
7058:FF FF FF FF FF FF FF FF 39
7060:FF FF FF FF FF FF FF FF 41
7068:FF FF F0 F0 F0 F0 F0 F0 94
7070:FF FF 3C 3C 3C 3C 3C 3C 24
7078:FF FF 0F 0F 0F 0F 0F 0F 0E
7080:00 00 00 00 00 00 00 00 61
7088:0C 0C 0C 3F 3F 3F FF FF DE

```

```

7090:00 00 00 00 00 00 C0 C0 B3
7098:00 00 00 00 00 00 00 02 7B
70A0:20 20 20 20 20 20 A8 56 C8
70A8:00 00 00 00 00 00 00 00 89
70B0:20 26 26 26 26 26 26 26 8E
70B8:00 00 40 66 66 66 66 66 08
70C0:02 26 66 66 66 66 66 66 5F
70C8:00 00 00 00 00 00 00 00 A9
70D0:04 04 04 0C 0C 0C 0C 0C AA
70D8:00 00 00 00 00 00 00 00 B9
70E0:00 00 00 00 00 00 00 00 C1
70E8:14 14 14 14 14 14 14 14 C9
70F0:00 00 00 00 00 00 00 00 D1
70F8:00 00 00 00 00 00 00 00 D9
7100:00 00 00 00 00 00 00 00 E2
7108:A0 A0 A0 A0 A0 A0 A0 A0 EA
7110:00 00 00 00 00 00 00 00 F2
7118:0A 0A 0A 0A 0A 0A 0A 0A FA
7120:AA AA AA AA AA AA AA AA 03
7128:AA AA AA AA AA AA AA AA 0B
7130:AA AA AA AA AA AA AA AA 13
7138:55 55 55 55 55 55 55 55 1B
7140:55 55 55 55 55 55 55 55 23
7148:55 55 55 55 55 55 55 55 2B
7150:FF FF FF FF FF FF FF FF 33
7158:FF FF FF FF FF FF FF FF 3B
7160:FF FF FF FF FF FF FF FF 43
7168:F0 F0 F0 FF F0 F0 F0 F0 3C
7170:3C 3C 3C FF 3C 3C 3C 3C 8F
7178:0F 0F 0F FF 0F 0F 0F 0F 6A
7180:00 03 03 03 0F 03 03 03 42
7188:FF FF FF FF FF FF FF FF 6B
7190:C0 F0 F0 F0 FC F0 F0 F0 BB
7198:09 25 96 5A 69 69 69 5A DF
71A0:65 A9 9A 56 55 65 55 56 9A
71A8:80 60 58 94 A4 A5 A4 94 D1
71B0:26 26 26 26 26 26 26 26 93
71B8:66 66 66 66 66 66 66 66 9B
71C0:66 66 66 66 66 66 66 66 A3
71C8:0A 09 09 09 FF 09 09 09 E3
71D0:AE 5D 5D 7F F7 7F 5D 5D DB
71D8:A8 58 58 58 FF 58 58 58 21
71E0:00 00 00 00 55 55 00 00 C3
71E8:14 14 14 14 55 55 14 14 DA
71F0:00 00 00 00 55 55 00 00 D3
71F8:00 00 00 00 00 00 00 00 DB
7200:00 00 00 00 00 00 00 00 E4
7208:A0 A0 A0 A0 A0 A0 AA AA 0B
7210:00 00 00 00 00 00 AA AA F4
7218:0A 0A 0A 0A 0A 0A AA AA DE

```

```

7220:AA AA AA AA AA AA AA AA 05
7228:AA AA AA AA AA AA AA AA 0D
7230:AA AA AA AA AA AA AA AA 15
7238:55 55 55 55 55 55 55 55 1D
7240:55 55 55 55 55 55 55 55 25
7248:55 55 55 55 55 55 55 55 2D
7250:FF FF FF FF FF FF FF FF 35
7258:FF FF FF FF FF FF FF FF 3D
7260:FF FF FF FF FF FF FF FF 45
7268:F0 F0 F0 F0 F0 F0 FF FF 7A
7270:3C 3C 3C 3C 3C 3C FF FF A0
7278:0F 0F 0F 0F 0F 0F FF FF 30
7280:00 00 00 00 00 00 00 00 65
7288:FF FF FF 3F 3F 3F 0C 0C 7C
7290:C0 C0 C0 00 00 00 00 00 1E
7298:96 25 09 02 00 00 00 00 53
72A0:9A A9 65 56 A8 20 20 20 75
72A8:58 60 80 00 00 00 00 00 E1
72B0:26 26 26 26 26 26 26 26 95
72B8:66 66 66 66 66 66 66 66 9D
72C0:66 66 66 66 66 66 66 66 A5
72C8:0A 00 00 00 00 00 00 00 B2
72D0:AE 0C 0C 0C 0C 0C 04 04 EE
72D8:A8 00 00 00 00 00 00 00 12
72E0:00 00 00 00 00 00 00 00 C5
72E8:14 14 14 14 14 14 14 14 CD
72F0:00 00 00 00 00 00 00 00 D5
72F8:00 00 00 00 00 00 00 00 DD
7300:00 00 00 00 00 00 00 00 E6
7308:55 55 00 00 55 55 00 00 EE
7310:55 55 00 00 55 55 00 00 F6
7318:55 55 00 00 55 55 00 00 FE
7320:FF FF 00 00 FF FF 00 00 07
7328:FF FF 00 00 FF FF 00 00 0F
7330:FF FF 00 00 FF FF 00 00 17
7338:AA AA 00 00 AA AA 00 00 1F
7340:AA AA 00 00 AA AA 00 00 27
7348:AA AA 00 00 AA AA 00 00 2F
7350:80 00 20 00 08 00 02 00 BF
7358:00 00 00 00 00 00 00 00 3F
7360:02 00 08 00 20 00 80 00 4B
7368:00 00 0A AA 2A 02 00 00 94
7370:0A AA A8 00 A0 AA 2A 00 20
7378:AA 80 00 00 00 00 A0 AA C0
7380:AA FF FF AA FF FF AA FC B9
7388:AA FF FF AA FF FF AA CC 91
7390:AA FF FF AA FF FF AA FF CC
7398:00 00 00 00 00 00 00 00 7F
73A0:00 00 00 00 00 00 00 00 87
73A8:00 01 01 05 05 15 15 55 3C

```

```

73B0:00 40 40 50 50 54 54 55 86
73B8:00 00 00 00 00 00 00 00 9F
73C0:00 00 00 00 00 00 00 00 A7
73C8:00 00 00 00 02 02 0A 0A E5
73D0:24 24 A5 A5 A5 A5 A5 A5 96
73D8:00 00 00 00 40 40 50 50 B3
73E0:AA BF 3F 2F 0F 0B 03 02 94
73E8:AA FF FF FF FF FF FF FF 25
73F0:AA FE FC F8 F0 E0 C0 80 29
73F8:00 00 00 00 00 00 00 00 DF
7400:00 00 00 00 00 00 00 00 E8
7408:55 55 00 00 55 55 00 00 F0
7410:55 55 00 00 55 55 00 00 F8
7418:55 55 00 00 55 55 00 00 01
7420:FF FF 00 00 FF FF 00 00 09
7428:FF FF 00 00 FF FF 00 00 11
7430:FF FF 00 00 FF FF 00 00 19
7438:AA AA 00 00 AA AA 00 00 21
7440:AA AA 00 00 AA AA 00 00 29
7448:AA AA 00 00 AA AA 00 00 31
7450:00 00 00 00 00 00 00 00 39
7458:82 00 28 00 28 00 82 00 CD
7460:00 00 00 00 00 00 00 00 49
7468:00 00 0A AA 2A 02 00 00 96
7470:0A AA A8 00 A0 AA 2A 02 24
7478:A8 80 00 00 00 00 A0 AA C1
7480:FC A8 FC FC A8 FC FC A8 5D
7488:CC 88 CC CC 88 CC CC 88 F9
7490:FF AA FF FF AA FF FF AA 24
7498:00 00 00 00 00 00 00 00 81
74A0:00 01 01 05 05 15 15 55 36
74A8:55 55 55 55 55 55 55 55 91
74B0:55 55 55 55 55 55 55 55 99
74B8:00 40 40 50 50 54 54 55 90
74C0:00 00 00 00 00 00 00 00 A9
74C8:2A 2A AA AA AA 55 55 15 BB
74D0:A5 A5 A5 A5 A5 5A 5A 5A AA
74D8:54 54 55 55 55 AA AA A8 54
74E0:00 00 00 00 00 00 00 00 C9
74E8:FF BE 3C 3C 3C 3C 3C BE D6
74F0:00 00 00 00 00 00 00 00 D9
74F8:00 00 00 00 00 00 00 00 E1
7500:00 00 00 00 00 00 00 00 EA
7508:55 55 00 00 55 55 00 00 F2
7510:55 55 00 00 55 55 00 00 FA
7518:55 55 00 00 55 55 00 00 03
7520:FF FF 00 00 FF FF 00 00 0B
7528:FF FF 00 00 FF FF 00 00 13
7530:FF FF 00 00 FF FF 00 00 1B
7538:AA AA 00 00 AA AA 00 00 23

```

7540:AA AA 00 00 AA AA 00 00 2B
7548:AA AA 00 00 AA AA 00 00 33
7550:02 00 08 00 20 00 80 00 3F
7558:00 00 00 00 00 00 00 00 43
7560:80 00 20 00 08 00 02 00 D3
7568:00 00 0A AA 2A 02 00 00 98
7570:0A AA A8 80 A0 AA 2A 02 2E
7578:A8 80 00 00 00 00 A0 AA C3
7580:FC FC AA FF FF AA FF AA D3
7588:CC CC AA FF FF AA FF AA B7
7590:FF FF AA FF FF AA FF AA 26
7598:00 01 01 05 05 15 15 55 30
75A0:55 55 55 55 55 55 55 55 8B
75A8:55 55 55 55 55 55 55 55 93
75B0:55 55 55 55 55 55 55 55 9B
75B8:55 55 55 55 55 55 55 55 A3
75C0:00 40 40 50 50 54 54 55 9A
75C8:15 05 05 01 01 00 00 00 38
75D0:5A 5A 5A 5A 5A 5A 5A 18 79
75D8:A8 A0 A0 80 80 00 00 00 60
75E0:00 02 03 0B 0F 2F 3F AA BB
75E8:FF FF FF FF FF FF FF AA 7E
75F0:00 80 C0 E0 F0 F8 FC AA 32
75F8:00 00 00 00 00 00 00 00 E3
7600:00 00 00 00 00 00 00 00 EC
7608:FF 02 02 02 02 02 02 02 F3
7610:FF 55 55 55 55 55 55 55 52
7618:FF 80 80 80 80 80 80 80 C4
7620:00 00 03 0F 0F 3F 3F FF 52
7628:3C FF FF FF FF D7 D7 55 97
7630:00 00 C0 F0 F0 FC FC FF B9
7638:C0 C0 C0 EA D5 D5 D5 D5 04
7640:00 00 00 AA 55 55 55 55 D7
7648:03 03 03 AB 57 57 57 57 B0
7650:A0 A0 AA A0 A0 AA A0 A0 A6
7658:00 00 AA 00 00 AA 00 00 45
7660:0A 0A AA 0A 0A AA 0A 0A E3
7668:00 00 00 00 02 08 20 80 46
7670:00 00 28 AA 00 00 00 00 0D
7678:00 00 00 00 80 20 08 02 FB
7680:F0 5C 5C 5C 57 57 57 55 6A
7688:00 00 00 00 00 00 00 C3 39
7690:0F 35 35 35 D5 D5 D5 55 53
7698:55 40 40 40 40 40 40 55 25
76A0:55 00 00 00 00 00 00 55 8D
76A8:55 01 01 01 01 01 01 55 14
76B0:80 80 AA 80 AA 80 AA 80 88
76B8:00 00 AA 00 AA 00 AA 00 A5
76C0:02 02 AA 02 AA 02 AA 02 59
76C8:EE BB EE BB EE BB EE BB B5

76D0:EE BB EE BB EE BB EE BB BD
76D8:EE BB EE BB EE BB EE BB C5
76E0:04 55 04 04 55 04 04 55 FD
76E8:10 55 10 10 55 10 10 55 96
76F0:41 55 41 41 55 41 41 55 97
76F8:00 00 00 00 00 00 00 00 E5
7700:00 00 00 00 00 00 00 00 EE
7708:02 02 02 02 02 02 02 02 FF F4
7710:55 55 55 55 55 55 55 55 FF A9
7718:80 80 80 80 80 80 80 80 FF 86
7720:FD FF 3F 3F 0F 03 00 00 6E
7728:55 55 D7 D7 FF FF FF 3C CB
7730:7F FF FC FC F0 C0 00 00 D8
7738:D5 D5 D5 D5 EA C0 C0 C0 3C
7740:55 55 55 55 AA 00 00 00 84
7748:57 57 57 57 AB 03 03 03 8B
7750:AA A0 A0 AA A0 A0 AA A0 F8
7758:AA 00 00 AA 00 00 AA 00 9C
7760:AA 0A 0A AA 0A 0A AA 0A EA
7768:80 80 20 08 02 00 00 00 4C
7770:00 00 00 00 00 00 AA 28 5A
7778:02 02 08 20 80 00 00 00 EF
7780:55 55 57 57 57 5C 5C FC B1
7788:7D C3 00 00 00 00 00 00 27
7790:55 55 D5 D5 D5 35 35 3F C4
7798:FF C0 C0 C0 C0 C0 C0 FF 66
77A0:FF 00 00 00 00 00 00 00 FF 8F
77A8:FF 03 03 03 03 03 03 FF 13
77B0:AA 80 80 80 80 80 80 AA DE
77B8:AA 00 00 00 00 00 00 AA A7
77C0:AA 02 02 02 02 02 02 AA AC
77C8:66 99 66 99 66 99 66 99 B7
77D0:66 99 66 99 66 99 66 99 BF
77D8:66 99 66 99 66 99 66 99 C7
77E0:0C 0C FF 0C 0C FF 0C 0C 1E
77E8:30 30 FF 30 30 FF 30 30 11
77F0:C3 C3 FF C3 C3 FF C3 C3 58

Budget Planner

Gregory L. Smith

Managing your budget can be a snap with "Budget Planner." It totals your income and bills week by week, letting you plan your expenses in advance and make adjustments for special occasions like vacations or birthdays.

Have you ever wondered if you can afford additional monthly expenses—like a mortgage or more credit cards? Or how much "discretionary income" (money left after paying bills) you might have around Christmastime? I used to manage all the details in my head, but usually had little or no idea as to how much money there actually was (or worse, where it all went).

"Budget Planner" is a mini-spreadsheet written in BASIC. It's designed specifically to handle weekly and monthly income/expenses for an entire year. Budget Planner will sum a column (a week of income and expenses) and present a total. The total is then carried over into the next week as a previous balance.

Typing It In

The program listing is the 64 version. Minor changes allow it to run on the 128 in 128 mode. If you have a Commodore 128, type in the listing for the 64 and add this line:

```
EX 90 FORI=1TO8:READ A:KEYI,CHR$(A):NEXT:DATA 133,
137,134,138,135,139,136,140
```

Commodore 128 owners can also add this line, though it's optional:

```
KJ 1890 COLOR 0,4:COLOR 4,4
```

When you first run it, Budget Planner asks for a filename of the information to be loaded. Once you've set up your yearly finances, you can save the data to disk or tape. Since you're just getting started, simply press RETURN and you'll be prompted for the date. This is the date you want to start budgeting. (I like to set my budget up based on Friday as the

end of the week. For 1987, 2-JAN-1987 is the first Friday of the year.)

Enter the day as a number, the month as a three-character abbreviation, and the year as a four-digit value. You're not required to use the beginning of the year. You may start in the middle or at the end if you like. The program is even smart enough to handle leap years.

BUDGET PLANNER 1987

	2-JAN	9-JAN	16-JAN	23-JAN	30-JAN
PREVBAL	0.00	428.00	620.50	952.37	1110.37
PAYROLL	638.00	238.00	638.00	238.00	638.00
HOUSE PMT	0.00	0.00	0.00	0.00	-650.00
ELECTRIC	0.00	0.00	-69.74	0.00	0.00
TELEPHONE	0.00	0.00	-17.36	0.00	0.00
CABLE	0.00	0.00	-9.03	0.00	0.00
CAR PMT	0.00	0.00	0.00	0.00	-208.12
GROCERIES	-50.00	0.00	-50.00	0.00	-50.00
GAS	-20.00	0.00	-20.00	0.00	-20.00
CHARGE CRD	0.00	0.00	0.00	0.00	-25.57
INSURANCE	0.00	0.00	0.00	0.00	0.00
SAVINGS	0.00	0.00	0.00	0.00	-650.00
MISC.	-140.00	-45.50	-140.00	-80.00	-140.00
TOTAL	428.00	620.50	952.37	1110.37	4.68

This sample printout shows how "Budget Planner" has been set up for a five-week month. Note that the first Previous Balance (PREVBAL) entry is 0, indicating that this is when the program was first put in use. Also note that decimal mode was selected.

The screen then displays four columns of information. The first column holds the labels for each row. The only two rows available when you first start are the *PREVBAL* (previous balance) and *TOTAL*.

The other three columns are the budget data. The week number is shown on the left side. The date of each week is above the corresponding column. When you use the left/right cursor keys to scroll around the budget, the week number changes.

Enter income and expenses by typing the amount (which appears on line 2 of the screen) and pressing RETURN. Only numbers and the +, -, and . keys will work here. Use the delete key to remove the last character typed.

The arrow keys are used to move the reverse-video cursor around the screen. You cannot alter the *TOTAL* row since that's controlled by the computer. You can see the next week

BUDGET PLANNER 1987				
	6-FEB	13-FEB	20-FEB	27-FEB
PREVBAL	5	3	382	395
PAYROLL	238	638	238	638
HOUSE PMT	0	0	0	-650
ELECTRIC	0	-61	0	0
TELEPHONE	0	-19	0	0
CABLE	0	-9	0	0
CAR PMT	0	0	0	-208
GROCERIES	0	-50	0	-50
GAS	0	-20	0	-20
CHARGE CRD	0	0	0	-112
INSURANCE	0	0	0	0
SAVINGS	-240	0	-225	0
MISC.	0	-100	0	0
Total	3	382	395	-7

In this example, we see a four-week month and decimal rounding. Also note the highlighted (reverse field) negative balance.

by scrolling off the edge of the screen. This is allowed in both directions. The up-/down-arrow keys work after more expenses have been added. However, you cannot scroll vertically as only 18 rows (one screenful) are allowed.

The Function Keys

The function keys trigger the various options. Pressing f1 displays the help menu. A window appears in the top left corner of the screen with reminders as to which key does what.

Press f2 to go to a specific week. You may jump to any week you like by entering its week number (week 1 is the start of the budget and week 52 is the end.) You can also do a "relative jump" by using a plus or minus sign (-3 means go back 3 weeks and +3 means go forward 3 weeks). This is faster than pressing the right-arrow key several times. Also, SHIFT-E jumps the cursor to the end of the next month. This is handy since most of us pay bills at the end of the month.

The f3 key is the LOAD key. You may load a new budget (data from the disk) at any time. You'll be asked to supply a filename. If you pressed f3 accidentally, just press RETURN in response to the filename prompt. You'll be returned to your original budget. Likewise, f4 (SHIFT-f3) is the SAVE key. You might want to use the year of the budget as the filename. Again, if you decide not to save the budget, simply press RETURN.

To add a new row, use f5 (labeled +ROW in the help menu). It inserts a row below the cursor for new expenses (or new income). You'll be prompted for a name, which can be up to ten characters long. As with the other options, pressing RETURN cancels the operation. Once a name has been assigned, you must specify whether it's a source of income (I) or an expense (E). Pressing any key other than I defaults to an expense.

This is important. Every value you enter in this field will be negative if it's an expense and positive if it's income. (You don't need to place a minus sign in front of an expense.)

Likewise, f6 deletes the row the cursor is on. You must verify by entering YES (or the letter Y) when asked. If you enter anything else, you'll be returned to the current budget.

The f7 key computes a new balance. Balances are *not* recalculated after each new amount is entered. Since the program is written in BASIC, it takes several seconds to recompute the array and display the results. This delay would be annoying if the program made all the computations every time you added another number.

Pressing f8 prints out the results. Be sure your printer is turned on. You may specify a range of months to print. Pressing RETURN in response to the FROM? prompt returns you to the budget without printing. Enter a valid 3-character abbreviation (JAN or AUG, for example). When prompted to enter TO?, you may press RETURN to print only one month. To print several months, use the appropriate three-character abbreviation.

Special Keys

There are five *SHIFT* functions in addition to the function keys. One has already been mentioned, the SHIFT-E key, which jumps to the end of the next month.

SHIFT-D switches between showing and not showing the decimals (pennies). Budget Planner defaults to showing decimals. (The variable DP is initialized in line 1950. Simply change it to DP=0 to change the default.) When decimals aren't being displayed, you see the rounded amount onscreen.

SHIFT-F finds the next negative value in the TOTAL row. If you ever "go negative," the TOTAL value and the offending value turn red. In such a case, you don't have enough income to cover expenses and it may be necessary to juggle some

bills. SHIFT-F saves you from trying to find these trouble spots manually.

Use SHIFT-M to flag monthly expenses. If your rent is \$500, for example, you type in 500 in that row and then, instead of RETURN, press SHIFT-M. The \$500 expense will be added to the end of every month. Note that this value is placed only on the same row as the cursor. Also, only the months AFTER the cursor position are affected. (For example, if you are on week 26, only weeks 26-52 are affected. The other 25 weeks will not be changed.) Naturally, this key works for income as well.

SHIFT-W is the weekly expense key (groceries, for example). Pressing it places the line-2 value in each week after the cursor position. Again, you use it in place of RETURN after entering a value. It works much like the monthly expense key and also works for the income fields.

Press SHIFT-T if you use a Datasette. It directs data storage and retrieval (f3 and f4) to the tape drive.

The SHIFT-Q key is the Disk Select key. It causes data to be saved to and loaded from a disk drive. (Note: Disk is the default. Changing DEV=8 to DEV=1 in line 100 changes the default device to tape.)

Budget Planner

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

128 owners: See modifications in the article.

```

SR 100 DEV=8:GOTO1850
HJ 110 PRINTHM$;LEFT$(X$,X);LEFT$(Y$,Y);:RETURN
SD 120 A=ABS(X)+.005-.5*(DP=0):A1=INT(A):A$=MID$(STR$(A1),2):IFX<0THENA$="-"+A$
AQ 130 IFDPTHEN A2$=MID$(STR$(INT((A-A1)*100)),2):IFLEN(A2$)<2THENA2$="0"+A2$
KX 140 IFDPTHENA$=A$+"."+A2$
JC 150 A$=RIGHT$(SP$+A$,F):RETURN
XF 160 A$=LEFT$(SP$, (X-LEN(A$))/2)+A$:RETURN
HC 170 FORI=1TONW:A=0:FORJ=0TONR-1:A=A+TB(J,I):NEXT T:TB(NR,I)=A:TB(0,I+1)=A:NEXT
DQ 180 FORI=1TONW:MM=0:IFTB(NR,I)<0THENMM=-1:I=NW
JG 190 NEXT:RETURN
QM 200 A$=RIGHT$(SP$+"WEEK"+STR$(CW),F-2):X=2:Y=3:GOSUB110:PRINT A$;:RETURN
JX 210 A$="BUDGET PLANNER"+STR$(YR):X=40:GOSUB160
SG 220 PRINTCL$;FG$;A$:PRINTD$:PRINTEA$:EA$=""
BA 230 GOSUB280:FORI=0TONR:IFMMANDI=NRTHEPRINTDEB T$;

```

```

FF 240 PRINTHM$;LEFT$(Y$,I+5);A$=TB$(I):GOSUB150:
    PRINTA$;FG$;
SP 250 FORJ=WTOW+2:IFI=NRANDTB(I,J)<0THENPRINTDEBT
    $;
RR 260 X=TB(I,J):GOSUB120:PRINTA$;FG$;
FD 270 NEXT:NEXT:RETURN
KA 280 GOSUB200:A=D:B=M:C=YR
PB 290 FORZ=1TO3
AJ 300 A$=STR$(D)+MNTH$(M):GOSUB150:PRINTA$;
GH 310 GOSUB580:NEXT
QR 320 D=A:M=B:YR=C
KX 330 PRINT "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
    CCC"
HP 340 RETURN
RD 350 GOSUB200:X=TB(CR,CW):GOSUB120:B$=RV$+A$+NV$
    :X=CC*10:Y=CR+5:GOSUB110
SD 360 PRINTB$;
MC 370 GETC$:IFC$=""THEN370
XE 380 IFC$="+ "ORC$="- "ORC$="."OR(C$>="0"ANDC$<="9
    ")THEN740
GG 390 IFC$=UP$THEN610
GS 400 IFC$=DN$THEN640
JR 410 IFC$=LF$THEN660
BE 420 IFC$=RT$THEN700
GR 430 IFC$=CR$THENRETURN
KS 440 IFC$=BK$THENIFLEN(D$)<>0THENC$="":D$=LEFT$(
    D$,LEN(D$)-1):GOTO740
XE 450 GOSUB110:PRINTA$;
QA 460 IFASC(C$)>132ANDASC(C$)<141THEN760
JP 470 IFC$="M"THEN1650
XA 480 IFC$="D"THENDP=NOTDP:GOSUB210:GOTO350
RA 490 IFC$="E"THEN1730
QK 500 IFC$="F"THEN1790
FR 510 IFC$="W"THEN1630
CJ 520 IFC$="T"THENDEV=1:EA$="{RVS}TAPE SELECTED
    {OFF}":GOSUB210:GOTO350
RQ 530 IFC$="Q"THENDEV=8:EA$="{RVS}DISK SELECTED
    {OFF}":GOSUB210:GOTO350
CG 540 GOSUB110:PRINTB$:GOTO370
FC 550 D=D-7:IFD>=1THENRETURN
DE 560 M=M-1:IFM>=1THEND=D+MNTH(M):RETURN
QG 570 YR=YR-1:M=12:D=D+MNTH(M):RETURN
CQ 580 D=D+7:IFD<=MNTH(M)THENRETURN
CX 590 D=D-MNTH(M):M=M+1:IFM<13THENRETURN
FB 600 YR=YR+1:M=1:RETURN
RC 610 IFCR=0THEN370
BJ 620 CR=CR-1
CR 630 GOSUB110:PRINTA$;:GOTO350
KH 640 IFCR<>NR-1THENCR=CR+1:GOTO630
GS 650 GOTO370

```

```

CQ 660 IFCC<>1THENCC=CC-1: CW= CW-1 :GOTO630
KF 670 IFCW=1THEN370
QC 680 CW= CW-1 :W=W-1 :GOSUB550
BS 690 GOSUB210:GOTO350
BX 700 IFCC<>3THENCC=CC+1: CW= CW+1 :GOTO630
ES 710 IFCW= NWTHEN370
KX 720 CW= CW+1 :W=W+1 :GOSUB580
QR 730 GOSUB210:GOTO350
GC 740 IFLEN(D$)<10THEND$=D$+C$:PRINTHM$;DN$;D$;LE
    FT$(SP$,10-LEN(D$)):GOTO370
SF 750 GOTO370
AQ 760 PRINT HM$;DN$;SP$:PRINTSP$
XD 770 ONASC(C$)-1.32GOSUB780,870,980,1090,1100,125
    0,1330,1400:GOTO350
DD 780 X=0:Y=0:GOSUB110:PRINTWH$;"<F1>HELP
    {3 SPACES}<F2>GOTO{2 SPACES}<E>ND OF MONTH
    {SPACE}"
BF 790 Y=Y+1:GOSUB110:PRINTWH$;"<F3>LOAD
    {2 SHIFT-SPACE} <F4>SAVE{2 SPACES}<F>IND NE
    G{5 SPACES}"
XK 800 Y=Y+1:GOSUB110:PRINTWH$;"<F5>+ROW
    {2 SHIFT-SPACE} <F6>-ROW{2 SPACES}<M>ONTHLY
    EXP{2 SPACES}"
SP 810 Y=Y+1:GOSUB110:PRINTWH$;"<F7>CALC
    {2 SHIFT-SPACE} <F8>PRNT{2 SPACES}<W>EEKLY
    {SPACE}EXP{3 SPACES}"
CD 820 Y=Y+1:GOSUB110:PRINTWH$;"<Q>DISK{4 SPACES}<
    T>APE{4 SPACES}<D>ECIMAL{6 SPACES}"
EH 830 Y=Y+1:GOSUB110:PRINT"<PRESS ANY KEY>
    {3 SPACES}"
CA 840 GETA$:IFA$=""THEN840
FH 850 PRINTFG$;
KC 860 GOSUB210:RETURN
QA 870 X=0:Y=0:GOSUB110:PRINTWH$;"{RVS}*LOAD*{OFF}
    "
HH 880 Y=Y+1:GOSUB110:F$="":INPUT"FILENAME";F$:IFF
    $=""THEN1210
FA 890 IFDE=8THEN910
BG 900 OPEN2,1,0,F$:GOTO920
PM 910 OPEN15,8,15:OPEN2,8,2,F$+" ,S,R":GOSUB2300:I
    FEATHENGOSUB210:GOTO970
XP 920 PRINTCL$;DN$;"LOADING ";F$
BP 930 INPUT#2,NR,NW,D,M,YR:M0=M:D0=D:Y0=YR:CC=1:C
    R=1:CW=1:W=1:GOSUB2140
SR 940 IFDE=8THENGOSUB2300:IFEATHEN970
HM 950 FORI=0TONR:INPUT#2,TB$(I):FORJ=0TONW:INPUT#
    2,TB(I,J):NEXTJ,I
QQ 960 GOSUB170:GOSUB210
FP 970 CLOSE2:CLOSE15:RETURN

```



```

KD 980 X=0:Y=0:GOSUB110:PRINTWH$;"{RVS}*+ROW*{OFF}
"
AG 990 IFNR=18THENY=Y+1:GOSUB110:PRINT"-NO MORE RO
OM-":GOTO830
KA 1000 N$="":Y=Y+1:GOSUB110:INPUT"NAME";N$
GS 1010 IFN$=" "THEN1080
BH 1020 Y=Y+1:GOSUB110:I$="E":INPUT"( I/E)";I$
SC 1030 A=-1:IFI$="I"THENA=1
FP 1040 CR=CR+1
GD 1050 FORI=NRTOCRSTEP-1:TB$(I+1)=TB$(I):FORJ=0TO
NW:TB(I+1,J)=TB(I,J):NEXTJ,I
RC 1060 FORI=1TONW:TB(CR,I)=0:NEXT:TB$(CR)=LEFT$(N
$,10)
SF 1070 NR=NR+1:TB(CR,0)=A
AR 1080 GOSUB210:RETURN
XE 1090 GOSUB110:PRINTA$;:GOSUB170:GOSUB210:RETURN
FX 1100 X=0:Y=0:GOSUB110:PRINTWH$;"{RVS}*GOTO*
{OFF}"
JM 1110 Y=Y+1:GOSUB110:W$="":INPUT"WEEK";W$
GJ 1120 A=INT(VAL(W$))
KA 1130 IFLEFT$(W$,1)="-"ORLEFT$(W$,1)="+"THENA=CW
+A
KP 1140 IFA=WTHEN1200
EB 1150 IFA>50THENA=50
HP 1160 IFA<=0THEN1210
GX 1170 IFA>WTHEN1220
AA 1180 W=W-1:GOSUB550
AB 1190 IFA<WTHEN1180
JC 1200 CW=W:CC=1
AG 1210 GOSUB210:RETURN
FE 1220 W=W+1:GOSUB580
PD 1230 IFA>WTHEN1220
BR 1240 GOTO1200
PK 1250 X=0:Y=0:GOSUB110:PRINTWH$;"{RVS}*SAVE*
{OFF}"
KC 1260 Y=Y+1:GOSUB110:F$="":INPUT"FILENAME";F$:IF
F$=" "THEN1210
RG 1270 IFDE=8THEN1290
JC 1280 OPEN2,1,2,F$:GOTO1300
XF 1290 OPEN15,8,15:OPEN2,8,2,"@0: "+F$+",S,W":GOSU
B2300:IFEATHEN1320
FJ 1300 PRINT#2,NR;"",";NW;"",";D0;"",";M0;"",";Y0
DJ 1310 FORI=0TONR:PRINT#2,TB$(I):FORJ=0TONW:PRINT
#2,TB(I,J):NEXTJ,I
XP 1320 CLOSE15:CLOSE2:GOSUB170:GOSUB210:RETURN
PM 1330 X=0:Y=0:GOSUB110:PRINTWH$;"{RVS}*--ROW*
{OFF}"
JE 1340 IFCR=0THENY=Y+1:GOSUB110:PRINT"ILLEGAL ROW
":GOTO830
XB 1350 Y=Y+1:GOSUB110:N$="":INPUT"SURE";N$

```

```

RS 1360 IFLEFT$(N$,1)<>"Y"THEN1210
HP 1370 NR=NR-1
DJ 1380 FORI=CRTONR:TB$(I)=TB$(I+1):FORJ=0TONW:TB(
I,J)=TB(I+1,J):NEXTJ,I
BS 1390 CR=CR-1:GOTO1210
SA 1400 Q$="":PRINTM$;WH$;"<PRNT>":INPUT"FROM";Q$
:IFQ$=""THEN1210
AE 1410 Q1$="":INPUT"TO{2 SPACES}";Q1$:IFQ1$=""THE
NQ1$=Q$
BE 1420 Q$="-"+Q$:M1=0:FORI=1TO12:IFMNTH$(I)=Q$THE
NM1=I:I=12
XJ 1430 NEXT:M2=0:Q1$="-"+Q1$:FORI=1TO12:IFMN$(I)=
Q1$THENM2=I:I=12
XA 1440 NEXT:IFM1=0ORM2=0THENEAS="ILLEGAL MONTH":G
OTO1210
ED 1450 OPEN4,4:CMD4:X=79:A$="BUDGET PLANNER"+STR$(
Y0)
MQ 1460 PRINT"{20 SPACES}"A$:PRINT:M=M0:Y=Y0:D=D0:
Q=1
AH 1470 M2=M2+1:IFM2=13THENM2=1
SG 1480 IFM<>M1:THENGOSUB580:Q=Q+1:GOTO1480
EP 1490 PRINTTAB(10);:N=1
PC 1500 A$=STR$(D)+MNTH$(M):GOSUB150:PRINTA$;
CD 1510 A=M:GOSUB580:IFA=MTHENN=N+1:GOTO1500
DR 1520 PRINT:FORI=0TON:PRINT"CCCCCCCC";:NEXT:PR
INT
PP 1530 FORI=0TONR:A$=TB$(I):GOSUB150
GS 1540 IFI=NRANDMMTHENPRINTRV$;A$;NV$;:GOTO1560
JR 1550 PRINTA$;
QM 1560 QQ=Q+N-1:IFQQ>52THENQQ=52
RM 1570 FORJ=QTOQQ:X=TB(I,J):GOSUB120
HG 1580 IFI=NRANDX<0THENPRINTRV$;A$;NV$;:GOTO1600
QS 1590 PRINTA$;
PG 1600 NEXTJ:PRINT:NEXTI
FJ 1610 PRINT:PRINT:Q=Q+N:IFQ<53ANDM<>M2THEN1490
SH 1620 M=M0:D=D0:YR=Y0:W=1:CW=1:CC=1:CR=0:PRINT#4
:CLOSE4:GOTO1210
XP 1630 IFD$=""THENRETURN
XJ 1640 A=TB(CR,0)*VAL(D$):FORI=CWTONW:TB(CR,I)=A:
NEXTI:D$="":GOSUB210:GOTO350
CF 1650 IFD$=""THENRETURN
FR 1660 A=D:B=M:C=YR:Q=W:X=M
JJ 1670 IFW<>CWTHENGOSUB580:W=W+1:GOTO1670
PR 1680 X=M
GH 1690 GOSUB580:W=W+1:IFX=MTHEN1690
PP 1700 X=M:W=W-1:GOSUB550:IFW>52THEND=A:M=B:YR=C:
W=Q:D$="":GOTO1720
AG 1710 TB(CR,W)=TB(CR,0)*VAL(D$):GOTO1690
PJ 1720 GOSUB210:GOTO350
PE 1730 IFW<>CWTHENW=W+1:GOSUB580:GOTO1730

```

```

JX 1740 A=D:B=M:C=YR:W=W+1:GOSUB580:X=M:FL=0
CE 1750 GOSUB580:W=W+1:IFX=MTHEN1750
GF 1760 GOSUB550:W=W-1:IFW>50THENFL=-1:GOTO1760
KQ 1770 CC=1:CW=W:IFFLTHENC=3:CW=52
FQ 1780 GOSUB210:GOTO350
SH 1790 IFW<>CWTHENW=W+1:GOSUB580:GOTO1790
GE 1800 A=D:B=M:C=YR:Q=W
MR 1810 GOSUB580:W=W+1:IFTB(NR,W)>=0ANDW<>50THEN1810
MR 1820 IFTB(NR,W)>=0THEND=A:M=B:YR=C:W=Q:GOTO350
SX 1830 CC=1:CW=W
MA 1840 GOSUB210:GOTO350
JH 1850 DIMTB(18,53),TB$(18),MN$(12),MN(12)
KR 1860 DEBT$=CHR$(28):FG$=CHR$(159):HM$=CHR$(19):
CL$=CHR$(147):UP$=CHR$(145)
AP 1870 DN$=CHR$(17):LF$=CHR$(157):RT$=CHR$(29):CR
$=CHR$(13):BK$=CHR$(20)
QD 1880 RV$=CHR$(18):NV$=CHR$(146):WH$=CHR$(5)+RV$
MS 1890 POKE53280,3:POKE53281,0:PRINTFG$
BQ 1900 REM FOR PLUS 4, PRIOR LINE IS COLOR 0,4,0:
COLOR 4,4,7
HE 1910 PRINTFG$:FORI=1TO12:READMNT$(I),MNT(I):NEX
T
MP 1920 DATA -JAN,31,-FEB,28,-MAR,31,-APR,30
KE 1930 DATA -MAY,31,-JUN,30,-JUL,31,-AUG,31
FF 1940 DATA -SEP,30,-OCT,31,-NOV,30,-DEC,31
AM 1950 W=1:F=10:DP=-1:NW=52
BJ 1960 SP$="{39 SPACES}"
QX 1970 X$="":FORI=1TO40:X$=X$+RT$:NEXT
XC 1980 Y$="":FORI=1TO24:Y$=Y$+DN$:NEXT
JM 1990 A=FRE(X)
DC 2000 NR=1:CW=1:CR=0:CC=1
JJ 2010 TB$(0)="PREVBAL":TB(0,0)=1
PS 2020 TB$(1)="TOTAL"
AE 2030 GOSUB2070
RX 2040 GOSUB350
FF 2050 IFD$=" "THEN2040
AX 2060 TB(CR,CW)=TB(CR,0)*VAL(D$):D$="":X=0:Y=1:G
OSUB110:PRINTSP$:GOTO2040
HR 2070 A$="BUDGET PLANNER":X=39:GOSUB160:PRINT"
{CLR}";FG$:A$
GD 2080 PRINT:PRINTEA$:F$="":EA$=""
SP 2090 INPUT"ENTER FILENAME";F$
DR 2100 IFF$=" "THEN2170
XK 2110 GOSUB890
QR 2120 IFEATHEN2070
BE 2130 RETURN
BD 2140 X=Y0:MNTH(2)=28:IFM0>2THENX=X+1
PM 2150 IFINT(X/4)=X/4THENMN(2)=29
JK 2160 RETURN

```

```
SH 2170 A$="BUDGET PLANNER":X=39:GOSUB160:PRINT "
{CLR}";FG$;A$
PP 2180 PRINTD$:INPUT"DAY{ 3 SPACES}";DD$:D=VAL(DD$
)
HC 2190 INPUT"MONTH ";MM$:MM$="-"+MM$
QK 2200 INPUT"YEAR{ 2 SPACES}";YR$:YR=VAL(YR$):Y0=Y
R
FA 2210 M=0:FORI=1TO12:IFMN$(I)=MM$THENM=I
FK 2220 NEXT:I FM=0THEN2280
BM 2230 IFYR<1910ORYR>1999THEN2290
DA 2240 GOSUB2140
CD 2250 IFD<1ORD>MNTH(M)THEN2270
HH 2260 D$="":GOSUB210:M0=M:D0=D:Y0=YR:RETURN
BJ 2270 D$="{RVS}ILLEGAL DAY VALUE TRY AGAIN{OFF}"
:GOTO2170
QA 2280 D$="{RVS}ILLEGAL MONTH - USE 3-CHAR ABREVI
ATION{OFF}":GOTO2170
QR 2290 D$="{RVS}ILLEGAL YEAR - (1910-1999 ONLY)
{OFF}":GOTO2170
XF 2300 INPUT#15,EA,EA$,A,A:EA=(EA<>0):RETURN
```

Address Cataloger

Kevin Mykytyn

There are other address file programs for the Commodore 64, but we haven't seen one this fast, versatile, and easy to use. "Address Cataloger" offers a number of useful features, including a directory, string search and multiple search, alphabetical sort, and several printout formats. Menu operations make it especially easy to manage.

"Address Cataloger" is a specialized database designed to maintain an up-to-date file of names, addresses, telephone numbers, and any special notes you may want to add for each entry. Written for the Commodore 64, it includes a number of useful features not frequently found in programs of this kind:

- A view of the disk directory
- An instant search for any group of characters
- Alphabetical sorting
- A printer option with your choice of three information formats.

And it's written entirely in machine language—so all operations are extremely fast.

Typing It In

To enter the program, you'll need "MLX," the machine language entry program found in Appendix D. Be sure you read and understand the instructions for MLX before you begin entering the data for Address Cataloger.

When you first run MLX, you're asked for the starting and ending addresses. The correct values for Address Cataloger are

Starting Address: 0801

Ending Address: 1458

Next, begin entering the data for Address Cataloger. When you've finished typing it in, be sure to use the MLX Save option to make at least one copy of the program. To use Address Cataloger, load it and type **RUN** as you would for a BASIC program.

The Menu

When you run Address Cataloger, a menu appears:

```
LOAD DATA
SAVE DATA
DIRECTORY
VIEW DATA
SEARCH DATA
ALPHABETIZE
PRINT DATA
QUIT
```

A red highlighting bar appears over the first choice, **LOAD DATA**. To make a selection, use the up-/down-cursor key to move the bar to one of the items, and press RETURN. Let's look at each of the menu choices and see how to use them.

LOAD DATA: Use this option to load a previously saved address file. (If you're using a disk, it's a good idea to save your files on the same disk as Address Cataloger—you'll be able to check the filenames by selecting **DIRECTORY**, the third choice on the menu.) After pressing RETURN, you're prompted to press T for tape or D for disk. You're then prompted to enter a filename—up to 12 characters are accepted. Disk users may use the wildcard (*) convention for filenames; **KIW*** will load the first file beginning with the letters **KIW**, for example, whether it's **KIWANIS** or **KIWIFRUIT**. If you wish to abort the load, just press RETURN without entering a filename—you're automatically returned to the menu.

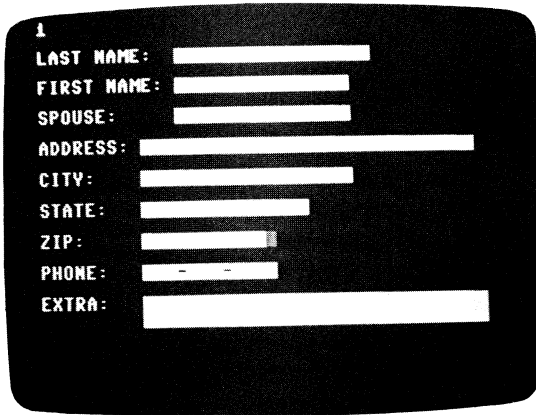
*Note: If you have an address file in memory, it's best if you press RUN/STOP-RESTORE, then type **RUN**, before loading another address file.*

SAVE DATA: After you've entered data and want to save it, cursor to this item on the menu. The procedure is identical to **LOAD DATA**, described above. If you're saving a large file, make sure you have enough room on the disk or tape. Address Cataloger allows a maximum of 230 records per file. (Disk users should note that 230 records will use 182 blocks of disk space.) If you save a file to disk with the same name as a previously saved file, the new file will replace the old one.

DIRECTORY: This option is for disk users only. When you press RETURN, the names of files created and saved by Ad-

dress Cataloger are displayed. Unrelated filenames aren't displayed. If a large number of filenames appear and begin scrolling, pause the list by pressing the SHIFT or CTRL keys.

VIEW DATA: This puts you in the data entry and editing section of the program. After you've pressed RETURN, a screen appears:



1
LAST NAME: _____
FIRST NAME: _____
SPOUSE: _____
ADDRESS: _____
CITY: _____
STATE: _____
ZIP: _____
PHONE: _____
EXTRA: _____

Notice the record number displayed in the upper left corner of the screen. There are 18 key combinations you can use for a variety of editing functions:

CTRL-A. Abort. If you're in the process of editing a record and decide to keep the original data, CTRL-A (the CTRL key and A pressed simultaneously) aborts the current editing changes and restores the data most recently saved for that record.

CTRL-D. Delete. This deletes the record currently displayed and numerically adjusts all subsequent records. For example, if you delete record 15, record 16 becomes record 15, 17 becomes 16, and so on.

CTRL-I. Insert. This moves the displayed record and all subsequent records up one (15 becomes 16, and so on) so you can insert a new record—for example, Morris between Miller and Muller. The Insert function won't work if the file is full (if it already contains 230 records).

- CTRL-F.** Find. This option can be used only after SEARCH DATA is selected from the menu. It displays the next record that matches the *search string* (see SEARCH DATA below).
- CTRL-P.** Print. This option can be used only after PRINT DATA is selected from the menu. It prints the displayed record in the selected format (see PRINT DATA).
- SHIFT-+.** Next Record. By pressing the SHIFT and plus-sign (+) keys, the current record is stored in memory and the next record is displayed.
- SHIFT-—.** Previous Record. The SHIFT and minus-sign (—) key combination enters the current record into memory and displays the previous record.
- Commodore-+ or Commodore-—.** Skip Ten Records. Press the Commodore key and the + or — key to skip ahead or back ten records. For example, if you're on record 36, Commodore-+ will display record 46. In the same situation, Commodore-— will display record 26.
- DEL.** Backspace. Press the DELEte key to erase the character to the left of the cursor.
- ←.** Delete. The left-arrow key (the upper left key on your keyboard) moves all characters to the right of the cursor one space to the left.
- INST.** Insert. This key (SHIFT-INST/DEL) moves all characters under and to the right of the cursor one space to the right.
- RETURN.** Next Field. Press RETURN to move the cursor to the beginning of the next field.
- CRSR-DOWN.** Next Field. Same as RETURN.
- CRSR-UP.** Previous Field. This key moves the cursor to the beginning of the previous field.
- RUN/STOP.** Return to Menu. This key enters the currently displayed record into memory and then displays the menu. (If SEARCH DATA was selected from the menu, RUN/STOP activates the search—see SEARCH DATA below.)
- HOME.** Clear Field. This key clears the field containing the cursor.
- CLR/HOME.** Clear Record. This key (SHIFT-CLR/HOME) erases all data in the record displayed. (It can be restored immediately with CTRL-A.)

SEARCH DATA: When you choose to search through the file, the data entry screen appears, as previously shown. Enter any combination of characters—a *search string*—in any field. Just type in the string you want to search for in the appropriate field and press the RUN/STOP key. (For example, a MAS in the Last Name field will find THOMAS, a 414 in the Phone field will find anyone in the 414 calling area, or CHRISTMAS in the Extra field will find anyone you wish to include on your Christmas card list—if you’ve set up your file for this purpose.)

When a match is found, the record is displayed. If you wish, you can edit any part of the record. To find additional matches, press CTRL-F. If no more matches are found, you’ll notice that the record number has not changed. You can do multiple searches also—for example, SMITH, LOS ANGELES in their respective fields.

If no matches are found, the screen displays NO MATCH and PRESS ANY KEY. The menu reappears after you’ve pressed a key.

ALPHABETIZE: This option sorts all records alphabetically by last name. In the case of identical last names, the sort looks at first names and alphabetizes accordingly. For example, if you have five Smiths in your address file, they’ll be alphabetized under Smith by first name: Al, Dorothy, Mike, Richard, and Walter.

You can also use this option to clean up your file from time to time. After you’ve used the file for a while, you’re likely to have some blank records. By alphabetizing, you’ll condense your file—the program removes all blank records and renumbers existing files, thus saving space. (It’s a good idea to avoid blank records when entering records—a lot of these in your file will slow down program operation.)

PRINT DATA: Before choosing any options from this menu item, be sure your printer is turned on and the paper properly aligned. There are several choices to make before printing out data from an address file. After you’ve selected this option, a small menu appears:

ALL INFORMATION
PHONE LIST
ADDRESS LABELS
MAIN MENU

The first choice, ALL INFORMATION, prints all the data in a file. The PHONE LIST selection prints all data except the information in the Extra field. ADDRESS LABELS prints all data in the Name, Address, City, State, and ZIP fields. When printing address labels, you can adjust the printer linefeed by typing **POKE 5176,x**, where *x* is the number of linefeeds you want. As written, the program uses six linefeeds. If you wish to make a permanent adjustment, load the program, type this POKE in immediate mode, and resave the program with a new filename. The final choice, MAIN MENU, provides an easy way to exit from the print option.

MICHAEL SANTORO	215-555-0719
LISA	
489 CRESTWOOD DRIVE	
MERCERVILLE, PA	
19027	
CHRISTMAS CARD/BIRTHDAY: JULY 12	
CHILDREN: CHRIS AND DANNY	

MICHAEL SANTORO	215-555-0719
LISA	
489 CRESTWOOD DRIVE	
MERCERVILLE, PA	
19027	

MICHAEL SANTORO
489 CRESTWOOD DRIVE
MERCERVILLE, PA
19027

Three printing formats are available with "Address Cataloger": The first includes all information (name, address, phone, and two lines of extra information), the second prints all data except the extra information, and the third format, for address labels, prints a name and address only.

*Note: Certain printers print a few garbage (worthless) characters just before the name field. If you encounter this, **POKE 3173,255** before running the program. Also, if you'd rather print the ZIP code on the same line as the state, **POKE 5173,32**. If you own a daisywheel printer, you may find that underlining causes a series of C's to be printed. To fix this, **POKE 3371,45**. To make any of these changes permanent, load Address Cataloger, enter the POKes, and save the program back to disk under a new name.*

After you've selected one of the three printing formats, another menu is displayed:

SELECTED
ALL
MAIN MENU

The first choice, **SELECTED**, lets you choose which record or records will be printed. You're presented with the data entry screen (as in the **SEARCH DATA** option). Enter a search string in the appropriate field(s), and press **RUN/STOP**. All records that match are then printed out. If you wish to stop the printing after any record, press **RUN/STOP**.

If the **ALL** option is chosen, all records are printed out. Again, **RUN/STOP** stops the procedure.

If you've selected a printing format (**ALL INFORMATION**, **PHONE LIST**, or **ADDRESS LABELS**), and then go to the main menu (ignoring **SELECTED** or **ALL**), you can then print out individual records from within the **VIEW DATA** option by pressing **CTRL-P**.

QUIT: This option exits the program and returns you to **BASIC**.

Address Cataloger

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```
0801:FF FF 0A 00 9E 32 30 36 A6
0809:31 00 00 00 A9 36 85 01 E3
0811:20 92 0D 20 BB 12 A5 8B 76
0819:0A AA BD 2D 08 8D 28 08 32
0821:BD 2E 08 8D 29 08 20 FF 1F
0829:FF 4C 14 08 A3 08 CF 09 35
0831:4B 0A 3D 08 4A 08 0D 0B 29
0839:E7 0B 5E 0D 20 C5 10 20 F4
0841:70 0F A9 00 85 04 4C 8E E5
0849:0E A9 00 2C A9 01 85 39 23
0851:20 C5 10 20 FA 0D A9 01 47
0859:85 04 20 8E 0E 20 1A 0E 4D
0861:A9 FF 85 FB 20 25 0E A5 0E
0869:39 F0 01 60 E0 E6 D0 CC 89
0871:20 93 08 A2 0B A0 10 18 B4
0879:20 F0 FF A9 89 A0 08 20 6F
0881:FB 0A 20 D5 0A E6 FB 60 B7
0889:1F 4E 4F 20 4D 41 54 43 04
0891:48 00 A9 0F 8D 20 D0 8D 08
```

0899:21 D0 8D 86 02 A9 93 4C B2
08A1:40 14 A9 00 85 8B 20 C4 6B
08A9:08 B0 17 A9 00 A2 70 A0 73
08B1:17 20 D5 FF A5 0C C9 01 02
08B9:F0 05 20 05 0A F0 03 20 12
08C1:D5 0A 60 A2 05 BD FF 09 9D
08C9:9D C0 02 CA 10 F7 20 93 F9
08D1:08 A2 05 A0 00 20 F0 FF 9B
08D9:A9 B8 A0 09 20 FB 0A A9 40
08E1:08 85 0C 20 E4 FF C9 44 D9
08E9:F0 0C C9 54 D0 F5 A9 01 A6
08F1:85 0C A9 00 85 8B A9 A6 51
08F9:A0 09 20 FB 0A 20 97 09 69
0901:A0 00 84 F7 20 E4 FF A4 AC
0909:F7 C9 0D F0 27 C9 14 F0 B3
0911:1C C9 20 90 ED C9 5B B0 AE
0919:E9 99 C6 02 C8 C0 0C F0 D1
0921:13 20 40 14 20 97 09 8C 0C
0929:3C 03 4C 03 09 C0 00 F0 10
0931:D1 88 10 ED C0 00 F0 5C 73
0939:20 46 0A 20 43 0A A5 8B 49
0941:F0 3C 98 48 A9 02 A2 CD DA
0949:A0 09 20 BD FF A9 0F A8 3B
0951:A2 08 20 BA FF 20 C0 FF 68
0959:20 05 0A A9 0F 20 C3 FF 19
0961:68 18 69 06 A2 C0 A0 02 96
0969:20 BD FF A9 0F A8 A2 08 FD
0971:20 BA FF 20 C0 FF 20 05 8F
0979:0A A9 0F 20 C3 FF A9 01 51
0981:A8 A6 0C 20 BA FF A5 F7 2E
0989:18 69 03 A2 C3 A0 02 20 51
0991:BD FF 18 60 38 60 98 48 48
0999:A9 A3 A0 09 20 FB 0A 68 7B
09A1:A8 60 AF 9D 00 1F 45 4E 45
09A9:54 45 52 20 46 49 4C 45 B8
09B1:4E 41 4D 45 3A 20 00 1C A7
09B9:28 54 29 41 50 45 20 4F 55
09C1:52 20 28 44 29 49 53 4B AE
09C9:0D 0D 0D 00 49 30 A9 01 A6
09D1:85 8B 20 C4 08 B0 26 20 49
09D9:C4 0B E8 20 60 10 A6 02 C2
09E1:A4 03 A9 70 85 02 A9 17 E1
09E9:85 03 A9 02 20 D8 FF A5 DE
09F1:0C C9 01 F0 05 20 05 0A 68
09F9:F0 03 20 D5 0A 60 53 30 4F
0A01:3A A6 A6 A6 A5 90 F0 39 A5
0A09:20 43 0A 20 43 0A A9 0F E5
0A11:A2 08 A8 20 BA FF A9 00 B8
0A19:20 BD FF 20 C0 FF A2 0F 09
0A21:20 C6 FF 20 CF FF 48 20 28

0A29:40 14 68 C9 0D D0 F4 A9 4B
0A31:02 20 C3 FF A9 0F 20 C3 54
0A39:FF 20 CC FF 20 43 0A A9 BA
0A41:01 60 A9 0D 2C A9 20 4C 88
0A49:40 14 A9 93 20 40 14 A9 C4
0A51:1F 20 40 14 A9 02 A2 08 E8
0A59:A0 00 20 BA FF A2 CE A0 36
0A61:0A A9 08 20 BD FF 20 C0 D6
0A69:FF A2 02 20 C6 FF 20 CF AE
0A71:FF 20 05 0A D0 38 20 CF 46
0A79:FF 20 CF FF 85 F8 20 CF AF
0A81:FF 05 F8 F0 29 20 CF FF 6E
0A89:85 F8 20 CF FF A6 F8 20 4C
0A91:BA 0A 20 CF FF C9 A6 F0 EB
0A99:F9 C9 00 F0 06 20 40 14 71
0AA1:4C 93 0A AD 8D 02 D0 FB EE
0AA9:20 43 0A 4C 7A 0A A9 02 F5
0AB1:20 C3 FF 20 CC FF 4C D5 9D
0AB9:0A 48 A9 37 85 01 68 20 AE
0AC1:CD BD 20 46 0A 20 46 0A FB
0AC9:A9 36 85 01 60 24 30 3A 2F
0AD1:A6 A6 A6 2A A2 18 A0 0E 1F
0AD9:18 20 F0 FF A9 EA A0 0A 64
0AE1:20 FB 0A 20 E4 FF F0 FB 4D
0AE9:60 12 1C 50 52 45 53 53 DC
0AF1:20 41 4E 59 20 4B 45 59 D7
0AF9:92 00 85 05 84 06 A0 FF D5
0B01:C8 B1 05 F0 06 20 40 14 DC
0B09:4C 01 0B 60 20 C4 0B E0 F7
0B11:01 B0 01 60 86 8D A9 00 B7
0B19:85 8B AA EE 20 D0 A9 70 21
0B21:85 02 A9 17 85 03 A5 02 A6
0B29:85 05 18 69 C8 85 02 A5 E2
0B31:03 85 06 69 00 85 03 A0 3E
0B39:FF C8 C0 25 F0 1F B1 02 55
0B41:D1 05 F0 F5 B0 17 A0 C7 E9
0B49:B1 02 85 F7 B1 05 91 02 AF
0B51:A5 F7 91 05 88 C0 FF D0 D2
0B59:EF A9 01 85 8B E8 E4 8D A1
0B61:D0 C4 C6 8D F0 04 A5 8B 31
0B69:D0 AC 60 A6 FB E0 E5 F0 A9
0B71:1A 20 60 10 A5 02 85 C1 AB
0B79:A5 03 85 C2 E8 20 60 10 98
0B81:A5 02 85 BD A5 03 85 BE 7A
0B89:20 66 13 20 92 0B 4C 81 88
0B91:10 A2 E5 4C CA 0D 20 C4 69
0B99:0B E0 E5 F0 25 A6 FB E0 D5
0BA1:E5 F0 1F 20 60 10 A5 02 5D
0BA9:85 BD A5 03 85 BE E8 20 EF
0BB1:60 10 A5 02 85 C1 A5 03 52

ØBB9:85 C2 20 A6 13 A6 FB 4C 29
ØBC1:D1 0F 60 A2 FF E8 E0 E6 07
ØBC9:F0 19 20 60 10 A0 C8 88 C5
ØBD1:C0 FF F0 F1 B1 02 C9 A0 4F
ØBD9:F0 F5 C9 AD F0 F1 86 F7 4E
ØBE1:4C C6 0B A6 F7 60 A9 00 30
ØBE9:85 3A A2 4C A0 12 A9 04 0F
ØBF1:20 C1 12 A5 8B C9 03 D0 7F
ØBF9:01 60 85 28 A2 93 A0 12 92
ØC01:A9 03 20 C1 12 A5 8B F0 FD
ØC09:1B C9 01 F0 01 60 20 C4 DE
ØC11:0B E8 86 8F A2 00 20 49 51
ØC19:0C 20 E1 FF F0 05 E8 E4 CD
ØC21:8F D0 F3 60 20 4D 08 E0 E0
ØC29:E6 F0 1C 20 81 10 A6 FB 0C
ØC31:20 49 0C 20 25 0E E0 E6 39
ØC39:F0 0D 20 81 10 A6 FB 20 5C
ØC41:49 0C 20 E1 FF D0 EC 60 A0
ØC49:8A 48 20 60 10 A9 00 85 6F
ØC51:8D A5 28 0A AA BD 3A 14 14
ØC59:8D 5B 0D BD 3B 14 8D 5C 2E
ØC61:0D A9 01 A0 00 A2 04 20 47
ØC69:BA FF 20 C0 FF A2 01 20 9B
ØC71:C9 FF A5 28 C9 02 F0 16 F3
ØC79:A5 3A C9 38 90 10 49 FF 07
ØC81:18 69 43 AA 20 43 0A CA FF
ØC89:D0 FA A9 01 85 3A A2 FF 68
ØC91:E8 20 5A 0D C9 FD D0 03 2D
ØC99:4C 22 0D C9 FC D0 09 A9 85
ØCA1:32 38 E5 8D 85 F7 D0 0A 2E
ØCA9:C9 FF D0 0F E8 20 5A 0D 3B
ØCB1:85 F7 20 46 0A C6 F7 D0 1F
ØCB9:F9 F0 D5 C9 FE D0 11 E8 A8
ØCC1:20 5A 0D 85 F7 F0 C9 20 B1
ØCC9:43 0A C6 F7 D0 F9 F0 C0 6F
ØCD1:C9 0A 90 06 20 40 14 4C 3A
ØCD9:91 0C A8 B9 83 0E 48 B9 0D
ØCE1:84 0E 85 F7 68 A8 20 40 56
ØCE9:0D A5 8D 18 65 8C 85 8D 1B
ØCF1:E6 8C C8 C4 8C D0 0A E8 AA
ØCF9:20 5A 0D 10 01 CA 4C 91 B8
ØD01:0C B1 02 C9 AD F0 EB C9 3D
ØD09:A0 F0 E7 88 C8 C4 8C F0 98
ØD11:ED B1 02 29 3F C9 1B B0 69
ØD19:02 09 40 20 40 14 4C 0D 78
ØD21:0D A5 28 C9 02 F0 0D A0 5B
ØD29:32 A9 C3 20 40 14 88 D0 75
ØD31:F8 20 43 0A A9 01 20 C3 2E
ØD39:FF 20 CC FF 68 AA 60 98 3C
ØD41:48 C8 84 8C 88 E6 F7 C8 A3

0D49:C4 F7 D0 03 68 A8 60 B1 66
0D51:02 C9 A0 F0 F2 84 8C D0 95
0D59:EE BD FF FF 60 A9 37 85 F7
0D61:01 00 A6 FB 20 60 10 A2 D5
0D69:00 BD 96 11 85 05 BD 97 2A
0D71:11 85 06 A0 00 B1 05 10 21
0D79:05 91 02 C8 10 F7 98 18 F0
0D81:65 02 85 02 A5 03 69 00 AB
0D89:85 03 E8 E8 E0 14 D0 D9 A5
0D91:60 A9 8E 20 40 14 A9 08 C7
0D99:20 40 14 A9 00 85 FE 85 8A
0DA1:FB A9 14 85 FA A9 80 8D 0C
0DA9:8A 02 78 A9 E3 8D 14 03 B3
0DB1:A9 0D 8D 15 03 58 A9 70 24
0DB9:85 02 A9 17 85 03 A2 E5 21
0DC1:20 CA 0D CA E0 FF D0 F8 8E
0DC9:60 20 60 10 A0 C7 A9 A0 41
0DD1:91 02 88 C0 FF D0 F9 A9 33
0DD9:AD A0 81 91 02 A0 85 91 6B
0DE1:02 60 A5 FE F0 10 C6 FA 0A
0DE9:D0 0C A9 14 85 FA A0 00 3F
0DF1:B1 FC 49 80 91 FC 4C 31 9F
0DF9:EA A9 00 85 FE 85 FF A8 03
0E01:B1 FC 09 80 91 FC 20 0F 2E
0E09:0E A9 01 85 FE 60 AD 96 7A
0E11:11 85 FC AD 97 11 85 FD 9B
0E19:60 A9 34 85 02 A9 03 85 F0
0E21:03 4C 68 0D A6 FB E8 20 C6
0E29:60 10 A0 09 86 62 B9 83 D2
0E31:0E 85 8E AA B9 84 0E 85 B3
0E39:8F 84 61 E8 E4 8F F0 36 76
0E41:BD 34 03 C9 A0 F0 F4 86 7F
0E49:8D A4 8E A6 8D C8 C4 8F 3A
0E51:F0 1C BD 34 03 D1 02 D0 1C
0E59:F4 84 8E C8 E8 E4 8F F0 5A
0E61:15 BD 34 03 C9 A0 F0 F3 D4
0E69:D1 02 D0 DD F0 ED A6 62 D5
0E71:E8 E0 E6 D0 B2 60 A4 61 E5
0E79:88 10 B3 A6 62 86 FB 4C 30
0E81:81 10 FF 11 21 31 4F 62 42
0E89:71 7D 89 A8 C7 A9 00 85 E3
0E91:FF 20 FA 0D 20 E4 FF F0 6B
0E99:FB C9 2D D0 06 A6 FF E0 84
0EA1:07 F0 F1 A2 00 DD 24 10 B5
0EA9:F0 15 E8 E0 14 D0 F6 29 A9
0EB1:3F 09 80 A0 00 91 FC A9 B3
0EB9:01 20 DA 0F 4C 95 0E A5 25
0EC1:04 F0 04 E0 0B B0 CD A9 0B
0EC9:00 A8 85 FE B1 FC 09 80 C4
0ED1:91 FC 8A 0A AA BD 38 10 B4

0ED9:8D E3 0E BD 39 10 8D E4 5D
0EE1:0E 20 FF FF A9 01 85 FE 68
0EE9:4C 95 0E C6 FF 10 04 A9 B1
0EF1:09 85 FF 4C 03 0F E6 FF DA
0EF9:A5 FF C9 0A 90 04 A9 00 AA
0F01:85 FF A5 FF 0A AA BD 96 A3
0F09:11 85 FC BD 97 11 85 FD 96
0F11:60 68 68 A5 04 D0 03 20 6A
0F19:63 0D A9 00 85 FE 60 A9 F3
0F21:FF 4C DA 0F A9 01 4C DA 63
0F29:0F 20 63 0D A5 FB C9 E5 AA
0F31:F0 43 E6 FB 4C 70 0F 60 D7
0F39:20 63 0D A5 FB C9 DC 90 8D
0F41:04 A9 E5 D0 05 A5 FB 18 64
0F49:69 0A 85 FB 4C 70 0F 20 71
0F51:63 0D A5 FB C9 0A B0 04 B4
0F59:A9 00 F0 05 A5 FB 38 E9 32
0F61:0A 85 FB 4C 70 0F 20 63 8D
0F69:0D A5 FB F0 08 C6 FB 20 79
0F71:81 10 20 FA 0D 60 20 20 52
0F79:0F A9 A0 91 FC 60 A0 00 61
0F81:C8 B1 FC 10 07 88 91 FC 8B
0F89:C8 4C 81 0F 88 A9 A0 91 FD
0F91:FC 60 A0 FF C8 B1 FC 30 91
0F99:FB 88 88 30 08 B1 FC C8 B5
0FA1:91 FC 4C 9A 0F A0 00 F0 E6
0FA9:E4 A6 FF 4C AF 0F 8A 48 BF
0FB1:0A AA BD 96 11 85 05 BD 07
0FB9:97 11 85 06 A0 FF C8 B1 41
0FC1:05 10 0A C9 AD F0 F7 A9 0F
0FC9:A0 91 05 D0 F1 68 AA 60 31
0FD1:A2 09 20 AF 0F CA 10 FA 41
0FD9:60 85 F8 85 F9 30 04 A9 43
0FE1:00 85 F9 A9 00 85 FE A5 F4
0FE9:FC 48 A5 FD 48 A5 F8 18 10
0FF1:65 FC 85 FC A5 FD 65 F9 6C
0FF9:85 FD B1 FC 10 10 C9 AD 62
1001:D0 15 A5 FF C9 07 D0 0F 9E
1009:20 E4 0F 4C 18 10 68 85 70
1011:FD 68 85 FC 4C 1A 10 68 1E
1019:68 A9 01 85 FE 60 A6 FB 13
1021:4C 49 0C 91 11 9D 1D 0D 9A
1029:14 94 5F 13 93 03 DB DD D3
1031:06 04 09 10 01 A6 DC EC C0
1039:0E F7 0E 20 0F 25 0F F7 45
1041:0E 77 0F 93 0F 7F 0F AA A0
1049:0F D1 0F 12 0F 2A 0F 67 0F
1051:0F 25 0E 6C 0B 97 0B 1F B6
1059:10 81 10 39 0F 50 0F 8A D9
1061:48 A9 70 85 02 A9 17 85 E0

1069:03 E0 00 F0 10 A5 02 18 85
1071:69 C8 85 02 A5 03 69 00 55
1079:85 03 CA D0 F0 68 AA 60 62
1081:A6 FB 20 60 10 A2 00 BD C6
1089:96 11 85 05 BD 97 11 85 2E
1091:06 A0 00 B1 05 10 07 B1 20
1099:02 91 05 C8 10 F5 98 18 ED
10A1:65 02 85 02 A5 03 69 00 D1
10A9:85 03 E8 E8 E0 14 D0 D7 C9
10B1:A9 13 20 40 14 A6 FB E8 8F
10B9:A9 00 4C BA 0A E6 02 D0 A4
10C1:02 E6 03 60 20 0F 0E 20 7C
10C9:93 08 A9 0B 8D 20 D0 8D B7
10D1:21 D0 A2 00 BD 0E 11 85 D8
10D9:02 E8 BD 0E 11 85 03 A0 13
10E1:00 E8 A5 02 05 03 F0 24 4B
10E9:BD 0E 11 F0 08 29 BF 91 93
10F1:02 E8 C8 10 F3 E8 C8 BD F9
10F9:0E 11 86 F8 AA F0 08 A9 98
1101:A0 91 02 C8 CA 10 FA A6 D7
1109:F8 E8 D0 C8 60 50 04 4C 21
1111:41 53 54 20 4E 41 4D 45 8C
1119:3A 20 00 11 A0 04 46 49 5C
1121:52 53 54 20 4E 41 4D 45 25
1129:3A 00 0F F0 04 53 50 4F B6
1131:55 53 45 3A 20 20 20 20 01
1139:00 0F 40 05 41 44 44 52 6D
1141:45 53 53 3A 00 1D 90 05 83
1149:43 49 54 59 3A 20 20 20 32
1151:00 12 E0 05 53 54 41 54 27
1159:45 3A 20 20 00 0E 30 06 51
1161:5A 49 50 3A 20 20 20 20 92
1169:00 0B 80 06 50 48 4F 4E 4F
1171:45 3A 20 20 20 A0 A0 A0 30
1179:AD A0 A0 A0 AD A0 A0 A0 8A
1181:A0 00 00 D0 06 45 58 54 4B
1189:52 41 3A 20 20 00 1E 00 AB
1191:07 00 1E 00 00 5C 04 AC 21
1199:04 FC 04 49 05 99 05 E9 94
11A1:05 39 06 89 06 D9 06 01 92
11A9:07 93 0D 0D 00 0B 1F 41 52
11B1:44 44 52 45 53 53 20 43 11
11B9:41 54 41 4C 4F 47 45 52 F2
11C1:0D 00 0B A3 A3 A3 A3 A3 9C
11C9:A3 A3 A3 A3 A3 A3 A3 A3 EB
11D1:A3 A3 A3 A3 0D 0D 0D 00 14
11D9:0D 1C 4C 4F 41 44 20 44 A7
11E1:41 54 41 FE 00 0D 53 41 ED
11E9:56 45 20 44 41 54 41 FE AD
11F1:00 0D 44 49 52 45 43 54 F6

11F9:4F 52 59 FE 00 0D 56 49 9D
1201:45 57 20 44 41 54 41 FE C2
1209:00 0D 53 45 41 52 43 48 51
1211:20 44 41 54 41 FE 00 0D D6
1219:41 4C 50 48 41 42 45 54 71
1221:49 5A 45 FE 00 0D 50 52 40
1229:49 4E 54 20 44 41 54 41 23
1231:FE 00 0D 51 55 49 54 9B 9F
1239:21 1C FF FE 04 4E 05 9E E6
1241:05 EE 05 3E 06 8E 06 DE 7D
1249:06 2E 07 93 0D 0D 0D 0D D9
1251:0D 0D 00 0D 1C 41 4C 4C DA
1259:20 49 4E 46 4F 52 4D 41 AD
1261:54 49 4F 4E FE 00 0D 50 33
1269:48 4F 4E 45 20 4C 49 53 BB
1271:54 FE 00 0D 41 44 44 52 46
1279:45 53 53 20 4C 41 42 45 B2
1281:4C 53 FE 00 0D 4D 41 49 E9
1289:4E 20 4D 45 4E 55 9B 21 FA
1291:1C FF 93 0D 0D 0D 0D 0D CA
1299:0D 00 0D 1C 53 45 4C 45 35
12A1:43 54 45 44 FE 00 0D 41 BC
12A9:4C 4C FE 00 0D 4D 41 49 50
12B1:4E 20 4D 45 4E 55 9B 21 23
12B9:1C FF A2 AA A0 11 A9 08 8F
12C1:85 63 86 26 84 27 20 93 49
12C9:08 A0 FF C8 B1 26 D0 0E 7C
12D1:C8 B1 26 AA A9 20 20 40 84
12D9:14 CA 10 F8 30 ED C9 FF 19
12E1:F0 1C C9 FE D0 12 8A 48 DA
12E9:A2 04 BD 61 13 20 40 14 DB
12F1:CA 10 F7 68 AA 4C CC 12 37
12F9:20 40 14 4C CC 12 A2 00 79
1301:86 8B 20 43 13 20 E4 FF 68
1309:C9 91 D0 0E 20 43 13 C6 6E
1311:8B 10 1A A6 63 CA 86 8B 8D
1319:D0 13 C9 11 D0 15 20 43 15
1321:13 E6 8B A5 8B C5 63 90 21
1329:04 A9 00 85 8B 20 43 13 8A
1331:4C 06 13 C9 0D D0 CE A5 EC
1339:C5 C9 01 F0 FA A9 00 85 E7
1341:C6 60 A5 8B 0A AA BD 3C 03
1349:12 85 02 BD 3D 12 85 03 36
1351:A0 00 B1 02 C9 21 F0 07 D9
1359:49 80 91 02 C8 D0 F3 60 68
1361:0D 0D 1C 21 9B A0 00 20 66
1369:E9 13 A5 BD 8D 8A 13 A5 3C
1371:BE 8D 8B 13 A5 C1 8D 8D D9
1379:13 A5 C2 8D 8E 13 A6 BC 8E
1381:F0 1E A9 00 85 BF A0 00 49

1389:B9 FF FF 99 FF FF C8 C4 7C
1391:BF D0 F5 EE 8B 13 EE 8E 8E
1399:13 E0 00 F0 07 CA D0 E2 78
13A1:A5 BB D0 E0 60 A0 01 20 59
13A9:E9 13 A5 BC AA 05 BB D0 BB
13B1:01 60 18 8A 65 BE 8D D6 34
13B9:13 A5 BD 8D D5 13 18 8A 19
13C1:65 C2 8D D9 13 A5 C1 8D DA
13C9:D8 13 E8 A4 BB D0 04 F0 A2
13D1:0D A0 FF B9 FF FF 99 FF 75
13D9:FF 88 C0 FF D0 F5 CE D6 0D
13E1:13 CE D9 13 CA D0 EA 60 81
13E9:A2 E6 C0 00 F0 01 CA 20 74
13F1:60 10 A5 02 38 E5 BD 85 7B
13F9:BB A5 03 E5 BE 85 BC 60 0C
1401:01 20 00 FC 14 07 0D 02 5A
1409:0D 03 0D 04 2C 20 05 0D 53
1411:06 0D FD 01 20 00 FC 14 5E
1419:07 0D 02 0D 03 0D 04 2C 99
1421:20 05 0D 06 0D 08 0D 09 48
1429:0D FD 01 20 00 0D 03 0D C0
1431:04 2C 20 05 0D 06 FE 06 3F
1439:FD 14 14 01 14 2B 14 48 B5
1441:A5 9A C9 03 F0 08 68 C9 8E
1449:0D D0 02 E6 3A 48 68 4C EA
1451:D2 FF 4C 00 00 00 00 00 6C

Custom Labels

R. E. Metzler, Jr.

Designing and printing customized mailing labels is easy with this menu-driven program. You can even save designs on disk for future use. For the Commodore 64 and 128 (in 64 mode), with Commodore 1525, 1526, MPS 801, 802, 803, or compatible printers. Disk drive required.

Printing your own customized labels is easy with this menu-driven program. All you have to do is answer the menu prompts and develop your design. "Custom Labels" does the rest. It's written in BASIC for the Commodore 64 and 128 (in 64 mode) with Commodore 1525, MPS 801, 803, or compatible printers. Type in the program and save a copy to disk.

If you have a Commodore 1526 or MPS-802 printer, you'll have to make a few changes to the program. Substitute these lines for those you find in the listing at the end of this article:

```
CF 310 PI$=PI$+SI$:X=X+1:NEXT:PI$=PI$+CHR$(13)
EF 320 PRINT#4,PI$:PI$="":IFP=1THEN X=X+4
```

Menu Choices

After typing in the program, save it to disk. To use Custom Labels, load it and type **RUN**. (Be sure your printer is turned on.)

A two-option menu appears: (1) Graphics Labels or (2) Enhanced Printing Labels. Graphics Labels is for standard-sized characters, and Enhanced Printing prints double-width characters. With either selection, you can use any keyboard characters, including Commodore graphics characters. After you've made your selection, you're presented with the main menu and a workspace for designing your label. The design area is automatically sized for the printing format, and it's centered for standard-sized labels, 3-13/16 inches long.

The main menu offers five choices: (1) Design a Label, (2) Load a Label from Disk, (3) Change the Label, (4) Save the

Label to Disk, and (5) Print the Label. Initially, you'll choose the first option.

After pressing 1, you'll see the workspace with a cursor at the upper left corner. You can begin designing your label. *Use only cursor keys to move around the workspace. Don't use the INST/DEL key.* Also, don't press RETURN after each line. Just use the cursor keys to get to another line (more on RETURN below).

Note the reminders at the bottom of the screen. The workspace size is proportional to the size of the label you'll be printing on, so you'll need to center what you want printed. If you're designing, say, an address label, be sure to leave a blank line between printed lines to make your label more readable.

Any characters may be used in your design, although there's a minor restriction on using commas and colons. Normally, when you've finished designing your label, you press RETURN, as noted at the bottom of the screen. However, if you've used commas or colons, you must press RETURN over a line *without* either of these punctuation marks. Say you're trying to create the following label:

John Doe
123 Main Street, Apt. A-5
Oakwood, CA 90067

You'd type *John Doe*, cursor down and type the street address, cursor down and type the city, state, and zip code, then cursor up to the first line (the only one without a comma), and press RETURN. If you're designing labels and wish to use commas or colons, be sure that there's one line which does not include them.

After you've completed a label design and pressed RETURN, the menu reappears. At this point, you can print out, save (to disk), or edit the label. When saving a design, enter a filename at the prompt. After you've pressed RETURN, the label is saved to disk. It can be reloaded at any time by selecting the load option from the menu. Just enter the filename at the prompt. Should you forget the filename, list the directory of the disk to which you saved the label (it will appear on the directory as the exact name you used to save it).

FRAGILE

**MAGNETIC MEDIA
DO NOT X-RAY
OR EXPOSE TO
MAGNETIC FIELDS**

These labels were produced with a Commodore 64 and MPS-802. The "Fragile" label was created with keyboard graphics characters. The "Magnetic Media" label illustrates the enhanced printing mode (double-wide characters).

Printer Modifications

Some Commodore-emulating printers may require slight program changes to set the line spacing or restore the printer to normal operation. **OPEN 6,4,6: PRINT#6, CHR\$(12)** is used in line 190 to adjust the space between lines on the 1526/802. In line 350, **CLOSE 6: OPEN 10,4,10: PRINT#10: CLOSE 10** is used to reset the 1526/802 printer to normal operation (neither of these commands has any effect on the 1525/801/803, so they were left in). To adjust spacing on the odd-numbered printers (1525/801/803), a **CHR\$(15)** precedes the string to be printed in line 320, and a **CHR\$(8)** follows. The 15 turns on normal (uppercase/graphics) mode. The 8 turns on graphics mode, which eliminates spacing between lines. These two codes cause problems with the 1526 and MPS-802, and must be removed, as noted above.

PI\$=CHR\$(14) is used in line 210 to turn on the enhanced (double-width) printing for each line. If you own a non-Commodore printer, see your printer manual for the equivalent commands, should you have any trouble.

Custom Labels

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

```

GS 10 POKE53280,2:POKE53281,15
XE 20 PRINT"{CLR}{BLK}{10 DOWN}{2 SPACES}PRESS
      {RVS}1{OFF} FOR GRAPHICS LABELS{2 DOWN}"
RJ 30 PRINT"{2 SPACES}PRESS {RVS}2{OFF} FOR ENHANC
      ED PRINTING LABELS"
HQ 40 GETP$:P=VAL(P$):IFP<1ORP>2THEN40
RS 50 PRINT"{CLR}":GOSUB630:X$="{RVS}{40 SPACES}"
KJ 60 PRINT"{HOME}{RED}{8 DOWN}";:FORI=8TO12:PRINT
      X$;:NEXT:PRINT"{OFF}{BLK}"
HS 70 IFP=2THENPRINT"{HOME}{9 RIGHT}{RED}{RVS}
      {2 SPACES}{OFF}"
EK 80 PRINT"{HOME}{RVS}{RED} {OFF}{BLK}":GOSUB680:
      PRINTCP$"{3 SPACES}(1){2 SPACES}DESIGN A SPE
      CIAL LABEL"
JR 90 PRINT"{DOWN}{3 SPACES}(2){2 SPACES}LOAD A LA
      BEL FROM DISK":PRINT"{DOWN}{3 SPACES}(3)
      {2 SPACES}CHANGE THE LABEL ABOVE
ER 100 PRINT"{DOWN}{3 SPACES}(4){2 SPACES}SAVE THE
      LABEL ABOVE TO DISK"
SQ 110 PRINT"{DOWN}{3 SPACES}(5){2 SPACES}PRINT TH
      E LABEL ABOVE":PRINT"{DOWN}{9 SPACES}{RVS}E
      NTER YOUR CHOICE{OFF}
KR 120 GETM$:M=VAL(M$):IFM<1ORM>5THEN120
GH 130 ONMGOTO550,460,550,370
XH 140 REM * PRINT ROUTINE *
EK 150 GOSUB680:PRINTCP$"{3 DOWN}POSITION THE LABE
      L IN THE PRINTER SO"
HS 160 PRINT"THAT THE TOP OF THE LABEL IS AT THE T
      OP"
DE 170 PRINT"OF THE PRINthead AND PRESS RETURN."
MR 180 GETX$:IFX$<>CHR$(13)THEN180
FE 190 OPEN3,3:OPEN4,4:OPEN6,4,6:PRINT#6,CHR$(12):
      Y=4
HK 200 IFP=1THENX=1026:FORI=0TO7:FORJ=3TO38
RF 210 IFP=2THENX=1035:FORI=0TO7:PI$=CHR$(14):FORJ
      =12TO29
FG 220 SI=PEEK(X):IFSI<128ANDY=3THENY=2
DC 230 IFSI>127ANDY=4THENY=1
AA 240 IFSI>127THENSI=SI-128
FP 250 IFSI=127THENSI=191:GOTO290
MP 260 IFSI>95THENSI=SI+32
KK 270 IFSI>63THENSI=SI+32
JF 280 IFSI<32THENSI=SI+64
AM 290 SI$=CHR$(SI):IFY=1THENSI$=CHR$(18)+SI$:Y=3
FS 300 IFY=2THENSI$=CHR$(146)+SI$:Y=4
GS 310 PI$=PI$+SI$:X=X+1:NEXT

```

```
GB 320 PRINT#4,CHR$(15)PI$CHR$(8):PI$="":IF P=1 TH
    EN X=X+4
DQ 330 IFP=2THENX=X+22
HA 340 Y=4:NEXT:PRINT#4,CHR$(10)
AH 350 CLOSE3:CLOSE4:CLOSE6:OPEN10,4,10:PRINT#10:C
    LOSE10:GOTO70
FP 360 REM * SAVE ROUTINE *
XM 370 GOSUB680:PRINTCP$:INPUT"SAVE FILE NAME";NF$
PM 380 IFNF$=""THEN70
XF 390 OPEN3,3:OPEN8,8,8,"0:"+NF$+",S,W"
JS 400 IFP=1THENX=1026:FORI=0TO7:FORJ=3TO38
QR 410 IFP=2THENX=1035:FORI=0TO7:FORJ=12TO29
QG 420 Y=PEEK(X):PRINT#8,Y:X=X+1:NEXT:IFP=1THENX=X
    +4
KH 430 IFP=2THENX=X+22
HJ 440 NEXT:CLOSE3:CLOSE8:GOTO70
DH 450 REM * LOAD ROUTINE *
EE 460 GOSUB680:PRINTCP$:INPUT"LOAD FILE NAME";NF$
CQ 470 IFNF$=""THEN70
FX 480 OPEN3,3:OPEN8,8,8,"0:"+NF$+",S,R"
QF 490 IFP=1THENX=1026:FORI=0TO7:FORJ=3TO38
HP 500 IFP=2THENX=1035:FORI=0TO7:FORJ=12TO29
RE 510 INPUT#8,Y$:Y=VAL(Y$):POKEY,X:X=X+1:NEXT:IFP
    =1THENX=X+4
AJ 520 IFP=2THENX=X+22
RC 530 NEXT:CLOSE3:CLOSE8:GOTO70
KG 540 REM * DESIGN A LABEL *
MH 550 GOSUB680:PRINTCP$"{4 DOWN}CENTER YOUR LABEL
    IN THE BLOCK ABOVE."
QH 560 PRINT"{DOWN}REMEMBER TO LEAVE SPACES BETWEE
    N LINES."
SP 570 PRINT"{DOWN}PRESS {RVS}RETURN{OFF} WHEN DON
    E.{HOME}"
XD 580 IFM=3GOTO600
HG 590 GOSUB630
GM 600 IFP=1THENINPUT"{HOME}";L$:GOTO70
JK 610 IFP=2THENINPUT"{HOME}{9 RIGHT}";L$:GOTO70
QA 620 REM * SUBROUTINE FOR TOP LINES *
CG 630 IFP=1THENX$="{RVS}{RED}{2 SPACES}{OFF}{BLK}
    {36 SPACES}{RVS}{RED}{2 SPACES}{OFF}{BLK}"
CQ 640 IFP=2THENX$="{RVS}{RED}{11 SPACES}{OFF}
    {BLK}{18 SPACES}{RVS}{RED}{11 SPACES}{OFF}
    {BLK}"
EQ 650 PRINT"{HOME}";:FORI=0TO7:PRINTX$;:NEXT:PRIN
    T"{HOME}"
AB 660 RETURN
SK 670 REM * SUBROUTINE FOR BOTTOM LINES *
EC 680 CP$="{HOME}{13 DOWN}":X$="{40 SPACES}"
XM 690 PRINTCP$;:FORI=13TO23:PRINTX$;:NEXT
BQ 700 PRINT"{39 SPACES}{HOME}":RETURN
```


Banners

Doug Smoak

Make your own custom banners with this flexible and easy-to-use program. For use with Commodore and Commodore-emulating printers.

Put your computer and printer to creative use with “Banners,” a short BASIC program. Banners is flexible and, since it’s menu driven, easy to use. The program lets you select the height and width of your banner’s characters, and even allows for customized (redefined) characters, such as foreign language letters, and special math or scientific symbols.

Selecting from the Menu

Type in the appropriate version of Banners and save a copy to tape or disk. Program 1 is for Commodore 1525, MPS 801 or 803, or compatible printers. Program 2 is for the Commodore 1526 or MPS 802.

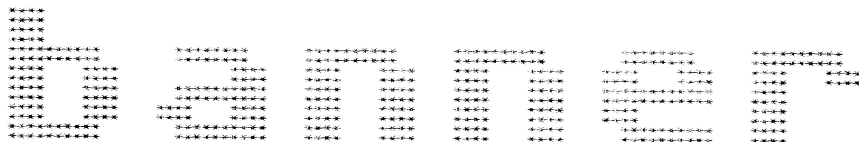
*Note: Spacing is very important in line 340. Make sure that you insert a space between the quotation marks in the statement **PR\$=PR\$+" "**. Without that space, the letters printed out look as if they have collapsed. If the crossbar on a capital H has fallen to the baseline, making it resemble a capital U, the problem is most likely in line 340. If problems persist, carefully check lines 280–290 and make sure you did not substitute a less-than sign for a left parenthesis or a B for an 8.*

Load and run the program once it’s typed in. Be sure your printer is turned on and the device number set to 4.

You’re first shown three choices:

- U (uppercase and graphics)
- L (upper-/lowercase)
- C (custom character set).

After pressing one of these keys, you’re asked to choose which character you wish to use to fill in the large letters for your banner (see the accompanying figure for an example). You’ll find that certain characters make lighter or darker banner letters.



This banner was reduced so it could fit on a book page. It was created with the upper-/lowercase option, an asterisk () character, a height of 2, and width of 2.*

The next prompt asks for your message. Type it in from the point where the cursor appears (don't cursor left before starting). Your message can be as long as two screen lines. After typing the message, press RETURN.

Next, you must indicate how tall the message's letters should be. Press a key in the range 1–9. The smallest size (1) is the height of seven normally printed characters. Choosing 2 yields a letter 14 characters high; 3, a letter 21 characters high; and so on up to 9, which is 63 characters high.

Now enter the width. The maximum width is determined by the number you selected for height. If you chose 9 for height, you'll have a choice of 1–9 for width. If you chose 4 for height, you'll only have a choice of 1–4.

Once you've chosen a width, printing begins. When the banner is complete, you're asked to press X (to exit the program), R (to reprint the message), or N (to enter a new message).

Helpful Hints

Larger letters take more time to print. If you have more than one message to print, you might consider smaller letters. This is important if you're making banners for children or if people are taking turns. Also, consider that the "blocky" look of the letters is more obvious in the larger sizes unless they're to be viewed from a distance.

Using Custom Characters

Banners lets you print out your own custom characters. If you're a seasoned programmer, you've probably worked with custom characters before. If you're new to programming and interested in learning how to create your own custom characters, refer to the *Commodore 64 Programmer's Reference Guide*

(pp. 108–114) or any of the following COMPUTE! books: *Creating Arcade Games on the 64*, *Commodore 64 Graphics*, *COMPUTE!'s First Book of Commodore 64*, or *COMPUTE!'s First Book of Commodore 64 Sound and Graphics*.

If you choose the custom character set option with Banners, the menu will remind you to locate the custom character set beginning at location 12288. Be sure to load the custom character set before loading and running Banners.

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

Program 1. Banners for Commodore 1525, 801, 803

```

JG 100 POKE56,48:POKE55,0:CLR:CU=53272:GOSUB520:SY
S828:POKE53281,15:MT$=""
AX 110 PRINT"{CLR}{BLK}{RVS}{13 SPACES}BANNER PRIN
TER{13 SPACES}{OFF}"
XM 120 CC=0:PRINT"{3 DOWN}PRESS A KEY:":PRINT"
{2 DOWN}{RVS}U{OFF}◀FOR UPPER CASE AND GRAP
HICS"
BP 130 PRINT"{2 DOWN}{RVS}L{OFF}◀FOR UPPER/LOWER C
ASE"
FE 140 PRINT"{2 DOWN}{RVS}C{OFF}◀FOR CUSTOM CHARAC
TER SET (CHARACTER{3 SPACES}DATA MUST BE AT
12288!)"
PC 150 GETCH$:IFCH$=""THEN150
AQ 160 IFCH$="U"THENCH=16384:PRINTCHR$(142):GOTO200
0
DX 170 IFCH$="L"THENCH=16384+2048:PRINTCHR$(14):GO
TO2000
MR 180 IFCH$="C"THENFORK=12544TO12551:POKEK,0:NEXT
:CH=12288:GOSUB510:GOTO2000
DX 190 GOTO150
HH 200 PRINT"{CLR}{RVS}WHAT CHARACTER DO YOU WANT
{SPACE}TO MAKE UP{3 SPACES}THE LARGE LETTER
S"
JP 210 GOSUB610:PP$=MT$
RS 220 MT$="":PRINT"{CLR}{RVS}WHAT IS YOUR MESSAGE
{2 SPACES}":GOSUB610:N$=MT$
XA 230 PRINT"{RVS}HOW TALL (1 TO 9)":INPUTTS:IFTS
<1THENTS=1
DF 240 PRINT"{RVS}HOW WIDE (1 TO"TS"{LEFT})":INPU
TWS:IFWS>TSORWS<1THENWS=TS
DQ 250 OPEN1,4
QR 260 FORL=1TOLEN(N$)
RD 270 A$=MID$(N$,L,1):PRINT"{RVS}{HOME}{9 DOWN}PR
INTING{OFF} "A$:IFA$=" "THENGOSUB480:GOTO38
0

```

```

HK 280 Q=ASC(A$):Q=Q+(Q>63)*64+(Q>191)*64+(Q=255)*
      33
CG 290 I=CH+Q*8
QR 300 FORJ=7TO0STEP-1
MC 310 FORK=7TO0STEP-1
PS 320 B=PEEK(I+K)
CF 330 IFBAND2↑JTHENFORT=1TOTS:PR$=PR$+PP$:NEXTT:G
      OTO350
CA 340 FORT=1TOTS:PR$=PR$+" ":NEXTT
PR 350 NEXTK
PR 360 FORX=1TOWS:PRINT#1,CHR$(15)SPC((72-TS*8)/2)
      PR$CHR$(8):NEXTX:PR$=""
PS 370 NEXTJ
HX 380 NEXTL
MJ 390 PRINT#1,CHR$(8):CLOSE1
XK 400 IF(PEEK(CU)AND12)=12THENPOKECU,(PEEK(CU)AND
      240)OR4:CC=1
SK 410 PRINT"{CLR}PRESS A KEY:":PRINT"{DOWN}X TO E
      XIT PROGRAM":PRINT"{DOWN}R TO REPRINT MESSA
      GE"
CE 420 PRINT"{DOWN}N TO ENTER NEW MESSAGE"
RP 430 GETY$:IFY$=""THEN430
KD 440 IFY$="R"THENGOSUB490:PRINT"{CLR}"N$:GOTO250
AA 450 IFY$="X"THENOPEN1,4:PRINT#1,CHR$(15):CLOSE1
      :PRINT"{CLR}":END
ES 460 IFY$="N"THEN110
EB 470 GOTO430
MJ 480 FORX=1TO8*WS:PRINT#1,CHR$(8):NEXT:RETURN
XF 490 IFCC=1THENC=0:GOSUB510
QF 500 RETURN
HQ 510 POKECU,(PEEK(CU)AND240)OR12:RETURN
SA 520 FORI=828TO887:READA:X=X+A:POKEI,A:NEXTI
XG 530 IFX<>8638THENPRINT"ERROR IN DATA STATEMENTS
      .":STOP
QJ 540 RETURN
RM 550 DATA 173,14,220,41,254,141,14,220,169,208
FP 560 DATA 133,252,169,64,133,254,165,1,41,251
AC 570 DATA 133,1,169,0,133,251,133,253,168,162
QX 580 DATA 16,177,251,145,253,200,208,249,230
KD 590 DATA 252,230,254,202,208,242,165,1,9,4
HH 600 DATA 133,1,173,14,220,9,1,141,14,220,96
HJ 610 POKE198,5:POKE631,155:POKE632,34:POKE633,34
      :POKE634,20:POKE635,144
MF 620 INPUTMT$:RETURN

```

Program 2. Banners for Commodore 1526, 802

```

RA 90 PRINT"{CLR}":OPEN6,4,6:PRINT#6,CHR$(27):CLOS
      E6
JG 100 POKE56,48:POKE55,0:CLR:CU=53272:GOSUB520:SY
      S828:POKE53281,15:MT$=""

```

```

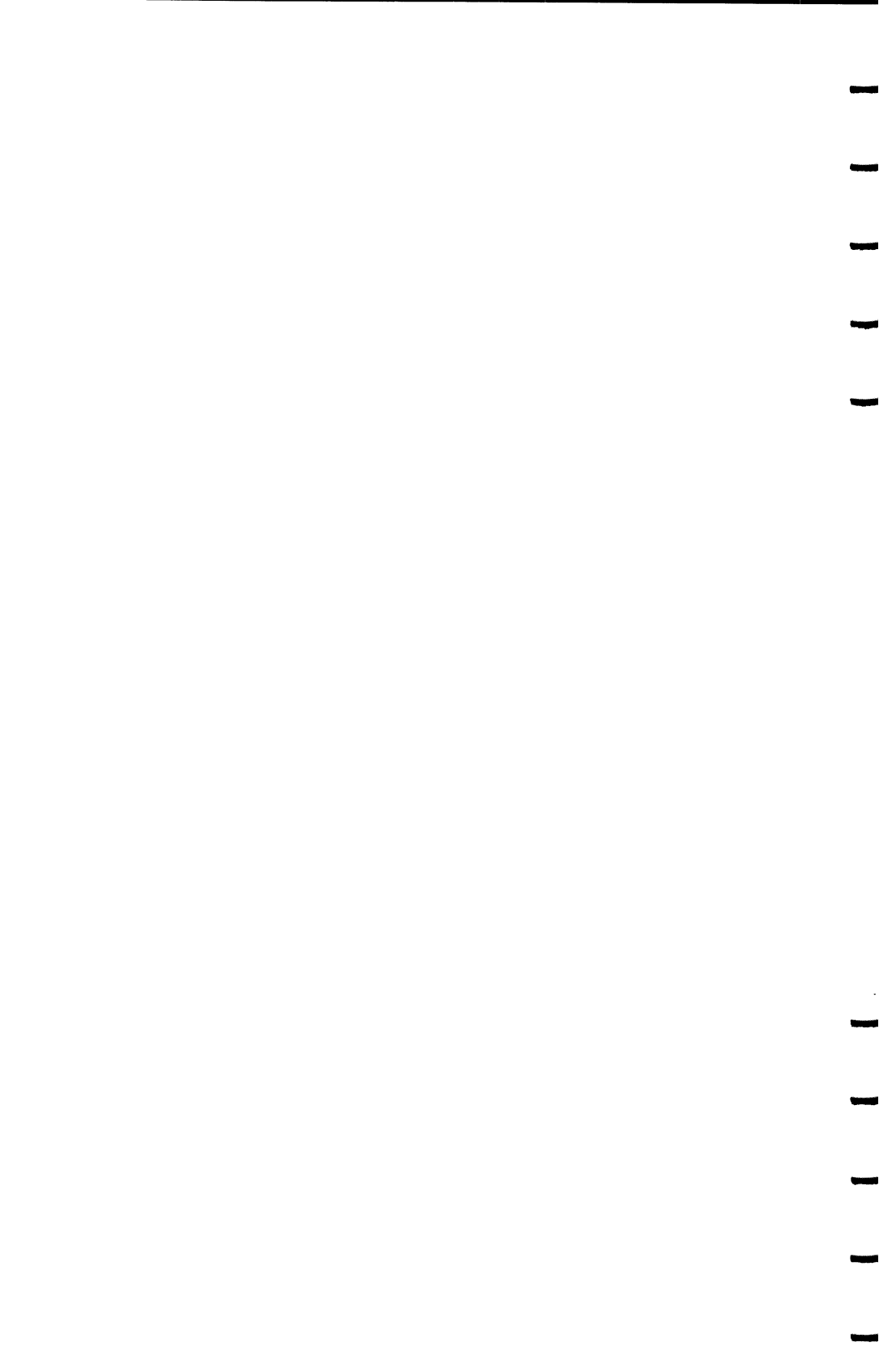
AX 110 PRINT "{CLR}{BLK}{RVS}{13 SPACES}BANNER PRIN
TER{13 SPACES}{OFF}"
XM 120 CC=0:PRINT "{3 DOWN}PRESS A KEY:":PRINT "
{2 DOWN}{RVS}U{OFF}◀FOR UPPER CASE AND GRAP
HICS"
BP 130 PRINT "{2 DOWN}{RVS}L{OFF}◀FOR UPPER/LOWER C
ASE"
FE 140 PRINT "{2 DOWN}{RVS}C{OFF}◀FOR CUSTOM CHARAC
TER SET (CHARACTER{3 SPACES}DATA MUST BE AT
12288!)"
PC 150 GETCH$:IFCH$=""THEN150
AQ 160 IFCH$="U"THENCH=16384:PRINTCHR$(142):GOTO20
0
DX 170 IFCH$="L"THENCH=16384+2048:PRINTCHR$(14):GO
TO200
MR 180 IFCH$="C"THENFORK=12544TO12551:POKEK,0:NEXT
:CH=12288:GOSUB510:GOTO200
DX 190 GOTO150
HH 200 PRINT "{CLR}{RVS}WHAT CHARACTER DO YOU WANT
{SPACE}TO MAKE UP{3 SPACES}THE LARGE LETTER
S"
JP 210 GOSUB610:PP$=MT$
RS 220 MT$="":PRINT "{CLR}{RVS}WHAT IS YOUR MESSAGE
{2 SPACES}":GOSUB610:N$=MT$
XA 230 PRINT "{RVS}HOW TALL (1 TO 9)":INPUTTS:IFTS
<1THENNTS=1
DF 240 PRINT "{RVS}HOW WIDE (1 TO"TS"{LEFT})":INPU
TWS:IFWS>TSORWS<1THENWS=TS
DQ 250 OPEN1,4
QR 260 FORL=1TOLEN(N$)
RD 270 A$=MID$(N$,L,1):PRINT "{RVS}{HOME}{9 DOWN}PR
INTING{OFF} "A$:IFA$=" "THENGOSUB480:GOTO38
0
HK 280 Q=ASC(A$):Q=Q+(Q>63)*64+(Q>191)*64+(Q=255)*
33
CG 290 I=CH+Q*8
QR 300 FORJ=7TO0STEP-1
MC 310 FORK=7TO0STEP-1
PS 320 B=PEEK(I+K)
CF 330 IFBAND2↑JTHENFORT=1TOTOTS:PR$=PR$+PP$:NEXTT:G
OTO350
CA 340 FORT=1TOTOTS:PR$=PR$+" ":NEXTT
PR 350 NEXTK
BH 360 FORX=1TOWS:PRINT#1,CHR$(15)SPC((72-TS*8)/2)
PR$CHR$(32):NEXTX:PR$=""
PS 370 NEXTJ
HX 380 NEXTL
FE 390 PRINT#1,CHR$(32):CLOSE1
XK 400 IF(PEEK(CU)AND12)=12THENPOKECU,(PEEK(CU)AND
240)OR4:CC=1

```

```
SK 410 PRINT "{CLR}PRESS A KEY:":PRINT "{DOWN}X TO E
      XIT PROGRAM":PRINT "{DOWN}R TO REPRINT MESSA
      GE"
CE 420 PRINT "{DOWN}N TO ENTER NEW MESSAGE"
RP 430 GETY$:IFY$="" THEN 430
KD 440 IFY$="R" THEN GOSUB 490:PRINT "{CLR}"N$:GOTO 250
AA 450 IFY$="X" THEN OPEN 1,4:PRINT#1,CHR$(15):CLOSE 1
      :PRINT "{CLR}":END
ES 460 IFY$="N" THEN 110
EB 470 GOTO 430
MJ 480 FOR X=1 TO 8*WS:PRINT#1,CHR$(8):NEXT X:RETURN
XF 490 IF CC=1 THEN CC=0:GOSUB 510
QF 500 RETURN
HQ 510 POKE CU,(PEEK(CU)AND 240)OR 12:RETURN
SA 520 FOR I=828 TO 887:READ A:X=X+A:POKE I,A:NEXT I
XG 530 IF X<>8638 THEN PRINT "ERROR IN DATA STATEMENTS
      .":STOP
QJ 540 RETURN
RM 550 DATA 173,14,220,41,254,141,14,220,169,208
FP 560 DATA 133,252,169,64,133,254,165,1,41,251
AC 570 DATA 133,1,169,0,133,251,133,253,168,162
QX 580 DATA 16,177,251,145,253,200,208,249,230
KD 590 DATA 252,230,254,202,208,242,165,1,9,4
HH 600 DATA 133,1,173,14,220,9,1,141,14,220,96
HJ 610 POKE 198,5:POKE 631,155:POKE 632,34:POKE 633,34
      :POKE 634,20:POKE 635,144
MF 620 INPUT MT$:RETURN
```

Chapter 4

For *SpeedScript*



SpeedCheck

An Expandable Spelling Checker

Ottis R. Cowper

This easy-to-use program can help eliminate misspellings in your writing. It builds a personalized dictionary of the words you use most frequently. It works with all versions of SpeedScript, and with other word processors which store text in a similar format, including WordPro and PaperClip. A disk drive is required.

Many commercial word processing programs include spelling checkers—programs which search for misspellings in documents. Since the initial debut of *SpeedScript* in January 1984, COMPUTE! Publications has received hundreds of requests for a compatible spelling checker. Here it is—"SpeedCheck."

SpeedCheck examines your documents word by word, comparing your typing with entries in its dictionary. What distinguishes this program from other spelling checkers is that it's *interactive*. You see your text on the screen as it's being checked. Words that aren't found in the dictionary are highlighted, so you can correct any misspellings immediately. SpeedCheck also highlights correctly spelled words if they aren't already in its dictionary. These new words can then be added with the touch of a single key.

Commercial spelling checkers come with prepared dictionary disks containing many thousands of words. SpeedCheck can't match that—at least not at first. You start with a blank dictionary disk and add words as you go along. This means that SpeedCheck may be a bit tedious the first few times you use it, since you'll have to add many words. However, you'll soon build a customized dictionary disk that contains all the words you use most often.

The 664 available blocks on a blank dictionary disk provide room to add over 168,000 characters—enough, for example, for over 24,000 seven-letter words.

Save Your Dictionaries

Let's clear up some common misconceptions about spelling checker programs before you discard your dictionaries and grammar books. Like most such programs, SpeedCheck can be deceived. If you type *form* when you mean *from*, SpeedCheck cannot detect the mistake if both words are in your dictionary. SpeedCheck makes no judgment about context; as far as it's concerned, *form* is a correctly spelled word. There's a way around this problem, though. If you frequently misspell *from* as *form*, just don't add *form* to the dictionary disk. Each instance of *form* will be flagged, because SpeedCheck won't find it on the disk, and you can check your use of the word *form*.

Many word processor users expect spelling checkers to magically improve the quality of their writing. However, programs that check grammar and syntax are enormously more complex than those that simply check spelling. A program that checked for good writing would approach artificial intelligence. While SpeedCheck can help you eliminate misspelled words, it can't make you a better writer.

Also, remember that SpeedCheck is *not* a word processor. Although the program allows you to correct spellings and make minor changes, it's not suitable for major editing tasks. That's what your word processor is for. SpeedCheck works especially well with *SpeedScript 3.0*, the version published in the March 1985 issue of *COMPUTE!* magazine (and with updated versions 3.1 and 3.2).

If you have *SpeedScript 3.0* or higher, you can exit the word processor and load SpeedCheck without losing the text in memory. You can also exit SpeedCheck and reload *SpeedScript 3.0* or higher without losing text. Most other compatible word processors—including *SpeedScript* versions 1.0 (*COMPUTE!'s Gazette*, January 1984) and 2.0 (*Gazette Disk*, May 1984)—clear the text area when run, so you must load the document to be checked into SpeedCheck, then save a corrected version after checking.

With minor modifications (explained below), SpeedCheck can also check documents from any word processor that stores characters in the same format as *SpeedScript*: as screen codes in program files. This includes the popular *WordPro 3 Plus/64* and *PaperClip* programs, but excludes those word processors that store text in sequential files, such as *Easy Script*.

For World-Class Misspellers

A price had to be paid to keep SpeedCheck short (it's only 2.6K) and simple, and to minimize the speed constraints of the 1541 disk drive. For one thing, SpeedCheck checks the spelling of only those words which have five or more characters, but this can easily be changed to any value you desire (more on this below).

Five characters is an arbitrary limit, but it yields a good speed-versus-performance ratio. You can drop the limit to as few as two characters and check nearly every word in the document, but then it could take hours to check a long file. Unless you're a world-class misspeller, you'll always spell words like *a*, *an*, and *the* correctly; for most people, it's words like *triumvirate* that cause problems.

When counting characters in words, SpeedCheck recognizes only the letters A–Z and the apostrophe ('). Thus, contractions like *can't* are counted as five-letter words. The hyphen is not recognized, so a word like *left-handed* would be counted as two words, one of four letters and one of six. A letter's case (capital or lowercase) is not significant; all letters are converted to lowercase when looked up or added to the dictionary. Thus, a single dictionary entry would match *basic*, *BASIC*, and *Basic*.

Words in the SpeedCheck dictionary disk are kept in sequential files, with one file for each of the 26 possible initial letters. When a new word is added, it is tacked on to the end of the appropriate file. This makes adding words fast and simple, but it makes looking up words slower because the words are in random—rather than alphabetical—order. Within the file for Z, for example, *zebra* might be found between *zymurgy* and *zipper*. The only way to find a particular item in randomly ordered data is to search sequentially from the first item until the desired item is found. Thus, when SpeedCheck looks up a word, it must hunt through all the words with the same initial letter before it can determine that the word is not present.

Nevertheless, SpeedCheck is faster than you might expect. The text for this article was checked in about 30 minutes using a dictionary disk containing about 2,000 words, including most of the words in the article.

Preparing the Program

Program 1 is a short BASIC program designed to manage the dictionary disk. Program 2 is SpeedCheck itself. Start by typing in and saving a copy of Program 1. This program allows you to create dictionary disks and to review and edit entries in the dictionary. If you've decided to use a value other than 5 for the minimum number of characters for checking, you'll need to change the value of LN in line 110. The values you set here and in Program 2 should agree (see "Customizing SpeedCheck" below for details).

Note: Make sure that there is a space between ST and OR in line 350 of Program 1. If the space is left out, the computer sees the keyword TO in the line.

Your next step is to create a blank dictionary disk.

Program 1 creates a blank disk by formatting it. If you type in the program, save it to disk, and then run the program on the same disk, that disk will be formatted and your copy of the program lost. To make sure this doesn't happen, load Program 1, take the disk containing Program 1 out of the drive, insert a blank disk into the drive, and type RUN.

After running Program 1, select menu option 5. You're prompted for an ID for the disk. Enter any two characters, but be sure that each dictionary disk has a unique ID. Don't change the disk name in line 220; SpeedCheck tests to see that the dictionary disk has this name, and won't let you proceed otherwise.

It's a good idea to occasionally make backup copies of your dictionary disk. Because it takes hours to build up a large dictionary, and because the dictionary disk is subjected to an above-average amount of reading and writing, you need to protect yourself against potential data loss. Any backup program can be used—for example, "ReDisk" (August 1985 *Gazette*). Just be sure to give all backup disks the name DICTIONARY DISK.

After you've created a blank dictionary disk, it would be worthwhile to spend a few minutes entering a basic list of commonly used words. Use option 2 of Program 1. Any words that frequently give you problems should be entered at this time. You'll need to have a real dictionary (the book kind) handy to insure that the words you enter are spelled correctly. You can use options 1 or 4 of Program 1 to review your dictionary, and make any necessary corrections with option 3.

Now you're ready to use SpeedCheck. Since it's written entirely in machine language, the listing for SpeedCheck is in MLX format. Be sure you read and understand the instructions for using the MLX program, found in Appendix D, before proceeding. Load and run MLX, and answer the prompts as follows:

Starting Address: 0801

Ending Address: 1270

Now enter the data from Program 2 and save it on disk.

Although it's written in machine language, the finished SpeedCheck program can be loaded, run, and saved as if it were a BASIC program. If you try to list the program, you'll see a single line of BASIC. To begin, just load SpeedCheck (Program 2) and type **RUN**. If you're a *SpeedScript* user, the initial screen should be familiar. You'll see a blank screen with a blinking cursor and a reverse-video command line at the top.

At this point, you'll have either one or two options. If the cursor is blinking in the same color as the command line, you must load text to begin, so press f7 to access the Load feature. No other key is active. If you had text from *SpeedScript* 3.0 (or higher) in memory when you loaded SpeedCheck—or if you previously exited SpeedCheck with text in memory—then the cursor will be in the bright highlight color. In this case, you can either press RETURN to check the text already in memory, or you can press f7 to load new text.

If any errors occur during the load, an error message will be displayed on the command line. To erase the error message and try another load, press any key. (In general, whenever SpeedCheck displays any message in the command line, it waits for a keypress before proceeding.) After the text is loaded, or after you've pressed RETURN if there's already text in memory, you're prompted to remove your text disk and insert the dictionary disk.

After inserting the dictionary disk, press RETURN and SpeedCheck will begin to fill the screen with text. You'll probably see a number of highlighted words—words in reverse video and a different color from the text (not to be confused with *SpeedScript* formatting commands and printkeys, which also appear in reverse video, but in the same color as the text). These are the words that SpeedCheck did not find in its dictionary, so they may be misspelled. It's also possible that they are correctly spelled, but not yet in your dictionary.

Moving Around the Text Area

After the screen is filled, a cursor appears in the upper left corner of the text, just below the command line. This is the home position for SpeedCheck. At this point, several more keys become active:

- **RETURN:** copies the word the cursor is on into the command line for editing or addition to the dictionary. (We'll call this "grabbing" the word. It will be discussed in more detail later.)
- **Cursor left/right:** behaves like the f1 and f2 keys in *SpeedScript*, moving the cursor left or right by whole words. When you're checking text, you can't move a single character at a time. The cursor should always stop on the first character of a word. It can be moved right one word at a time to the end of the document, with new lines scrolled in at the bottom as necessary. However, it can be moved left only as far as the home position of the current screen. You can't scroll backward in SpeedCheck—only forward.
- **Cursor down/up:** moves the cursor either down to the next line or up to the previous line. When moving up or down, the cursor shifts left or right to land on the first character of a word. The cursor can be moved down to the end of the document, with new lines scrolled in as necessary. However, it can be moved upward only as far as the home position of the current screen.
- **CLR/HOME:** moves the cursor to the home position. If the cursor is already home, pressing this key returns to the start of your document and rechecks the first screen of text.
- **SHIFT-CLR/HOME:** exits SpeedCheck. You'll be asked to verify this choice. Pressing Y exits to BASIC; any other key returns you to text. After exiting, you can type RUN again and reenter SpeedCheck without losing your text. The text will also be retained if you load and run *SpeedScript* 3.0 or higher.
- **↑ (up arrow):** displays the next full screen of text. It's a bit faster than using cursor-right or cursor-down to scroll in one line at a time.
- **SHIFT-↑:** adds all highlighted words on the current screen to the dictionary before displaying the next full screen of text. This saves time when many words must be added to the dictionary, as when you're starting out with a new dictionary disk. You should be sure that all highlighted words on the

screen are spelled correctly before using this feature. **SHIFT-↑** has the same effect as **↑** if no highlighted words are on the screen.

- **f1**: searches forward in text for the next occurrence of a highlighted word and grabs the word into the command line. New screen lines are scrolled in if needed. The search can be cancelled by holding down the RETURN key until the cursor reappears. The search stops at the end of text if no highlighted word is found.
- **f2 (SHIFT-f1)**: searches backwards in text for the last occurrence of a highlighted word and grabs the word into the command line. The search stops at the home position of the current screen if no highlighted word is found.
- **f7**: loads a new document from disk. Pressing RETURN alone at the filename prompt cancels the load and returns you to text. SpeedCheck has no merge feature. A load will overlay any text currently in memory. Load automatically appends the drive designation prefix **0:**, so you should *not* include this when entering your filename. The check character (**√**) in the command line will disappear to indicate disk access. After the load is complete, you'll be prompted to insert the dictionary disk and press RETURN, after which the first screen of the new file will be checked. The cursor will return at the home position.
- **f8 (SHIFT-f7)**: saves the text currently in memory to disk. Before the save, you are prompted to remove the dictionary disk and press RETURN. Pressing any other key cancels the save and returns you to text. (As with Load, Save can also be cancelled by pressing RETURN alone at the filename prompt.) The check character in the command line will disappear during the save to indicate disk access. After the save, the cursor will be returned to its previous position in text. Save automatically appends the drive designation prefix **0:** to the filename you enter, so you should *not* include this in the filename. This also precludes the use of the **@** prefix for Save-with-Replace.

Command Line Editing

As mentioned earlier, SpeedCheck is not a word processor. You can move the cursor to a word in text, but you cannot directly edit it. To modify the word under the cursor, you must grab it into the command line by pressing RETURN (or

searching with f1 or f2). Once the word and the cursor have moved to the command line, the following keys are active.

- All alphabetic, numeric, and punctuation keys.
- **Cursor right/left:** moves the cursor one character to the right or left, but not beyond the leftmost character of the word or more than one space beyond the rightmost.
- **INST/DEL:** allows you to insert spaces or delete characters within the word. The total length of the word in the command line cannot exceed 32 characters.
- **f3:** adds the current contents of the command line to the dictionary. The word in the command line must be at least as long as the specified minimum word length or an error message will result. The word "added" will appear on the command line to let you know that the addition was successful. No check is made to determine if the word already exists in the dictionary before it is added. It doesn't hurt anything to have a word in the dictionary more than once; it just takes up extra space. However, it *is* important that you insure that words added to the dictionary are spelled correctly. You defeat the purpose of the program if you add misspelled words. You can use the editing feature (option 3) of Program 1 to remove duplicated words or correct misspelled words.
- **f4 (SHIFT-f3):** searches the dictionary for the current contents of the command line. The check character (✓) in the command disappears when you press f4. If the word is found, the check mark reappears in the normal command-line color. However, if the word is not found, the check comes back in the bright highlight color (it reverts to normal as soon as a key is pressed). This feature is especially useful after editing. For example, suppose the word *paralell* appeared highlighted in your text, so you grabbed it into the command line and corrected it to read *parallel*. You could then use f4 to determine whether *parallel* is already in the dictionary or needs to be added.
- **f5:** incorporates the current contents of the command line into text in place of the grabbed word. Any highlighting of the original word is removed. If the word hasn't been edited, no obvious change takes place other than the removal of highlighting. If the word returned is shorter, then all characters to the right on that screen line are pulled in to cover the extra space. If you lengthened the word, there will be a slight delay as the screen is reformatted to accommodate the extra

- characters. (The RETURN key also has the same effect as f5.)
- **f6 (SHIFT-f5):** clears the command line and returns the cursor to the text area without incorporating any changes and without removing any highlighting that may be present. It has the effect of "ungrabbing" the word.

You might start a typical SpeedCheck session by loading text, inserting the dictionary disk, and pressing RETURN. When the cursor appears, you press f1 to move to the first highlighted word. You find it spelled correctly, so you press f3 to add it to the dictionary, then f5 to return to text, then f1 to find the next highlighted word. You notice that this word is misspelled, so you correct it, press f4 and discover that the corrected version is already in the dictionary, then press f5 to incorporate the correction into the text.

Then you notice a place where you used *their* when you meant *there*. SpeedCheck didn't catch the mistake, since both words are in its dictionary, but you can correct it simply enough. Use the cursor keys to move the cursor to the word, press RETURN to grab it into the buffer, change the spelling, and then press RETURN again to incorporate the change. As you become comfortable with SpeedCheck's commands, you'll find that you can move quickly and smoothly from key to key as your document is checked.

Customizing SpeedCheck

It's easy to change the value for the minimum number of characters a word must have to be added or looked up. Just load SpeedCheck and POKE the desired value into location 2083. **PRINT PEEK(2083)** will show that the current value is 5. To change it, say, to a value of 4, type **POKE 2083,4**. If you save a copy of SpeedCheck after changing the value in 2083, your new value will be incorporated into SpeedCheck. Do not drop the limit below two characters.

You can use one minimum-length value for a while and then change to another without having to create a new dictionary disk. But consider that if you use a four-character limit extensively and then switch to a five-character limit, you'll have many four-character words in the dictionary that the new version will never use. Likewise, if you use a five-character limit extensively, then switch to a four-character limit, searching will be slowed because all four-character words will start far into the dictionary.

If you'd prefer other colors for text, background, or highlighting, they're easily changed. The following locations hold color values:

2079 Background color
2080 Text color
2081 Highlight color

Simply POKE the appropriate location with the standard Commodore code (0–15) for the desired color. The only restriction is that the color you choose for highlighting (location 2081) must be different from the text color (2080). If you're not satisfied with your new choices, exit SpeedCheck and try again. If you save a new copy of SpeedCheck, your color changes will be incorporated.

*Note: If you're using the original version of SpeedScript (version 1.0, published in the January 1984 issue of COMPUTE!'s GAZETTE), make a minor adjustment to SpeedCheck. After loading the program, type **POKE 4242,24**. Then save the modified copy of SpeedCheck to disk.*

SpeedScript doesn't waste any text memory. The spaces you see at the ends of lines and following return marks are not stored in memory. This is not the case with all word processors. Some, including *WordPro* and *PaperClip*, include within text all the spaces shown at the ends of lines. To use SpeedCheck with these word processors, you must defeat SpeedCheck's handling of the *SpeedScript* return mark (↵) character. To create a version for *WordPro* and similar processors, substitute the following line while entering Program 2 with MLX:

```
0E69:28 B0 05 8A EA EA 90 C5 F8
```

When you use this version, text will look a bit different from what you may be accustomed to—there will be many extra spaces and words are no longer split at the right screen margin. Nevertheless, SpeedCheck should perform normally, and you can use the same dictionary disks you use with the *SpeedScript* version.

One additional change is required to create a version for *PaperClip* files. Since that program stores text slightly differently from the other two word processors, you must modify the Load routine a bit. Substitute the following lines when entering Program 2 with MLX:

```

1029:03 A6 AE A4 AF 20 D8 FF 44
1079:FF A2 01 20 C6 FF A9 C2 B0
1081:85 AE A9 C4 85 03 A9 23 40
1089:85 AF 85 04 A9 00 85 B0 52
1091:A9 24 85 B1 A0 00 A5 90 3C

```

If you're using a 1541-compatible dual-drive unit (drive numbers 0 and 1) with your 64, it's simple to modify SpeedCheck to use both drives. Load SpeedCheck, but don't run it. Enter the following line in immediate mode:

POKE 4700,49:POKE 4707,49

Then save a copy of the modified program. This new version requires that the text disk be in drive 0 and the dictionary disk in drive 1. When using the Save command, you'll still get the prompt to remove the dictionary disk. Simply ignore this prompt and press RETURN.

Program 1. Dictionary Manager

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

```

XF 100 POKE55,0:POKE56,25:CLR:SA=PEEK(56)*256
MG 110 LN=5:D$="{5 DOWN}":S$="{8 SPACES}"
DM 120 PRINT CHR$(142)" {CLR}"D$TAB(8)" {RVS} DICTIO
    NARY MANAGER "
FC 130 PRINT "{2 DOWN} PRESS: [1] TO REVIEW ENTRIE
    S{DOWN}":PRINT S$"[2] TO ADD ENTRIES{DOWN}"
HC 140 PRINT S$"[3] TO EDIT ENTRIES{DOWN}":PRINT S
    $"[4] TO PRINT ENTRIES{DOWN}"
QJ 150 PRINT S$"[5] TO PREPARE A NEW DISK{DOWN}":P
    RINT S$"[6] TO EXIT THE MANAGER{3 DOWN}"
GE 160 GET K$:IF K$<"1" OR K$>"6" THEN160
CD 170 PRINT"{CLR}{4 DOWN}":ON VAL(K$) GOSUB370,45
    0,670,550,190,180:GOTO120
ED 180 PRINT"{CLR}":END
BM 190 PRINT S$"{RVS} CREATE NEW DICTIONARY DISK "
    :PRINT"{2 DOWN}ARE YOU SURE";
GC 200 INPUT" [Y/N]";K$:IF LEFT$(K$,1)<>"Y" THEN P
    RINT"{CLR}":RETURN
QR 210 INPUT"{2 DOWN}UNIQUE ID FOR THIS DISK";ID$:
    PRINT"{2 DOWN}FORMATTING....PLEASE WAIT"
PJ 220 OPEN 15,8,15,"N0:DICTIONARY DISK,"+ID$:GOSU
    B910:IF ER THEN RETURN
FE 230 PRINT"{DOWN}FORMATTING COMPLETE...CREATING
    {SPACE}HEADINGS:"
EB 240 FOR I=65 TO 90:OPEN 1,8,8,"0:"+CHR$(I)+" ,S,
    W":GOSUB910:IF ER THEN270

```

```

AK 250 CLOSE 1:PRINT CHR$(I);:NEXT:PRINT
QX 260 PRINT "{DOWN}** DICTIONARY DISK CREATED **
      {2 DOWN}":GOTO880
CG 270 PRINT"CREATING FILE FOR ";CHR$(I):PRINT"
      {DOWN}TRY AGAIN";:GOTO200
EK 280 PRINT S$"{DOWN}(ENTER {RVS}@{OFF} FOR MENU)
      {DOWN}"
HX 290 INPUT "{DOWN}STARTING WITH";L$:L=ASC(L$+CHR$
      (0)):IF L=64 OR L=42 THEN RETURN
XP 300 L$=CHR$(L):IF L<65 OR L>90 THEN PRINT S$L$
      IS NOT A VALID LETTER":GOTO290
JG 310 OPEN15,8,15:OPEN1,8,8,"0:"+L$+K$:GOSUB910:I
      F ER=0 THEN RETURN
BK 320 PRINT "{DOWN}TRYING TO ACCESS FILE FOR ";L$:
      GOTO290
DE 330 F=0:GET#1,K$:K=ASC(K$+CHR$(0)):IF K>127 THE
      N K=K AND 127:F=1
DF 340 IF K<>39 THEN K=K OR 64
RS 350 W$=W$+CHR$(K):IF ST OR F THEN RETURN
QM 360 GOTO330
KK 370 PRINT S$"{RVS} REVIEW ENTRIES ":K$=",S,R":G
      OSUB280:IF L=64 OR L=42 THEN RETURN
AQ 380 GET#1,K$:W$=L$:GOSUB330:IF MID$(W$,2,1)<>"@
      " THEN400
MX 390 PRINT "{DOWN}** NO ENTRIES FOR ";L$;" ***":GO
      TO880
SK 400 PRINT CHR$(14)" {CLR} {2 DOWN}ENTRIES FOR "L$
      :PRINT "{DOWN}(HOLD DOWN {RVS} SPACE {OFF} T
      O ADVANCE){DOWN}"
HC 410 PRINT S$;W$:IF ST THEN430
PB 420 WAIT 203,64,64:W$=L$:GOSUB330:GOTO410
DD 430 IF ST<>64 THEN910
DM 440 WAIT 203,64:POKE 198,0:GOTO880
GF 450 PRINT S$"{RVS} ADD ENTRIES ":K$=",A":GOSUB2
      80:IF L=64 OR L=42 THEN RETURN
JA 460 PRINT CHR$(14)" {CLR} {DOWN}ENTER WORDS START
      ING WITH {RVS}";L$
MJ 465 PRINT "{DOWN}(TYPE {RVS}@{OFF} TO QUIT)
      {DOWN}"
PH 470 W$="":INPUT W$:F$=LEFT$(W$,1):IF F$="@"THEN
      CLOSE1:CLOSE15:PRINT "{CLR}":RETURN
DA 480 IF F$<>L$ AND F$<>CHR$(L+128)THENPRINTW$ " D
      OESN'T START WITH "L$"{DOWN}":GOTO470
HR 490 N=LEN(W$):IF N<LN THEN PRINT W$ " HAS FEWER
      {SPACE}THAN"LN"LETTERS{DOWN}":GOTO470
EH 500 FOR I=2 TO N:K=ASC(MID$(W$,I)):IF K=39 THEN
      520
FR 510 IF K<65 OR(K>90 AND K<193)OR K>218 THEN K=0
HP 520 K=K AND 63:IF I=N THEN K=K OR 128
GP 530 IF K THEN PRINT#1,CHR$(K);:IF ST THEN910

```

```

CA 540 NEXT:PRINT S$W$" ADDED.":GOTO470
CQ 550 PRINT S$"{RVS}{2 SPACES}PRINT ENTRIES
      {2 SPACES}":PRINT"{DOWN}{2 SPACES}(ENTER
      {RVS}*{OFF} TO PRINT ALL ENTRIES)"
RP 560 K$=",S,R":GOSUB280:IF L=64 THEN RETURN
XM 570 OPEN 4,4,7:IF L=42 THEN FOR I=65 TO 90:L$=C
      HR$(I):K$=",S,R":GOSUB310
DM 580 GET#1,K$:W$=L$:GOSUB330:K=ST:IF MID$(W$,2,1)
      )<>"@" THEN600
EB 590 PRINT#4,"** NO ENTRIES FOR "L$" **":GOTO650
QF 600 PRINT#4,"ENTRIES BEGINNING WITH "L$":":PRIN
      T#4
GE 610 FOR J=0 TO 3:PRINT#4,W$SPC(20-LEN(W$)):IF
      {SPACE}K THEN J=4:GOTO630
XF 620 W$=L$:GOSUB330:K=ST
FA 630 NEXT J:PRINT#4:IF K=0 THEN610
PJ 640 IF K<>64 THEN CLOSE 4:GOTO910
FS 650 CLOSE 1:CLOSE 15:PRINT#4:PRINT#4:IF L=42 TH
      EN NEXT I
RC 660 CLOSE 4:RETURN
BB 670 PRINT S$"{RVS} EDIT ENTRIES ":K$=",S,R":GOS
      UB280:IF L=64 OR L=42 THEN RETURN
CA 680 PRINT CHR$(14)"{CLR}{2 DOWN}EDIT WORDS STAR
      TING WITH "L$":
XC 690 PRINT"{DOWN}(ENTER {RVS}@{OFF} TO CANCEL ED
      ITING){2 DOWN}":GET#1,K$:AD=SA
PJ 700 W$=L$:GOSUB330:IF MID$(W$,2,1)="@" THEN PRI
      NT"** NO ENTRIES **":GOTO880
JA 710 PRINT"{2 SPACES}"W$:K$="":INPUT"{UP}";K$:N=
      LEN(K$):IF N=0 THEN800
AM 720 F$=LEFT$(K$,1):IF F$="@" THEN PRINT"{DOWN}E
      DITING CANCELED":GOTO880
AH 730 IF F$<>L$ AND F$<>CHR$(L+128)THEN PRINT K$"
      DOESN'T START WITH "L$:GOTO710
KX 740 IF N<LN THEN PRINT K$"HAS FEWER THAN"LN"LET
      TERS":GOTO710
HE 750 FOR I=2 TO N:K=ASC(MID$(K$,I)):IF K=39 THEN
      780
FS 760 IF (K>64 AND K<91)OR(K>192 AND K<219) THEN78
      0
JG 770 PRINT K$" HAS INVALID CHARACTER":AD=AD-I-2:
      GOTO710
AM 780 K=K AND 63:IF I=N THEN K=K OR 128
SJ 790 POKE AD,K:AD=AD+1:NEXT:IF AD>40959 THEN AD=
      AD-N:PRINT"{RVS}BUFFER FULL":GOTO820
BC 800 IF ST=0 THEN700
GP 810 IF ST<>64 THEN910
FH 820 PRINT"{2 DOWN}** WRITING NEW FILE FOR "L$"
      {SPACE}**"

```

```

QE 830 CLOSE 1:PRINT#15,"S0:"+L$:GOSUB910:IF ER<2
    {SPACE}THEN860
HF 840 INPUT"TRY AGAIN [Y/N]";K$:IF LEFT$(K$,1)<>"
    Y" THEN RETURN
DE 850 OPEN 15,8,15:GOTO820
CC 860 OPEN 1,8,8,"0:"+L$+"",S,W":PRINT#1:IF ST THE
    N GOSUB910:GOTO840
BQ 870 FOR I=SA TO AD-1:PRINT#1,CHR$(PEEK(I));:NEX
    T:IF ST THEN GOSUB910:GOTO840
PD 880 CLOSE 1:CLOSE 15:PRINT"{DOWN}{RVS} PRESS AN
    Y KEY FOR MENU "
RS 890 GET K$:IF K$="" THEN890
BA 900 PRINT"{CLR}":RETURN
AA 910 INPUT#15,ER,E$:IF ER>1 THEN PRINT"{DOWN}
    {RVS} DISK ERROR:";ER;E$:CLOSE 15:CLOSE 1
XS 920 RETURN

```

Program 2. SpeedCheck

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```

0801:1D 08 0A 00 99 22 93 0E 6D
0809:08 22 3A 9E 32 30 38 32 CB
0811:3A 53 50 45 45 44 43 48 7B
0819:45 43 4B 00 00 00 0F 0C 30
0821:01 A9 05 09 B0 8D 14 12 43
0829:AD 1F 08 8D 21 D0 8D 20 39
0831:D0 A5 01 29 FE 85 01 A9 7F
0839:00 85 FE 20 90 FF A9 CB 30
0841:CD 6E 23 8D 6E 23 D0 1B CD
0849:85 FE AD 08 20 85 B0 AD 38
0851:09 20 85 B1 18 AD 17 20 7F
0859:69 01 85 AE AD 18 20 69 71
0861:00 85 AF 20 75 0B A2 00 E7
0869:A0 00 20 9A 0B 20 20 0B 9B
0871:AD 20 08 A6 FE F0 03 AD 3B
0879:21 08 8D 28 D8 20 A2 08 E4
0881:A5 FE F0 09 E0 0D D0 05 B4
0889:20 BD 10 D0 0B E0 88 D0 E5
0891:EC 20 43 10 A5 F8 F0 CB 48
0899:20 A2 08 20 C8 08 4C 99 FD
08A1:08 A4 A6 B1 A8 48 B1 A8 41
08A9:49 80 91 A8 A9 E2 85 FF 1F
08B1:20 E4 FF A4 A6 AA D0 09 DF
08B9:CA D0 FD C6 FF D0 F1 F0 A7
08C1:E5 68 91 A8 60 A2 0D 20 63
08C9:66 09 20 90 09 8A B0 39 69
08D1:C9 0D D0 03 4C EC 0C C9 4C
08D9:11 D0 03 4C DB 09 C9 1D 7F
08E1:D0 03 4C B4 09 C9 5E D0 EC

```

08E9:03 4C E8 0A C9 85 D0 19 6B
08F1:20 B4 09 B0 6F 20 E4 FF 31
08F9:C9 0D F0 68 A4 A6 B1 AA A4
0901:29 0F CD 21 08 D0 E9 F0 7F
0909:BC C9 13 D0 10 A5 A7 C9 8B
0911:01 D0 07 A5 A6 D0 03 4C DD
0919:FE 0A 4C 20 0B C9 88 D0 1A
0921:03 4C 43 10 C9 89 D0 19 60
0929:20 24 0A A4 A6 B1 AA 29 5A
0931:0F CD 21 08 F0 8F A5 A7 9B
0939:C9 02 B0 EC C0 00 D0 E8 26
0941:60 C9 8C D0 03 4C EF 0F CC
0949:C9 91 D0 03 4C 06 0A C9 47
0951:93 D0 03 4C 46 0B C9 9D 16
0959:D0 03 4C 24 0A C9 DE D0 66
0961:03 4C A6 0A 60 A0 00 84 87
0969:FD 84 FE B1 FB F0 1E C9 40
0971:27 D0 04 C0 00 D0 0C C9 FC
0979:1B 90 08 C9 41 90 0E C9 0D
0981:5B B0 0A 99 6F 12 C8 84 22
0989:FD C0 20 90 DE 18 60 A5 95
0991:FC C5 AF 90 04 A5 FB C5 07
0999:AE 60 20 90 09 B0 13 A5 FE
09A1:FD D0 02 E6 FD 18 A5 FB 2D
09A9:65 FD 85 FB A5 FC 69 00 52
09B1:85 FC 60 20 9B 09 B0 21 57
09B9:A5 A6 65 FD 85 A6 C9 28 57
09C1:B0 0A A8 B1 AA 29 0F CD C4
09C9:1F 08 D0 06 A9 00 85 A6 E6
09D1:F0 08 20 66 09 A5 FD F0 94
09D9:DA 60 A6 A7 E8 8A 20 38 AA
09E1:0B 20 90 09 B0 F3 86 A7 2E
09E9:E0 19 90 06 20 D1 0E 4C D5
09F1:66 09 A5 A8 69 28 85 A8 58
09F9:85 AA A5 A9 69 00 20 44 98
0A01:0A D0 36 F0 D5 A5 A7 C9 82
0A09:02 B0 03 4C 20 0B C6 A7 D1
0A11:A5 A8 E9 28 85 A8 85 AA 66
0A19:A5 A9 E9 00 20 44 0A D0 9E
0A21:18 F0 E2 A4 A6 D0 07 A5 50
0A29:A7 C9 02 B0 06 60 20 75 36
0A31:0A D0 06 A9 27 85 A6 D0 47
0A39:CC 84 A6 84 FD 20 36 0B D9
0A41:4C A6 09 85 A9 29 03 09 9F
0A49:D8 85 AB A4 A6 B1 AA 29 65
0A51:0F CD 1F 08 D0 03 88 10 78
0A59:F4 B1 A8 C9 20 D0 20 C0 4B
0A61:27 B0 11 B1 AA 29 0F CD 58
0A69:1F 08 F0 08 C8 B1 A8 C9 D5
0A71:20 F0 EC 60 88 B1 A8 C9 9B

0A79:20 D0 04 C0 00 D0 F5 C0 4E
0A81:00 F0 08 88 B1 A8 C9 20 3F
0A89:D0 F5 C8 B1 A8 C9 20 60 C4
0A91:20 75 0B A2 00 BD 6F 12 86
0A99:09 80 99 00 04 C8 E8 E4 7F
0AA1:F8 90 F2 C8 60 20 20 0B 10
0AA9:A0 00 B1 AA 29 0F CD 1F 2F
0AB1:08 F0 20 CD 21 08 D0 16 C7
0AB9:20 3A 0A A9 A0 8D 00 04 87
0AC1:20 66 09 84 F8 20 9F 0F 7F
0AC9:18 A5 A6 65 FD A8 C8 C0 63
0AD1:28 90 D7 E6 A7 A5 A7 C9 74
0AD9:19 B0 0C A5 AA 69 28 85 53
0AE1:AA 90 C5 E6 AB D0 C1 A9 64
0AE9:19 20 38 0B 20 90 09 B0 50
0AF1:28 A5 FB 8D 42 03 A5 FC 42
0AF9:8D 43 03 D0 0A A5 B0 8D E8
0B01:42 03 A5 B1 8D 43 03 A9 F1
0B09:93 20 D2 FF A2 55 20 97 8D
0B11:0B 20 20 0B 20 3B 0E 90 04
0B19:FB 20 20 0B 4C 75 0B A9 E1
0B21:28 85 A8 85 AA A9 04 85 A3
0B29:A9 A9 D8 85 AB A9 00 85 7B
0B31:A6 A9 01 85 A7 A5 A7 0A AA
0B39:A8 B9 40 03 85 FB C8 B9 B1
0B41:40 03 85 FC 60 A2 CC 20 FF
0B49:97 0B 20 AA 0B C9 59 D0 9F
0B51:23 A5 B0 8D 08 20 A5 B1 0F
0B59:8D 09 20 38 A5 AE E9 01 BC
0B61:8D 17 20 A5 AF E9 00 8D 15
0B69:18 20 A9 0F 20 C3 FF 68 32
0B71:68 6C 16 03 A0 27 A9 A0 5F
0B79:99 00 04 AD 20 08 99 00 0C
0B81:D8 88 10 F2 A9 FA 8D 00 AB
0B89:04 A9 BA 8D 01 04 A0 03 98
0B91:60 20 97 0F A2 65 20 75 24
0B99:0B BD 67 11 D0 01 60 99 87
0BA1:00 04 E8 C8 D0 F3 20 9A 93
0BA9:0B 20 E4 FF C9 00 F0 F9 14
0BB1:60 A2 10 A9 00 85 F8 85 CA
0BB9:FE F0 07 20 91 0A A2 20 88
0BC1:A0 03 86 F7 84 F9 A9 04 9C
0BC9:85 FA A0 00 B1 F9 48 29 A4
0BD1:7F 91 F9 84 A5 20 AA 0B A1
0BD9:85 FF A5 FE 10 05 20 91 BD
0BE1:0A 84 FE A4 A5 68 91 F9 34
0BE9:A5 FF A6 FE C9 0D D0 01 BC
0BF1:60 C9 9D D0 05 88 30 D2 E8
0BF9:10 D2 C9 14 F0 47 C4 F7 6D
0C01:B0 CA C9 C0 90 04 C9 E0 72


```

0C09:90 25 C9 1D D0 07 C4 F8 E2
0C11:B0 BA C8 90 B7 C9 20 90 08
0C19:B3 C9 40 90 10 C9 60 90 87
0C21:0A C9 94 F0 43 E0 00 F0 E0
0C29:A3 D0 69 29 BF 09 80 91 BB
0C31:F9 29 7F E0 00 D0 02 A5 7B
0C39:FF 99 6F 12 C8 C4 F8 90 A2
0C41:8B 84 F8 B0 87 C0 00 F0 9A
0C49:1C 84 FF B1 F9 BE 6F 12 67
0C51:88 91 F9 8A 99 6F 12 A4 4D
0C59:FF C4 F8 B0 03 C8 90 E9 13
0C61:C6 F8 A4 A5 88 4C CD 0B 26
0C69:A5 F8 C5 F7 B0 23 A8 84 B2
0C71:FF B1 F9 BE 6F 12 C8 91 08
0C79:F9 8A 99 6F 12 A4 FF C4 43
0C81:A5 F0 03 88 10 E9 A9 A0 AD
0C89:91 F9 A9 20 99 6F 12 E6 B5
0C91:F8 4C CD 0B C9 86 F0 14 02
0C99:C9 8C B0 F5 C9 87 D0 01 3E
0CA1:60 C9 8A 90 EC F0 05 68 54
0CA9:68 4C 75 0B A4 F8 CC 23 2E
0CB1:08 B0 0E 20 95 0B A2 9A 76
0CB9:20 A7 0B 20 91 0A 4C CB 48
0CC1:0B A2 A0 8E 00 04 C9 8A 33
0CC9:F0 06 20 9F 0F 4C CB 0B 26
0CD1:A9 00 85 FE 20 2A 0F A9 D0
0CD9:FA 8D 00 04 A5 FE F0 06 24
0CE1:AD 21 08 8D 00 D8 E6 FE 23
0CE9:4C CB 0B A5 FD D0 01 60 6C
0CF1:20 B6 0B 38 A5 FD E5 F8 96
0CF9:F0 05 B0 24 4C AB 0D A2 F1
0D01:00 18 8A A8 65 A6 85 FF CD
0D09:BD 6F 12 91 FB A4 FF 91 3D
0D11:A8 AD 20 08 91 AA E8 E4 5D
0D19:F8 90 E7 86 FD 4C 75 0B 50
0D21:85 FF A5 F8 D0 02 E6 FF 9E
0D29:A5 AE E5 FF 85 AE A5 AF 60
0D31:E9 00 85 AF A5 FC 85 FA 13
0D39:A6 FB A5 F8 D0 07 CA E0 03
0D41:FF D0 02 C6 FA 86 F9 A4 C6
0D49:FF B1 F9 A0 00 91 F9 A6 F9
0D51:F9 E8 D0 02 E6 FA E4 AE 78
0D59:D0 EB A5 FA C5 AF D0 E5 AF
0D61:A6 A7 E8 8A 0A A8 38 B9 9B
0D69:40 03 E5 FF 99 40 03 C8 BD
0D71:B9 40 03 E9 00 99 40 03 61
0D79:E0 19 90 E6 18 A5 A6 85 F4
0D81:A5 65 FF A8 84 FF B1 A8 82
0D89:48 B1 AA A4 A5 91 AA 68 05
0D91:91 A8 E6 A5 A4 FF C8 C0 4D

```

ØD99:28 90 E9 A4 A5 20 C0 0E B0
ØDA1:A5 F8 F0 03 4C 00 0D 4C E3
ØDA9:75 0B 49 FF 85 FF E6 FF 64
ØDB1:A5 AE 85 F9 65 FF 85 AE 7F
ØDB9:A5 AF 85 FA 69 00 85 AF F8
ØDC1:A0 00 B1 F9 A4 FF 91 F9 44
ØDC9:A6 F9 D0 02 C6 FA CA 86 2E
ØDD1:F9 E4 FB D0 EB A5 FA C5 60
ØDD9:FC D0 E5 A6 A7 E8 8A 0A CD
ØDE1:A8 18 B9 40 03 65 FF 99 D8
ØDE9:40 03 C8 B9 40 03 69 00 7A
ØDF1:99 40 03 E0 19 90 E6 20 50
ØDF9:00 0D A2 06 B5 A5 9D 3B C6
ØE01:03 CA D0 F8 86 A6 A6 A7 BE
ØE09:E8 8A 20 38 0B 85 FA A5 CD
ØE11:FB 85 F9 20 36 0B 20 3B 27
ØE19:0E B0 0C A5 FB C5 F9 D0 FF
ØE21:E5 A5 FC C5 FA D0 DF A2 13
ØE29:06 BD 3B 03 95 A5 CA D0 F8
ØE31:F8 A8 4C 3A 0A 20 9B 09 32
ØE39:B0 39 20 66 09 AA C0 00 DA
ØE41:F0 20 18 98 65 A6 C9 28 EB
ØE49:B0 29 CC 23 08 90 03 20 7C
ØE51:2A 0F A0 00 84 FF 20 A9 68
ØE59:0E A4 FF C8 C4 FD 90 F4 66
ØE61:B0 D3 20 AD 0E A5 A6 C9 C7
ØE69:28 B0 05 8A C9 1F D0 C5 41
ØE71:20 9B 09 A4 A6 20 C0 0E 35
ØE79:A9 00 85 A6 E6 A7 A5 A7 4E
ØE81:0A A8 A5 FB 99 40 03 C8 DD
ØE89:A5 FC 99 40 03 20 90 09 B1
ØE91:B0 15 A5 A7 C9 19 B0 0F 9D
ØE99:A5 A8 69 28 85 A8 85 AA E6
ØEA1:90 05 E6 A9 E6 AB 18 60 35
ØEA9:B1 FB 05 FE A4 A6 91 A8 B9
ØEB1:AD 20 08 05 FE 10 03 AD E9
ØEB9:21 08 91 AA E6 A6 60 C0 98
ØEC1:28 90 01 60 A9 20 91 A8 D5
ØEC9:AD 1F 08 91 AA C8 D0 EF A8
ØED1:20 20 0B A2 01 18 A5 A8 ED
ØED9:85 F7 85 F9 69 28 85 A8 A6
ØEE1:85 AA A5 A9 85 F8 A5 AB C1
ØEE9:85 FA 90 04 E6 A9 E6 AB 31
ØEF1:A0 27 B1 A8 91 F7 B1 AA 63
ØEF9:91 F9 88 10 F5 E8 E0 18 9C
ØF01:90 D3 A2 01 8A 0A A8 B9 48
ØF09:42 03 99 40 03 C8 B9 42 31
ØF11:03 99 40 03 E8 E0 19 90 DC
ØF19:EB A0 00 20 C0 0E 20 25 FA
ØF21:0F 20 3B 0E A9 18 4C 34 91

```

0F29:0B 88 84 FF AD 6F 12 09 D7
0F31:40 8D 5E 12 A9 01 A2 08 5E
0F39:A0 08 20 BA FF A9 07 A2 B0
0F41:5C A0 12 20 BD FF 20 C0 E8
0F49:FF A2 01 20 C6 FF 20 CF 78
0F51:FF A0 00 A5 90 D0 31 20 3C
0F59:CF FF C8 C4 FF 90 06 C9 DC
0F61:80 90 18 29 7F C9 80 B0 4E
0F69:E8 8D 73 0F B9 6F 12 29 97
0F71:3F C9 00 D0 06 C4 FF 90 82
0F79:DA F0 1B A5 90 D0 09 20 F8
0F81:CF FF C9 80 B0 CB 90 F3 92
0F89:C9 40 F0 06 20 3B 11 4C 77
0F91:75 0B A9 80 85 FE A9 01 E6
0F99:20 C3 FF 4C CC FF 88 84 79
0FA1:FF AD 6F 12 09 40 8D 65 04
0FA9:12 A9 01 A2 08 A0 08 20 78
0FB1:BA FF A9 05 A2 63 A0 12 A8
0FB9:20 BD FF 20 C0 FF A2 01 A5
0FC1:20 C9 FF A0 00 C8 B9 6F 72
0FC9:12 29 3F C4 FF 90 02 09 BE
0FD1:80 20 D2 FF C4 FF 90 ED C7
0FD9:20 3B 11 F0 03 4C 91 0A 7E
0FE1:20 97 0F 20 91 0A A2 E6 BA
0FE9:86 FE 20 9A 0B 60 A2 4E 26
0FF1:20 97 0B A2 30 20 A7 0B ED
0FF9:C9 0D D0 43 A2 B7 20 97 5A
1001:0B A2 C0 20 9A 0B 20 B2 5D
1009:0B A5 F8 F0 32 20 1A 11 9D
1011:D0 EA A9 01 A2 08 A0 01 11
1019:20 BA FF 18 A9 02 65 F8 92
1021:A2 6D A0 12 20 BD FF A9 C4
1029:B0 A6 AE A4 AF 20 D8 FF 1B
1031:90 05 20 3B 11 D0 C5 20 0A
1039:91 0A A2 ED 20 A7 0B 4C D9
1041:75 0B A2 BC 20 97 0B 20 94
1049:B2 0B A5 F8 F0 F1 20 1A 73
1051:11 D0 EF A9 01 A2 08 A0 0A
1059:08 20 BA FF A2 00 A4 F8 34
1061:BD 68 12 99 6F 12 C8 E8 94
1069:E0 04 90 F4 C8 C8 98 A2 99
1071:6D A0 12 20 BD FF 20 C0 A3
1079:FF A2 01 20 C6 FF 20 CF AA
1081:FF 85 B0 85 AE A5 90 D0 6F
1089:2B 20 CF FF 85 B1 85 AF EE
1091:C9 20 90 13 A0 00 A5 90 C2
1099:D0 16 20 CF FF 91 AE E6 33
10A1:AE D0 F3 E6 AF D0 EF 20 FA
10A9:92 0B E8 20 A7 0B D0 92 92
10B1:C9 40 F0 05 20 3B 11 D0 16

```

10B9:89 20 97 0F A2 2A 20 97 20
10C1:0B 20 E4 FF C9 0D D0 F9 2A
10C9:20 1A 11 D0 EF A9 01 A2 7A
10D1:08 A0 08 20 BA FF A9 02 4C
10D9:A2 5B A0 12 20 BD FF 20 6F
10E1:C0 FF A2 01 20 C6 FF A0 83
10E9:8E 20 CF FF 88 D0 FA A5 76
10F1:90 F0 05 20 3B 11 D0 C4 BD
10F9:20 CF FF 29 3F 09 80 D9 A9
1101:98 11 D0 0B C8 C0 0F 90 76
1109:EF 20 97 0F 4C FE 0A 20 A1
1111:92 0B A2 86 20 A7 0B D0 82
1119:A3 A9 0F 20 C3 FF 20 CC 86
1121:FF A9 A0 8D 00 04 A9 0F 0D
1129:A8 A2 08 20 BA FF A9 03 77
1131:A2 6C A0 12 20 BD FF 20 0D
1139:C0 FF 20 CC FF A2 0F 20 55
1141:C6 FF 20 CF FF 85 FF C9 A7
1149:30 D0 03 4C CC FF 20 95 19
1151:0B A5 FF 09 80 99 00 04 61
1159:C8 20 CF FF C9 0D D0 F3 F9
1161:20 97 0F 4C AA 0B D3 90 D9
1169:85 85 84 FA BA A0 D3 90 80
1171:85 8C 8C 89 8E 87 A0 C3 3B
1179:88 85 83 8B 85 92 A0 86 A8
1181:8F 92 A0 D3 90 85 85 84 8B
1189:D3 83 92 89 90 94 20 00 78
1191:C9 8E 93 85 92 94 A0 84 B3
1199:89 83 94 89 8F 8E 81 92 D8
11A1:99 A0 84 89 93 8B A0 A6 94
11A9:A0 88 89 94 A0 D2 C5 D4 69
11B1:D5 D2 CE 00 D2 85 8D 8F A4
11B9:96 85 00 D0 8C 85 81 93 A6
11C1:85 A0 97 81 89 94 A0 AE 68
11C9:AE AE 00 C5 92 92 8F 92 DB
11D1:BA A0 00 C9 8E 96 81 8C 74
11D9:89 84 A0 93 94 81 92 94 93
11E1:89 8E 87 A0 81 84 84 92 21
11E9:85 93 93 00 C9 8E 83 8F 45
11F1:92 92 85 83 94 A0 84 89 A4
11F9:93 8B A0 8E 81 8D 85 00 13
1201:CD 95 93 94 A0 88 81 96 ED
1209:85 A0 81 94 A0 8C 85 81 55
1211:93 94 A0 80 A0 8C 85 94 17
1219:94 85 92 93 00 D3 81 96 5D
1221:85 00 CC 8F 81 84 A0 86 80
1229:89 8C 85 8E 81 8D 85 BA D6
1231:A0 00 C5 98 89 94 BA A0 9C
1239:81 92 85 A0 99 8F 95 A0 54

```

1241:93 95 92 85 A0 9B D9 AF 16
1249:CE 9D BF 00 81 84 84 85 E0
1251:84 AE 00 93 81 96 85 84 92
1259:AE 00 24 30 3A 00 2C 53 D9
1261:2C 52 30 3A 00 2C 41 2C 39
1269:50 2C 52 49 30 3A 12 00 2E

```

SpeedView

An 80-Column Preview Program for *SpeedScript*

Mark Schreiner

"SpeedView," an enhancement for COMPUTE!'s SpeedScript word processor (versions 3.0 and higher), provides 80-column previewing of text before the actual printing. The program shows exactly how each page of the document will look when printed. A disk drive is required to create the combined file containing both SpeedScript and SpeedView.

SpeedScript, COMPUTE!'s popular word processor, becomes an even more useful resource with this program. "SpeedView" lets you preview your text file in 80 columns so you can see precisely how a printout of your document will look.

Create a Program

Because SpeedView is written entirely in machine language, it must be entered using the "MLX" machine language entry program found in Appendix D. Be sure you read and understand the instructions for using MLX before you begin entering the SpeedView data from Program 1. When you run MLX, you'll be asked for a starting address and an ending address for the data you'll be entering. Use the following values for SpeedView:

Starting address: 2500

Ending address: 26B7

After you finish entering all the data from Program 1, be sure to save a copy before you leave MLX.

To use SpeedView, load *SpeedScript* but *do not run it yet*.

Next, load SpeedView with the command **LOAD"SPEED-VIEW",8,1**, replacing SPEEDVIEW with the name you used when you saved the data with MLX (tape users should substitute ,1,1 for ,8,1 in this command). Finally, type **SYS 9480** and press RETURN. This activates *SpeedScript* with the SpeedView enhancement.

Two for One

If you plan to use SpeedView regularly, you can create a combined file which contains both *SpeedScript* and SpeedView. Like *SpeedScript*, the unified file loads and runs just like an ordinary BASIC program. To create this file, type in and save Program 2. Reset the computer by turning it off and on; then enter this command and press RETURN:

POKE 44,48:POKE 48*256,0:NEW

Load Program 2 back into memory and insert a disk containing copies of both *SpeedScript* and SpeedView. When you run Program 2, it asks you to enter the names of the *SpeedScript* and SpeedView files on the disk in the drive. After you've entered these filenames, you're prompted to enter a name for the new, combined file. Program 2 reads both files into memory, modifies the BASIC portion of *SpeedScript*, then writes the unified package back to disk using the filename you selected.

To use this combined program, load and run it as you would any BASIC program. You now have a copy of *SpeedScript* with SpeedView permanently installed.

SpeedView Operation

To use SpeedView, press CTRL-SHIFT-P, followed by S, the command sequence to direct *SpeedScript*'s output to the screen. Instead of the usual jumble of scrolling lines, SpeedView presents a neatly formatted representation of the document's first full page. Press RETURN to view succeeding pages. SpeedView shows exactly how each page will look when printed on paper. When the last page has been displayed, press any key to return to *SpeedScript*. The screen preview option is the only *SpeedScript* command changed by SpeedView. All others function normally.

Never press RESTORE while previewing a document with SpeedView. If you must exit *SpeedScript*, press RETURN until you've reached the end of the SpeedView display, and return to *SpeedScript*; then press RESTORE to exit. If you don't use the combined file, remember that the proper sequence is

- Load *SpeedScript*
- Load SpeedView
- Activate SpeedView and *SpeedScript* with the SYS 9480 command

Some printkey values may not show up during the 80-column preview. SpeedView displays only those characters whose Commodore ASCII values are in the range 32–90, or 193–218, inclusive. This includes the uppercase and lowercase alphabets, numerals, and punctuation marks. The SpeedView program code occupies 1.5K of space that's otherwise available for text memory. As a result, you may not be able to preview a very long document without breaking it into two smaller files.

SpeedView behaves erratically if you change the page length to any value other than 66, or the right margin to any value greater than 80. To obtain the best results, make sure to use those settings.

Program 1. SpeedView

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```
2500:00 00 00 00 00 00 00 00 4A
2508:A2 C4 A0 25 8E 26 03 8C DA
2510:27 03 A9 26 8D A9 09 A9 15
2518:CB 8D B1 09 A9 00 8D A4 7F
2520:1F A2 94 A0 26 8E 6E 17 9E
2528:8C 6F 17 A2 A3 A0 26 8E 1C
2530:89 18 8C 8A 18 A9 20 8D B4
2538:88 18 4C 0D 08 A9 00 8D 9B
2540:02 25 8D 03 25 AD 04 25 C3
2548:18 69 03 8D 04 25 EE 05 C9
2550:25 AD 05 25 C9 42 D0 0A 8E
2558:20 E4 FF C9 0D D0 F9 20 48
2560:65 25 4C F8 25 A9 17 8D 4B
2568:18 D0 A9 1B 8D 11 D0 A9 D5
2570:97 8D 00 DD A9 00 8D 00 30
2578:25 60 A9 94 8D 00 DD A9 BD
2580:3B 8D 11 D0 A9 38 8D 18 5C
2588:D0 8D 00 25 A9 E0 8D A6 83
2590:25 A9 00 8D 05 25 8D A5 2E
2598:25 8D 02 25 8D 03 25 8D BB
25A0:04 25 AA 8A 9D 00 E0 E8 CB
25A8:D0 FA EE A6 25 AD A6 25 B4
25B0:D0 F1 A9 10 9D 00 CC 9D 3A
25B8:00 CD 9D 00 CE 9D 00 CF E6
25C0:E8 D0 F1 60 48 8D 01 25 97
25C8:98 48 8A 48 AD 07 25 F0 0C
25D0:2D AD 00 25 D0 03 20 7A BC
25D8:25 AD 01 25 C9 0D D0 03 BA
25E0:4C 3D 25 C9 20 D0 03 4C 78
25E8:12 26 90 0C C9 DB B0 08 BF
25F0:C9 C1 B0 12 C9 5B 90 0E B2
```



```

25F8:68 AA 68 A8 68 60 68 AA F9
2600:68 A8 68 4C CA F1 20 26 01
2608:26 EE 04 25 20 26 26 CE AA
2610:04 25 A9 04 18 6D 02 25 BC
2618:8D 02 25 AD 03 25 69 00 AA
2620:8D 03 25 4C F8 25 A5 FE 04
2628:48 A5 FF 48 AD 04 25 4A 98
2630:4A 4A 85 FF A0 00 84 FE F1
2638:4A 66 FE 4A 66 FE 65 FF C1
2640:85 FF AD 04 25 29 07 18 39
2648:65 FE 85 FE A5 FF 69 E0 88
2650:85 FF AD 02 25 29 F8 18 0D
2658:65 FE 85 FE A5 FF 6D 03 C2
2660:25 85 FF AD 02 25 29 07 79
2668:AA A9 00 38 6A CA 10 FC 93
2670:8D 06 25 4A 0D 06 25 4A 63
2678:0D 06 25 48 A0 00 A6 01 49
2680:A9 34 78 85 01 68 11 FE E0
2688:91 FE 86 01 58 68 85 FF AD
2690:68 85 FE 60 48 E0 03 D0 F4
2698:05 A9 01 8D 07 25 68 20 88
26A0:BA FF 60 8D 1D 0D 48 A9 86
26A8:00 8D 07 25 20 65 25 68 D4
26B0:60 00 00 00 00 00 00 2D

```

Program 2. Unified File Maker

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

```

RJ 100 IFA=0THENINPUT "{CLR}NAME OF SPEEDSCRIPT";SS
      $:INPUT "NAME OF SPEEDVIEW";SV$
AE 120 IFA=0THENA=1:INPUT "NAME OF COMBINED FILE";N
      F$:PRINT "{DOWN}LOADING FILES"
GQ 130 IFA=1THENA=2:LOADSS$,8,1
SF 140 IFA=2THENA=3:LOADSV$,8,1
MQ 150 A$="9480":FORI=0TO3:POKE2054+I,ASC(MID$(A$,
      I+1,1)):NEXT
QS 155 PRINT "{DOWN}WRITING COMBINED FILE":OPEN1,8,
      8,"0:"+NF$+"",P,W"
JX 165 PRINT#1,CHR$(1);CHR$(8);
HS 170 FORI=2049TO9910:PRINT#1,CHR$(PEEK(I));:NEXT
      :CLOSE1:END

```

Word Counter

Thomas K. Tucker

Need a quick word count of a document? "Word Counter" works with text files—program or sequential—created by almost any word processor, including SpeedScript. Disk drive required.

Teachers are fond of giving assignments in terms of words: a "3,000-word term paper" or a "500-word essay," for instance. I recently wrote such a paper using the word processor *SpeedScript*, but when I finished, I had no idea of its length in number of words. It seemed to me it would be a fairly easy task to write a program to count the words in a file, but first I had to determine what constituted a *word*.

Spaces separate words from neighboring words, so the number of spaces in a document should equal the number of words. The only snag is when multiple spaces appear in the file. Thus the program shouldn't count *all* the spaces, just those immediately preceded by a character that wasn't a space.

The BASIC program looked something like this:

```
10 Z=0:A$="":B$=""
20 INPUT"FILENAME";F$
30 OPEN 1,8,0,F$+"",P,R"
40 GET#1,A$
50 IF 64 AND ST THEN 90
60 IF A$=CHR$(32) THEN IF B$<>CHR$(32) THEN Z=Z+1
70 B$ = A$
80 GOTO 40
90 CLOSE 1
100 PRINT"NUMBER OF WORDS IN FILE : ";Z+2
110 END
```

Line 50 checks for the end of the file. Line 60 rules out counting consecutive spaces as more than one word. By experiment, a more accurate count was shown when 2 was added to the counter (Z). Since printer format codes and carriage returns are counted as words, a 100-percent-accurate count isn't possible. But it's rarely important that the final number of words be that exact.

Speeding It Up

That BASIC program took over four minutes to count about 2500 words. Much too slow.

Writing the loop part of the program (lines 40–80) in machine language (ML) seemed to be the answer. Since it's a short routine, it fits nicely into the cassette buffer at \$033C (828). The ML data is POKEd into the cassette buffer using DATA statements.

The directory routine and the option of counting sequential as well as program files were added later. This program should read files written on any word processor—but remember, the more printer code strings used in the file, the less accurate the word count. In any case, the program is fast, taking about 40 seconds to count a 2500-word, 60-block file.

How To Use It

Word Counter is easy to use. Type in the program and save a copy to disk. As listed, the program works on the 64 or 128 in 64 mode. To use the program on a 128 in 128 mode, substitute the following lines:

```
10 PRINT "{CLR}"CHR$(142):BS=3072:COLOR0,7:COLOR
  4,7
340 DATA 169,0,141,0,12,141,1,12,141,66,12,141,
  67,12,162,1,32
350 DATA 198,255,32,183,255,41,64,208,34,32,207
  ,255,141,66,12,201,32,208,15
360 DATA 32,207,255,201,32,240,8,238,0,12,208,3
  ,238,1,12,173,66,12,141
370 DATA 67,12,76,21,12,32,231,255,96,3,4,-1
```

After you've finished writing and saving your document, load Word Counter and type **RUN**. The first prompt is **PRESS D FOR DIRECTORY**. Insert the disk containing the text file and press D. You're then asked to type in the filename and to type P (Program) or S (Sequential) for file type. Word Counter reads through the file and seconds later displays the number of words. You're then asked if you'd like to count the words in another file.

Editor's Note: We tested Word Counter with text files created by SpeedScript (64), Paperback Writer (128 and 64 versions), and Word Writer 128. The program gave a reasonably accurate count with these files (program or sequential), which were of varying length.

Word Counter

For mistake-proof program entry, be sure to use "The Automatic Proofreader" (Appendix C).
128 Users—See instructions in article before typing in.

```

JM 10 PRINT "{CLR}{CYN}"CHR$(142):BS=828:POKE53280,
    6:POKE53281,6
CP 20 I=I+1:READA:IFA<0THEN50
QQ 30 POKEBS+1+I,A
AC 40 GOTO20
RA 50 Z=0:D$=""
EF 60 PRINT "{CLR}{2 DOWN}PRESS D FOR DIRECTORY"
KG 70 GETA$:IFA$=""THEN70
FX 80 IFA$<>"D"THEN160
XG 90 OPEN1,8,0,"$0"
ES 100 PRINT:FORA=1TO32:GET#1,C$:NEXT
KR 110 GET#1,B$:IFST<>0THENCLOSE1:SYS65484:GOTO160
BC 120 IFB$<>CHR$(34)THEN110
PC 130 GET#1,B$:IFB$<>CHR$(34)THEND$=D$+B$:GOTO130
MB 140 GET#1,B$:IFB$=CHR$(32)THEN140
AE 150 PRINT " ";B$;" {3 SPACES}";D$:D$="":GOTO110
HD 160 INPUT "{2 DOWN}FILE NAME";F$
AJ 170 IFF$=""THENPRINT "{4 UP}":GOTO160
FE 180 PRINT "{CLR}{2 DOWN}FILE TYPE?"
EM 190 PRINT "{2 DOWN}{RVS}P{OFF}ROGRAM"
GM 200 PRINT "{DOWN}{RVS}S{OFF}EQUENTIAL"
HR 210 GETG$:IFG$<>"P"ANDG$<>"S"THEN210
KX 220 PRINT "{2 DOWN}COUNTING"
BR 230 IFG$="S"THEN250
JK 240 OPEN1,8,0,F$+" ,P,R":GOTO260
MP 250 OPEN1,8,0,F$+" ,S,R"
FA 260 SYSBS+2
AK 270 Z=PEEK(BS)+256*PEEK(BS+1)+2
RC 280 PRINT "{CLR}{DOWN}NUMBER OF WORDS:"Z:CLOSE1
DF 290 OPEN15,8,15,"I0":CLOSE15
GP 300 PRINT "{DOWN}ANOTHER FILE?{2 SPACES}(Y/N)"
AQ 310 GETA$:IFA$="Y"THEN50
AB 320 IFA$="N"THENEND
CG 330 GOTO310
JR 340 DATA169,0,141,58,3,141,59,3,141,60,3,141,61,
    3,162,1,32,198,255,32
KR 350 DATA183,255,41,64,208,34,32,207,255,141,58,
    3,201,32,208,15,32,207,255
CQ 360 DATA201,32,240,8,238,60,3,208,3,238,61,3,17
    3,58,3,141,59,3,76,81
CX 370 DATA3,32,231,255,96,-1

```

Sequential File Converter for *SpeedScript*

Ronald Carnell

SpeedScript becomes an even more valuable word processor with this program that converts sequential files into SpeedScript format. It's written in machine language for fast results.

I've been using *SpeedScript*, in one version or another, since it was originally published in the January 1984 issue of *COMPUTE!'s Gazette*. The only drawback I've ever found is its incompatibility with sequential files. (*SpeedScript* stores documents in program files.)

My database program creates sequential files, and many terminal programs download text in sequential file format. And I have a few friends who use other word processors that store documents in sequential files. In short, there are many instances when I'd like to load a sequential file into *SpeedScript*.

Over the years, *COMPUTE!* magazine and *COMPUTE!'s Gazette* have published several BASIC programs designed to convert sequential files into *SpeedScript* files. And I've used them, mostly with great success. But BASIC is inherently slow—when my database started creating sequential files over 100 blocks long, it became obvious that I needed something faster.

"Sequential File Converter" (SFC), the program accompanying this article, is the solution. SFC accomplishes the same thing as the earlier BASIC conversion programs, but at the speed of machine language.

To type in the program, you'll need "MLX," the machine language entry program found in Appendix D. After loading and running MLX, answer the starting and ending address prompts with the following:

Starting address: 0801
Ending address: 0AD0

Although most of Sequential File Converter is written in machine language, you need only load it and type **RUN** as you would for a BASIC program. You'll be prompted for the name of the source file (the sequential file) and for the name you'd like for the new *SpeedScript* file. (Be sure to have the correct disk in the drive when you enter the destination name.)

If you load SFC and type LIST, you'll find several lines of BASIC. With one exception, you should *never* change any of this code. The machine language is attached to the end of the BASIC program, and changing any of the BASIC code will almost certainly disrupt the machine language and cause SFC to crash. The exception is line 140. In this line, SFC is adding ,S,R to the end of your source filename. The ,S tells the disk drive to access a sequential file, and the ,R indicates that the file is to be read. You can, in this instance, change the S to another letter without changing the length of SFC and therefore without causing the machine language to crash. You might wish to do this if you want to change a file format other than sequential into a *SpeedScript* file. For example, a few word processors store their text as *USR* files. By changing that S to a U, you could make SFC a *USR* file converter able to access *USR* files.

Sequential File Converter for *SpeedScript*

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```
0801:25 08 64 00 53 4E B2 32 9D
0809:37 31 36 3A 53 4C B2 32 CE
0811:37 33 36 3A 44 4E B2 32 E6
0819:37 33 39 3A 44 4C B2 32 47
0821:37 35 39 00 48 08 6E 00 80
0829:99 22 93 11 11 05 22 A6 99
0831:31 32 29 22 43 4F 44 45 D2
0839:20 43 4F 4E 56 45 52 53 B8
0841:49 4F 4E 11 11 22 00 7E 34
0849:08 78 00 99 22 50 4C 45 45
0851:41 53 45 20 49 4E 50 55 FA
0859:54 20 53 4F 55 52 43 45 BA
0861:20 50 52 4F 47 52 41 4D 28
0869:27 53 20 44 49 53 4B 20 78
0871:4E 41 4D 45 3A 22 3A 85 4B
0879:20 53 4E 24 00 B1 08 82 D3
0881:00 8B 20 C3 28 53 4E 24 04
0889:29 B1 31 36 20 A7 20 99 9D
0891:22 54 48 41 54 20 4E 41 E5
```

```

0899:4D 45 20 49 53 20 54 4F 4D
08A1:4F 20 4C 4F 4E 47 2E 2E F9
08A9:2E 22 3A 89 31 32 30 00 EB
08B1:C4 08 8C 00 53 4E 24 B2 86
08B9:53 4E 24 AA 22 2C 53 2C CA
08C1:52 22 00 F2 08 96 00 81 CE
08C9:20 49 B2 30 20 A4 20 C3 2D
08D1:28 53 4E 24 29 AB 31 3A 6B
08D9:97 20 53 4E AA 49 2C C6 A6
08E1:28 CA 28 53 4E 24 2C 49 97
08E9:AA 31 2C 31 29 29 3A 82 19
08F1:00 28 09 A0 00 99 22 11 F2
08F9:50 4C 45 41 53 45 20 49 3B
0901:4E 50 55 54 20 44 45 53 2E
0909:54 49 4E 41 54 49 4F 4E 2A
0911:20 50 52 4F 47 52 41 4D D9
0919:27 53 20 4E 41 4D 45 22 68
0921:3A 85 20 44 4E 24 00 5B 58
0929:09 AA 00 8B 20 C3 28 44 C7
0931:4E 24 29 B1 31 36 20 A7 FD
0939:20 99 22 54 48 41 54 20 5B
0941:4E 41 4D 45 20 49 53 20 B5
0949:54 4F 4F 20 4C 4F 4E 47 C8
0951:2E 2E 2E 22 3A 89 31 36 7E
0959:30 00 6E 09 B4 00 44 4E 5E
0961:24 B2 44 4E 24 AA 22 2C DB
0969:50 2C 57 22 00 9C 09 BE FE
0971:00 81 20 49 B2 30 20 A4 B7
0979:20 C3 28 44 4E 24 29 AB D6
0981:31 3A 97 20 44 4E AA 49 A9
0989:2C C6 28 CA 28 44 4E 24 28
0991:2C 49 AA 31 2C 31 29 29 16
0999:3A 82 00 C3 09 C8 00 97 A8
09A1:20 53 4C 2C C3 28 53 4E 98
09A9:24 29 AA 32 3A 97 20 44 45
09B1:4C 2C C3 28 44 4E 24 29 BC
09B9:AA 32 3A 9E 20 32 35 30 43
09C1:31 00 00 00 A2 08 A9 0F 04
09C9:A8 20 BA FF A9 02 A2 C8 F2
09D1:A0 0A 20 BD FF 20 C0 FF 98
09D9:A9 0F 20 C3 FF AD B0 0A E6
09E1:A2 9A A0 0A 20 BD FF A9 42
09E9:02 A2 08 A8 20 BA FF 20 3D
09F1:C0 FF AD C7 0A A2 B1 A0 75
09F9:0A 20 BD FF A9 03 A2 08 77
0A01:A8 20 BA FF 20 C0 FF A9 76
0A09:00 85 90 A2 03 20 C9 FF E6
0A11:A9 01 20 D2 FF A9 08 20 42
0A19:D2 FF 20 CC FF A2 02 20 16
0A21:C6 FF 20 CF FF 85 FB A4 4C

```

```

0A29:90 84 FC 20 CC FF A5 FB F5
0A31:C9 0D D0 04 A9 1F D0 3E 71
0A39:C9 20 B0 04 A9 3F D0 36 B2
0A41:C9 40 90 32 C9 60 B0 06 B6
0A49:38 E9 40 4C 77 0A C9 80 B8
0A51:B0 06 38 E9 20 4C 77 0A 10
0A59:C9 A0 B0 05 A9 3F 4C 77 3B
0A61:0A C9 C0 B0 06 38 E9 40 35
0A69:4C 77 0A C9 FF D0 04 A9 54
0A71:5E D0 03 38 E9 80 85 FB 25
0A79:A2 03 20 C9 FF A5 FB 20 EE
0A81:D2 FF 20 CC FF A4 FC C0 1D
0A89:00 F0 92 A9 02 20 C3 FF DE
0A91:A9 03 20 C3 FF 20 CC FF 95
0A99:60 30 3A 20 20 53 42 2E 34
0AA1:44 49 53 4B 20 52 45 50 6E
0AA9:4F 52 54 2C 53 2C 52 14 4B
0AB1:30 3A 20 20 53 53 20 44 DE
0AB9:49 53 4B 20 52 45 50 4F 4A
0AC1:52 54 2C 50 2C 57 14 49 CE
0AC9:30 A4 00 00 00 00 00 00 1F

```


Chapter 5

For Programming

Sprint

A Compiler for the 64

Robert A. Stoerrle

Here's a BASIC compiler that can make your programs execute up to 50 times faster. Written in machine language, and extremely easy to use, "Sprint" supports most Commodore BASIC commands, statements, and functions. Disk drive required.

What's the chief complaint when it comes to Commodore BASIC? Speed, or more precisely, the lack of it. Yet the language itself is not what earns BASIC its reputation as painstakingly slow; it's the way that it's implemented. BASIC is *interpreted*. Most faster languages are *compiled*.

Whether a program in a high-level programming language is interpreted or compiled, each statement has to be translated into the machine language of the computer. With an interpreter, this has to be done every time the statement is executed. This slows down execution speed considerably. However, a compiler translates the entire program once—before execution. So, when you run the program, it executes much faster than a comparable interpreted program.

"Sprint" is a BASIC compiler that supports a subset of BASIC statements available on the Commodore 64. It supports most of the Commodore statements, including LET, FOR, NEXT, IF, POKE, PRINT, READ, DATA, and INPUT. It supports integer numbers (but no fractions), numeric and string variables, and one-dimensional numeric arrays. Sprint also supports a wide range of functions, including LEFT\$, RIGHT\$, MID\$, PEEK, STR\$, and VAL.

Getting Ready to Sprint

Sprint is written entirely in machine language and must be typed using "MLX," a machine language entry program found in Appendix D.

After loading MLX, run it. When prompted for the starting and ending addresses, enter these values:

Starting Address: 8000

Ending Address: 9737

Since Sprint is a very large program (almost 6K of pure machine language), you may want to type it in more than one sitting. Be sure to follow the instructions in Appendix D to save your work between sittings, and always load up your previous work before typing further.

Once the program has been typed in and saved, it can be loaded with the following statement:

LOAD"SPRINT",8,1

To run it, type

SYS 32768

Using Sprint

The first step, of course, is to write the BASIC program you want to compile. You can do this with the normal BASIC built into your 64. If you wish, you can test the program on the interpreter first by running it. This will not always work, as Sprint has a slightly different syntax for certain keywords.

When you're ready to compile the program, save it to disk (Sprint compiles a program from disk, not from memory). Load Sprint and type **SYS 32768**. You'll be prompted for the filename of your BASIC program. When you enter it, Sprint will proceed to compile your program. As the program is compiled, each line will be listed to the screen, followed by any error messages for the line. Note that you can stop the compiler simply by pressing the STOP key. If the compilation finishes with no errors, you can simply type RUN to execute the compiled program. However, if there are errors, you'll have to load the original BASIC program (the source program) and make the necessary changes.

Sprint compiles programs from the beginning and continues straight to the end—it does not follow GOTOs or GOSUBs, or evaluate IF/THEN statements. Because of this, you can't have conditional DIMs (IF...THEN DIM...) or conditional FORs.

Sprint is a two-pass compiler—it scans the program twice. Once a program has been compiled, there's no need for the compiler to be in memory in order to run the program; Sprint programs are self-sufficient.

After the source program is compiled, the object program (the resulting code) will reside in memory as if it were an ordinary BASIC program. You can run and save it, but if you type LIST, you'll see:

10 SYS 2061

This is another advantage of using a compiler—people can't easily LIST or modify your program. Editing can only be done on the source program.

Constants, Variables, and Expressions

Sprint works much the same as regular Commodore BASIC. However, there are some differences you should keep in mind when you're writing a program.

Constants are data values used in BASIC programs, such as -1, 4353, and "HELLO". Sprint supports two types of constants: integer and string. Integer numbers are limited to the range -32768-32767. However, numbers greater than that are allowed as addressed in POKE, PEEK, and SYS statements. String constants are characters enclosed in quotation marks. The closing quote may be omitted if it's at the end of a line.

Sprint supports both numeric and string variables. Variable names may be of any length, but must not contain any *reserved words* (words that are significant to the compiler, such as POKE and THEN). Only the first 15 characters are looked at, however. Integer variables may be terminated by a percent sign (%) if desired, but all regular numeric variables are assumed to be integers anyway, since there are no floating-point numbers. String variable names must be terminated with a dollar sign (\$).

Strings are limited to ten characters, unless specified otherwise. If you want to include more characters, DIMension the string, as if it were an array, to the maximum number of characters it will hold. The maximum number of characters allowed in a string is 253.

In addition to integer variables, Sprint supports one-dimensional integer arrays. Subscripts may range from 0 to 126. Arrays that are not DIMensioned are assumed to consist of 11 elements, numbered 0-10. The format for assigning, reading, and dimensioning integer arrays is identical to that for interpreted BASIC, except that the subscript of a variable

may not be another subscripted variable or an expression. It must be an integer constant or integer variable.

Numeric expressions in Sprint may contain integer constants, variables, arrays, functions, and operators. The operators supported by Sprint are $+$, $-$, $*$, and $/$, as well as the logical operators AND and OR, and the relational operators $=$, $<$, and so on. *Parentheses are not permitted.* Unlike interpreted BASIC, expressions are evaluated strictly left to right; there is no operator precedence. To get around this, you must break up the expression into several smaller expressions, and then put the results of these back together. For example, this expression:

SUM = A + X * Y + B + I * J,

becomes the following:

Q1 = X * Y:Q2 = I * J:SUM = A + Q1 + B + Q2

Strings can be compared using the relational operators as in regular BASIC. They can also be *concatenated* (one string appended to the other) using the plus sign. Note that the result of a string expression must not be longer than the number of characters allocated for the string.

When an expression is assigned to a variable, the LET may be omitted.

Statements

A complete list of the keywords available in Sprint appears in the Sprint Keywords table. Most keywords function the same as in regular BASIC, except for the following:

- **DATA.** The DATA statement has one small idiosyncrasy: Items that include spaces must be enclosed in quotes, or the item will be READ as if it doesn't contain a space (YOU ARE becomes YOUARE.)

- **DIM.** The DIM statement can only be used to declare one-dimensional numeric arrays. Multidimensional arrays are not permitted; nor are string arrays. A string is dimensioned to the maximum number of characters it will hold, like an array of numbers representing each character of the string. Only integer constants are allowed between the parentheses of variables in a DIM statement.

• **FOR.** This statement functions the same as in Commodore BASIC, but its syntax is somewhat more particular. The TO value, and the STEP value, if specified, must be either integer constants or integer variables. Expressions are not allowed. Also, every FOR statement must have one, *and only one*, matching NEXT statement.

• **GET.** In Sprint, the GET statement can be followed only by a string variable. This shouldn't be a problem, as numeric GETs are prone to errors because of non-numeric keystrokes, and the VAL function can be used to change the string to a number.

• **IF.** This statement functions a bit abnormally when AND or OR is used. Because expressions are evaluated without operator precedence, the following statement will not work:

IF A=B AND C=D THEN...

It would be evaluated as if it were

IF (A=B AND C) = D THEN...

To get around this, you can use the following modification:

X = A=B:Y = C=D:IF X AND Y THEN...

The result of the comparison $A = B$, which is -1 for true or 0 for false, is assigned to variable X. The result of $C = D$ is assigned to Y. Now X and Y can be substituted for those comparisons in the IF statement.

Sprint Keywords

ABS	INPUT	RESTORE
AND	LEFT\$	RETURN
ASC	LEN	RIGHT\$
CHR\$	LET	READ
DATA	MID\$	RND
DIM	NEXT	SGN
END	OR	SPC
FOR/TO/STEP	PEEK	STR\$
GET	POKE	SYS
GOSUB	POS	TAB
GOTO	PRINT	VAL
IF/THEN	REM	

- **INPUT.** Each INPUT statement can assign input to just one variable. In addition, all characters, except leading and trailing spaces, typed on the screen are assigned to the variable. This includes commas, colons, and quotes.

Functions

Sprint supports most Commodore functions that don't require floating-point numbers. The arguments of functions must be constants or variables, not expressions. Also, note that functions cannot be nested. For example, `ABS(RND(0))` is illegal. The following functions work differently in Sprint from the way they work in regular BASIC:

- **RND.** The RND function always returns a number in the range `-32768-32767`, no matter what its argument is. To scale the number down to size, you can divide, using the equation

$$Z=32767/\textit{upper range:N}=\text{RND}(0) \text{ AND } 32767/Z$$

The AND is necessary to insure that no negative numbers result. Sprint gets its random numbers from voice 3 of the SID chip.

- **MID\$.** This string function has two forms in Commodore BASIC: `MID$(v$,x)` and `MID$(v$,x,y)`. Only the latter form is supported by Sprint.

- **ASC.** In Commodore BASIC, using this function on a null string ("") results in an error. This does not occur in Sprint. Instead, a value of 0 is returned.

Error Messages

Sprint has many error messages which make it easier to debug programs. However, only certain types of errors can be flagged during compilation. Sprint catches errors such as data type mismatches, NEXT without FOR, and so on. Errors such as overflowing numbers and running out of DATA items cannot be flagged during compilation because they occur during execution of the actual compiled program. In these cases it will be harder, but not impossible, to debug programs.

Sometimes, an error earlier in the program leads to another error later on. For example, if there's an error in a FOR statement, the compiler will act as if that FOR doesn't exist. Because of this, the corresponding NEXT statement will cause

a NEXT WITHOUT FOR error message. You'll have to use your judgment to weed out these extraneous messages. Note that you should *never* attempt to execute a program with even a single error in it.

See the Sprint Error Messages table for more details.

Sprint Error Messages

BAD FILE—Sprint only reads program files. This error results if the filename of a sequential file is entered. This error may also happen if a line in the program file has more than 80 characters, or a read error occurs.

DISK ERROR—The disk error channel is checked prior to each pass. If there's an error, it's displayed and compilation is aborted.

DEVICE NOT PRESENT—The disk drive isn't connected to the serial bus, or it isn't turned on.

NOT SUPPORTED—A keyword was used that's legal in Commodore BASIC but illegal in Sprint.

ILLEGAL FUNCTION USE—A function was used where a statement should have been used.

ILLEGAL STATEMENT USE—A statement was used where a function should have been used.

NON-EXISTING LINE—A GOTO or GOSUB attempted to branch to a program line that doesn't exist.

SYNTAX—A misspelled keyword, extra parenthesis, and so on.

TOO MANY FORS—It's unlikely that you'll ever encounter this error, as up to 19 FOR/NEXT loops can be nested.

NEXT WITHOUT FOR—A NEXT statement attempted to end a loop which didn't have a corresponding FOR statement.

TYPE MISMATCH—Numeric data was used where string data was expected, or vice versa.

COMPILATION ABORTED—Either the STOP key was pressed or an irrecoverable error has occurred (file not found, and so on).

MISSING COMMA

MISSING PARENTHESIS

MISSING SEMICOLON

MISSING EQUAL SIGN

How Fast Is Sprint?

I ran several test programs through the compiler to determine the speed of Sprint. Results varied according to the type of program. The compiled test programs ran 15 to 50 times faster than Commodore BASIC. A program to sort 100 numbers took over two minutes in Commodore BASIC, but the same program compiled by Sprint executed in only *five seconds*.

For demonstration purposes, I've included a sample program. Type in Program 2, "64 Doodler Demo," and run it, noting the speed with which it executes. Now, save, compile, and run it again. Notice the difference?

Program 1. Sprint

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```
8000:A2 7F B5 00 9D 00 7F CA 9F
8008:10 F8 A9 40 85 12 A9 00 50
8010:85 27 A9 10 85 31 A9 FE 17
8018:85 42 A9 FF 85 43 A9 00 2E
8020:85 44 20 8C 88 20 CF FF 26
8028:C9 0D F0 06 9D 81 97 E8 DA
8030:D0 F3 E0 00 D0 01 60 86 84
8038:0A A9 0D 20 D2 FF A9 00 36
8040:20 BD FF A9 0F A2 08 A0 0F
8048:0F 20 BA FF 20 C0 FF A9 DD
8050:00 85 3A A9 01 85 F7 A9 4C
8058:08 85 F8 A9 EE 85 2A A9 04
8060:92 85 2B A0 00 B1 2A 20 B6
8068:E3 8B E6 2A D0 02 E6 2B 45
8070:A5 2B C9 97 D0 04 A5 2A CD
8078:C9 31 90 E9 A9 FF 85 F9 AD
8080:85 06 A9 7D 85 FA 85 07 FC
8088:A9 00 85 2E 8D FF 7D 85 DE
8090:17 85 18 85 1F A9 00 85 FE
8098:38 A9 C0 85 39 A9 02 20 25
80A0:8C 88 20 15 88 20 0D 8C CA
80A8:E0 00 D0 03 4C 30 82 A5 32
80B0:17 18 69 04 85 17 90 02 5C
80B8:E6 18 A0 00 A5 10 91 38 10
80C0:C8 A5 11 91 38 A5 38 18 AB
80C8:69 02 85 38 90 02 E6 39 C6
80D0:20 4D 8C A5 33 C9 3A F0 47
80D8:04 C9 80 90 04 A2 00 86 98
80E0:1F C9 86 D0 02 E6 1F C9 75
80E8:83 D0 5C 20 4D 8C A5 44 99
80F0:D0 0C E6 44 A5 F7 8D E3 8A
80F8:0B A5 F8 8D E7 0B A0 00 8D
```

```
8100:A5 F7 91 42 C8 A5 F8 91 8A
8108:42 A5 33 C9 22 D0 06 20 19
8110:4D 8C 4C 09 81 A5 34 D0 D2
8118:12 A5 33 C9 2C D0 08 A9 EE
8120:00 20 E3 8B 4C 0F 81 C9 CB
8128:3A F0 04 A5 33 D0 F2 A9 CB
8130:00 20 E3 8B A9 00 20 E3 E1
8138:8B A5 F7 85 42 A5 F8 85 E1
8140:43 20 03 8C 4C D3 80 C9 92
8148:22 D0 06 20 CE 81 4C D3 3C
8150:80 C9 8F D0 03 4C A5 80 1A
8158:20 C3 8C 90 4D 20 47 8D FD
8160:A5 1F D0 02 B0 47 20 F4 10
8168:81 A5 1D 18 69 04 85 0F 30
8170:A5 06 38 E5 0F 85 06 B0 78
8178:02 C6 07 A0 00 A5 1D 05 EE
8180:1E 91 06 C8 A5 0C 91 06 CA
8188:C8 A5 F9 38 E5 0C 85 F9 80
8190:91 06 C8 A5 FA E9 00 85 56
8198:FA 91 06 A2 00 C8 BD D1 D8
81A0:97 91 06 E8 E4 1D 90 F5 D5
81A8:B0 03 20 4D 8C A5 33 F0 EF
81B0:03 4C D3 80 4C A5 80 A5 6A
81B8:06 38 E9 04 85 06 B0 02 F1
81C0:C6 07 A0 02 A5 F7 91 06 53
81C8:C8 A5 F8 91 06 60 20 B7 7B
81D0:81 A2 00 20 4D 8C F0 0A C7
81D8:C9 22 F0 06 20 E3 8B E8 58
81E0:D0 F1 84 0F A0 00 A9 30 D2
81E8:91 06 C8 8A 91 06 A4 0F F4
81F0:20 4D 8C 60 A5 1E D0 05 3B
81F8:A9 02 85 0C 60 A9 0A 85 06
8200:0C A5 1F F0 1D 20 74 91 4B
8208:20 D0 8C B0 05 A9 07 4C 17
8210:70 8A 20 DD 8C A5 33 C9 FC
8218:2C F0 F2 20 77 91 A5 13 30
8220:85 0C E6 0C A5 1E C9 20 E1
8228:F0 03 06 0C 60 E6 0C 60 FE
8230:E6 3A 20 CC FF A9 01 20 D0
8238:C3 FF A5 06 38 E5 17 85 41
8240:17 85 2C 85 19 A5 07 E5 63
8248:18 85 18 85 2D 85 1A A5 6F
8250:F7 8D 3D 08 A5 F8 8D 3E 47
8258:08 A9 04 20 8C 88 20 15 2A
8260:88 20 0D 8C E0 00 D0 03 C7
8268:4C 00 83 20 AC 8A A0 00 D6
8270:A5 10 91 2C C8 A5 11 91 D1
8278:2C C8 A5 F7 91 2C C8 A5 6E
8280:F8 91 2C A5 2C 18 69 04 DE
```

8288:85 2C 90 02 E6 2D 20 7C 36
8290:83 20 4D 8C A5 33 C9 3A 99
8298:F0 15 C9 80 90 3A A2 00 4F
82A0:DD 54 8B F0 13 E8 E0 1F 47
82A8:90 F6 A9 07 4C 70 8A 20 B2
82B0:4D 8C A5 33 D0 DE F0 A9 F4
82B8:E0 12 90 05 A9 09 4C 70 8F
82C0:8A 20 4D 8C 8A 0A AA A9 01
82C8:82 48 A9 B1 48 BD DD 82 E8
82D0:48 BD DC 82 48 A5 33 60 CC
82D8:A2 07 D0 E8 49 85 7E 86 7D
82E0:F9 86 C2 83 5E 85 C2 83 27
82E8:AE 85 CF 83 CD 84 8F 84 FC
82F0:07 88 D0 84 49 85 C9 83 75
82F8:C2 87 E0 85 F7 87 4E 85 B5
8300:20 7C 83 A9 60 20 E3 8B 18
8308:A5 F8 48 A5 F7 48 A5 1A C9
8310:C5 18 D0 06 A5 19 C5 17 AE
8318:B0 33 A0 00 B1 19 85 F7 4D
8320:C8 B1 19 85 F8 C8 B1 19 DA
8328:85 13 C8 B1 19 85 14 20 12
8330:93 83 A0 02 B1 2A 20 E3 70
8338:8B C8 B1 2A 20 E3 8B A5 5D
8340:19 18 69 04 85 19 90 02 FA
8348:E6 1A 4C 0E 83 A9 05 20 A0
8350:8C 88 20 CC FF A9 01 20 59
8358:C3 FF A9 0F 20 C3 FF A2 1A
8360:7F BD 00 7F 95 00 CA 10 E0
8368:F8 68 85 2D 68 85 2E A5 E4
8370:F9 8D 1B 08 A5 FA 8D 1F 0F
8378:08 6C 02 A0 A5 2E F0 12 C2
8380:20 CD 8B A0 00 A5 F7 91 9E
8388:13 C8 A5 F8 91 13 C6 2E 24
8390:D0 EA 60 A5 17 85 2A A5 E9
8398:18 85 2B A0 00 B1 2A C5 5D
83A0:13 D0 07 C8 B1 2A C5 14 A8
83A8:F0 18 A5 2A 18 69 04 85 79
83B0:2A 90 02 E6 2B A5 2B C5 AB
83B8:2D D0 04 A5 2A C5 2C 90 B6
83C0:DA 18 60 A9 3A 20 A4 8C 0A
83C8:B0 05 68 68 4C 61 82 60 4A
83D0:20 80 8C 20 C3 8C B0 05 52
83D8:A9 0B 4C 70 8A A9 00 85 88
83E0:FD A9 B2 20 A4 8C B0 05 67
83E8:A9 18 4C 70 8A 20 4D 8C 57
83F0:20 F1 8D A5 33 48 A5 32 C8
83F8:85 41 20 8F 8C 20 47 8D 11
8400:A5 1E D0 04 A9 8D D0 0F F1
8408:C9 10 D0 48 20 B9 8B 20 B7

```

8410:5B 8F 20 CD 8B A9 9D 85 4F
8418:40 A5 3F F0 05 A9 10 4C DC
8420:70 8A 20 60 91 A9 02 20 65
8428:E3 8B A5 40 20 E3 8B 20 86
8430:03 8C 20 60 91 A9 03 20 41
8438:E3 8B E6 13 D0 02 E6 14 94
8440:A5 40 20 E3 8B 20 03 8C DD
8448:A5 FD D0 07 A5 41 85 32 9D
8450:68 85 33 60 A5 3F F0 C5 2D
8458:A9 A2 20 E3 8B E6 13 D0 10
8460:02 E6 14 A5 13 20 E3 8B 6D
8468:A9 A0 20 E3 8B A5 14 20 EB
8470:E3 8B 20 5A 91 A9 D8 20 FC
8478:E3 8B A9 08 20 E3 8B 20 D3
8480:63 91 A5 13 D0 02 C6 14 B5
8488:C6 13 20 03 8C 4C 48 84 98
8490:E6 2E 20 F1 8D A2 00 BD 70
8498:41 85 20 E3 8B E8 E0 09 B0
84A0:D0 F5 A5 F8 20 73 8B A5 5F
84A8:F7 20 73 8B 20 03 8C A9 AC
84B0:A7 20 A4 8C B0 0C A9 89 85
84B8:20 A4 8C B0 0E A9 0B 4C 11
84C0:70 8A 20 4D 8C 20 D0 8C 90
84C8:B0 04 60 20 4D 8C A9 4C 75
84D0:2C A9 20 20 E3 8B 20 CC BA
84D8:85 20 DD 8C 20 93 83 B0 38
84E0:53 A9 00 85 2A A9 C0 85 55
84E8:2B A0 00 B1 2A C5 13 D0 2A
84F0:07 C8 B1 2A C5 14 F0 1C 05
84F8:A5 2A 18 69 02 85 2A 90 04
8500:02 E6 2B A5 2B C5 39 D0 39
8508:04 A5 2A C5 38 90 DA A9 83
8510:0A 4C 70 8A A0 00 A5 19 53
8518:38 E9 04 85 19 B0 02 C6 E8
8520:1A A5 F7 91 19 C8 A5 F8 E9
8528:91 19 C8 A5 13 91 19 C8 8F
8530:A5 14 91 19 C8 B1 2A 20 58
8538:E3 8B C8 B1 2A 20 E3 8B 71
8540:60 A5 03 D0 07 A5 02 D0 F5
8548:03 4C A9 60 4C E3 8B 20 4C
8550:5A 91 A9 85 20 E3 8B A9 CB
8558:0B 20 E3 8B 4C 82 85 C9 67
8560:22 D0 12 20 6F 86 A9 3B 19
8568:20 A4 8C B0 05 A9 14 4C 8C
8570:70 8A 20 4D 8C 20 5A 91 5A
8578:A9 94 20 E3 8B A9 0B 20 F8
8580:E3 8B 20 C3 8C B0 05 A9 7B
8588:0B 4C 70 8A 20 47 8D A9 C5
8590:01 85 3F A5 1E C9 20 F0 09

```

8598:0F C6 3F 20 5A 91 A9 69 9C
85A0:20 E3 8B A9 0A 20 E3 8B E4
85A8:A9 01 85 FD 4C 00 84 20 E4
85B0:5A 91 A9 BB 20 E3 8B A9 8F
85B8:0B 20 E3 8B 20 82 85 A5 42
85C0:33 C9 2C D0 06 20 4D 8C 42
85C8:4C AF 85 60 20 9B 8E B0 D9
85D0:0F 20 C3 8C B0 05 A9 0B A4
85D8:4C 70 8A A9 07 4C 70 8A E6
85E0:60 A5 33 F0 2E C9 3B F0 FA
85E8:37 C9 3A F0 26 C9 A3 F0 E8
85F0:39 C9 A6 F0 45 C9 2C F0 89
85F8:51 20 F1 8D A5 3F D0 0D A4
8600:20 5A 91 A9 D9 20 E3 8B 23
8608:A9 0A 20 E3 8B 20 72 86 F6
8610:4C E1 85 20 5A 91 A9 B3 8E
8618:20 E3 8B A9 0B 4C E3 8B 17
8620:20 4D 8C F0 04 C9 3A D0 BD
8628:B8 60 20 63 86 A9 6D 20 B9
8630:E3 8B A9 0B 20 E3 8B 4C EB
8638:E1 85 20 63 86 A9 80 20 CD
8640:E3 8B A9 0B 20 E3 8B 4C FB
8648:E1 85 20 5A 91 A9 59 20 57
8650:E3 8B A9 0B 20 E3 8B 20 DF
8658:4D 8C F0 CD C9 3A F0 C9 0D
8660:4C E1 85 20 4D 8C 20 F1 8D
8668:8D 20 77 91 4C 5A 91 20 5B
8670:F1 8D 20 5A 91 A9 ED 20 B2
8678:E3 8B A9 0B 4C E3 8B 20 69
8680:80 8C 20 D0 83 20 8F 8C 4A
8688:20 47 8D 20 B9 8B A9 A4 1F
8690:20 A4 8C B0 05 A9 0B 4C A4
8698:70 8A 20 4D 8C 20 CD 86 60
86A0:A5 33 C9 A9 D0 16 20 4D 8D
86A8:8C 20 CD 86 A5 F8 20 80 F7
86B0:8B A5 F7 20 80 8B 20 CD 2E
86B8:8B 4C C3 8B A9 00 20 80 DD
86C0:8B A9 01 20 80 8B A9 00 A5
86C8:20 80 8B D0 DF 20 9B 8E C9
86D0:90 0B 20 DD 8C 20 C3 8B C2
86D8:A9 00 4C 80 8B 20 C3 8C 3D
86E0:B0 05 A9 0B 4C 70 8A 20 C6
86E8:47 8D 20 C3 8B A9 01 D0 13
86F0:E9 A2 08 20 A3 8B CA D0 50
86F8:FA 60 20 C3 8C 90 1C 20 DA
8700:47 8D 20 A3 8B 85 2A AA C5
8708:20 A3 8B 85 2B C5 14 D0 43
8710:04 E4 13 F0 10 20 F1 86 37
8718:4C 02 87 20 A3 8B 85 2A 41

```

8720:20 A3 8B 85 2B 20 D8 8B 09
8728:20 B9 8B A9 02 85 28 85 BD
8730:3D 20 A6 87 20 FF 8E 20 71
8738:AF 87 20 5A 91 A9 58 20 AE
8740:E3 8B A9 0A 20 E3 8B 20 C1
8748:A6 87 20 60 91 A9 02 20 ED
8750:E3 8B 20 63 91 20 03 8C 0E
8758:20 60 91 A9 03 20 E3 8B 48
8760:E6 13 D0 02 E6 14 20 63 0D
8768:91 20 03 8C 20 AF 87 20 60
8770:5A 91 A9 5F 20 E3 8B A9 8D
8778:0A 20 E3 8B 20 60 91 A9 19
8780:02 20 E3 8B A9 F0 20 E3 03
8788:8B A9 03 20 E3 8B A9 4C 17
8790:20 E3 8B 20 CD 8B 20 03 FB
8798:8C A5 33 C9 2C D0 06 20 2B
87A0:4D 8C 4C FA 86 60 A5 2A DD
87A8:85 13 A5 2B 85 14 60 A9 8D
87B0:04 85 28 20 A3 8B 48 20 26
87B8:D8 8B 68 D0 03 4C B7 8E 78
87C0:4C FF 8E 20 F1 8D 20 5A 2A
87C8:91 A9 FD 20 E3 8B A9 0B 78
87D0:20 E3 8B A5 33 C9 2C F0 BE
87D8:05 A9 12 4C 70 8A 20 4D 17
87E0:8C 20 F1 8D A2 00 BD F2 D8
87E8:87 20 E3 8B E8 E0 06 90 60
87F0:F5 60 A0 00 A5 02 91 14 93
87F8:20 F1 8D 20 5A 91 A9 B8 6D
8800:20 E3 8B A9 0B 4C E3 8B 03
8808:20 5A 91 A9 E2 20 E3 8B 77
8810:A9 0B 4C E3 8B A5 0A A2 2A
8818:81 A0 97 20 BD FF A9 01 49
8820:A2 08 A0 08 20 BA FF 20 25
8828:C0 FF 90 15 C9 00 F0 54 81
8830:C9 05 D0 0D A9 15 20 8C C0
8838:88 A9 16 20 8C 88 4C 84 60
8840:88 20 4F 88 A2 01 20 C6 30
8848:FF 20 CF FF 4C CF FF A2 9F
8850:0F 20 C6 FF A2 00 20 CF EE
8858:FF C9 0D F0 06 9D 00 02 35
8860:E8 D0 F3 86 0E AD 00 02 2A
8868:C9 32 B0 01 60 A9 08 20 EA
8870:8C 88 A2 00 BD 00 02 20 50
8878:D2 FF E8 E4 0E 90 F5 A9 A6
8880:0D 20 D2 FF A9 11 20 8C D8
8888:88 4C 52 83 0A AA BD A5 8F
8890:88 85 08 BD A6 88 85 09 8F
8898:A0 00 B1 08 F0 06 20 D2 63
88A0:FF C8 D0 F6 60 D7 88 06 E6

```

88A8:89 19 89 25 89 25 89 31 6D
 88B0:89 49 89 59 89 67 89 7B 18
 88B8:89 90 89 A2 89 A9 89 A9 BD
 88C0:89 B7 89 CD 89 DE 89 EC 5A
 88C8:89 02 8A 10 8A 24 8A 36 A1
 88D0:8A 49 8A 5D 8A 5D 8A 0D 8C
 88D8:0D 0D 53 50 52 49 4E 54 CB
 88E0:20 20 20 20 20 20 20 0D DE
 88E8:42 41 53 49 43 20 43 4F DA
 88F0:4D 50 49 4C 45 52 0D 0D 45
 88F8:53 4F 55 52 43 45 20 46 0D
 8900:49 4C 45 3A 20 00 0D 2A 5C
 8908:2A 2A 20 42 41 44 20 46 84
 8910:49 4C 45 20 2A 2A 2A 0D E0
 8918:00 50 41 53 53 20 31 2E 48
 8920:2E 2E 0D 0D 00 50 41 53 5F
 8928:53 20 32 2E 2E 2E 0D 0D 67
 8930:00 0D 43 4F 4D 50 49 4C 6E
 8938:41 54 49 4F 4E 20 46 49 E7
 8940:4E 49 53 48 45 44 2E 0D 60
 8948:00 20 20 20 2A 2A 2A 20 D7
 8950:45 52 52 4F 52 3A 20 20 B5
 8958:00 4E 4F 54 20 53 55 50 77
 8960:50 4F 52 54 45 44 00 2A 64
 8968:2A 2A 20 44 49 53 4B 20 B1
 8970:45 52 52 4F 52 20 2A 2A 8B
 8978:2A 0D 00 49 4C 4C 45 47 DD
 8980:41 4C 20 46 55 4E 43 54 6E
 8988:49 4F 4E 20 55 53 45 00 62
 8990:4E 4F 4E 2D 45 58 49 53 AC
 8998:54 49 4E 47 20 4C 49 4E 79
 89A0:45 00 53 59 4E 54 41 58 F4
 89A8:00 54 4F 4F 20 4D 41 4E B6
 89B0:59 20 46 4F 52 53 00 49 5F
 89B8:4C 4C 45 47 41 4C 20 53 F0
 89C0:54 41 54 45 4D 45 4E 54 9D
 89C8:20 55 53 45 00 4E 45 58 1C
 89D0:54 20 57 49 54 48 4F 55 4D
 89D8:54 20 46 4F 52 00 54 59 70
 89E0:50 45 20 4D 49 53 4D 41 B9
 89E8:54 43 48 00 0D 43 4F 4D 61
 89F0:50 49 4C 41 54 49 4F 4E D0
 89F8:20 41 42 4F 52 54 45 44 5C
 8A00:0D 00 4D 49 53 53 49 4E A2
 8A08:47 20 43 4F 4D 4D 41 00 48
 8A10:4D 49 53 53 49 4E 47 20 EF
 8A18:50 41 52 45 4E 54 48 45 DD
 8A20:53 49 53 00 4D 49 53 53 25
 8A28:49 4E 47 20 53 45 4D 49 F3


```

8A30:43 4F 4C 4F 4E 00 2A 2A 2A
8A38:2A 20 49 2F 4F 20 45 52 5E
8A40:52 4F 52 20 2A 2A 2A 0D F9
8A48:00 44 45 56 49 43 45 20 7E
8A50:4E 4F 54 20 50 52 45 53 96
8A58:45 4E 54 0D 00 4D 49 53 1A
8A60:53 49 4E 47 20 45 51 55 BD
8A68:41 4C 20 53 49 47 4E 00 6E
8A70:A2 FA 9A 48 A9 40 85 12 D8
8A78:A9 10 85 31 A5 3A D0 03 E4
8A80:20 AC 8A A9 06 20 8C 88 0F
8A88:68 20 8C 88 A9 0D 20 D2 88
8A90:FF A9 3A 20 A4 8C B0 0A 1C
8A98:A5 3A D0 03 4C A5 80 4C 9F
8AA0:61 82 A5 3A D0 03 4C D3 5E
8AA8:80 4C 94 82 A0 00 84 20 F9
8AB0:A5 10 85 2A A5 11 85 2B 97
8AB8:A2 08 A0 00 A5 2B DD 24 EE
8AC0:8B D0 05 A5 2A DD 23 8B 65
8AC8:90 12 A5 2A 38 FD 23 8B 8D
8AD0:85 2A A5 2B FD 24 8B 85 B7
8AD8:2B C8 D0 E0 E0 00 F0 08 CE
8AE0:C0 00 D0 04 A5 20 F0 08 48
8AE8:98 09 30 20 D2 FF E6 20 19
8AF0:CA CA 10 C6 A9 20 20 D2 6D
8AF8:FF A0 00 84 34 B9 00 02 09
8B00:10 0A A6 34 D0 06 20 2D C5
8B08:8B 4C 19 8B C9 22 D0 06 52
8B10:45 34 85 34 A9 22 20 D2 B3
8B18:FF C8 C4 0E 90 DF A9 0D 3F
8B20:4C D2 FF 01 00 0A 00 64 AE
8B28:00 E8 03 10 27 38 E9 7F 48
8B30:AA 84 0F A0 FF CA F0 08 BE
8B38:C8 B9 9E A0 10 FA 30 F5 C2
8B40:C8 B9 9E A0 30 06 20 D2 B4
8B48:FF 4C 40 8B 29 7F 20 D2 8D
8B50:FF A4 0F 60 80 81 82 83 0B
8B58:85 86 87 88 89 8B 8C 8D 6E
8B60:8E 8F 97 99 9E A1 B4 B6 CA
8B68:B9 BB C2 C3 C4 C5 C6 C7 72
8B70:C8 C9 CA 84 0F A4 12 88 B7
8B78:99 00 7E 84 12 A4 0F 60 16
8B80:84 0F A4 27 88 C0 40 B0 1D
8B88:05 A9 0D 4C 70 8A 99 00 D3
8B90:7E 84 27 A4 0F 60 86 0F 4D
8B98:A6 12 BD 00 7E E8 86 12 F5
8BA0:A6 0F 60 86 0F A6 27 BD 62
8BA8:00 7E E8 E0 01 D0 05 A9 89
8BB0:0F 4C 70 8A 86 27 A6 0F 46

```

```

8BB8:60 A5 14 20 73 8B A5 13 16
8BC0:4C 73 8B A5 14 20 80 8B 54
8BC8:A5 13 4C 80 8B 20 96 8B 9E
8BD0:85 13 20 96 8B 85 14 60 D7
8BD8:20 A3 8B 85 13 20 A3 8B 9E
8BE0:85 14 60 84 0F A0 00 91 A0
8BE8:F7 E6 F7 D0 02 E6 F8 A5 05
8BF0:F8 C9 C0 D0 04 A5 F7 C9 8C
8BF8:00 B0 03 A4 0F 60 A9 0D 41
8C00:4C 70 8A A5 13 20 E3 8B 73
8C08:A5 14 4C E3 8B A2 01 20 C9
8C10:C6 FF 20 CF FF C9 00 D0 85
8C18:03 AA F0 20 20 CF FF 20 DD
8C20:CF FF 85 10 20 CF FF 85 98
8C28:11 A2 00 20 CF FF C9 00 86
8C30:F0 0A 9D 00 02 E8 E0 50 BD
8C38:B0 0B 90 EF 86 0E A9 00 3D
8C40:85 32 85 34 60 A9 01 20 68
8C48:8C 88 4C 84 88 84 26 20 5E
8C50:E1 FF D0 03 4C 84 88 A4 CE
8C58:32 C4 0E B0 1D B9 00 02 5A
8C60:85 33 C8 C9 22 D0 04 45 60
8C68:34 85 34 C9 20 D0 04 A5 12
8C70:34 F0 E6 84 32 A4 26 A5 1B
8C78:33 60 A9 00 85 33 F0 F3 47
8C80:84 26 A4 31 88 A5 32 99 E5
8C88:00 7E 84 31 A4 26 60 84 E7
8C90:26 A4 31 B9 00 7E 85 32 DE
8C98:C8 84 31 A8 B9 FF 01 85 3D
8CA0:33 A4 26 60 85 35 20 80 09
8CA8:8C A5 33 F0 09 C5 35 F0 A1
8CB0:0A 20 4D 8C D0 F7 20 8F 7F
8CB8:8C 18 60 A5 34 D0 F2 E6 36
8CC0:31 38 60 A5 33 C9 5B B0 0F
8CC8:05 C9 41 90 01 60 18 60 22
8CD0:A5 33 C9 3A B0 05 C9 30 C3
8CD8:90 01 60 18 60 A9 00 85 37
8CE0:36 85 13 85 14 85 15 A5 B7
8CE8:33 C9 AB D0 05 E6 36 20 E0
8CF0:4D 8C A5 32 85 37 20 D0 C5
8CF8:8C 90 2F 29 CF 85 16 A5 5B
8D00:14 85 15 A5 13 0A 26 15 A5
8D08:0A 26 15 65 13 85 13 A5 25
8D10:15 65 14 85 14 06 13 26 EE
8D18:14 A5 13 65 16 85 13 90 DC
8D20:02 E6 14 20 4D 8C D0 CE 87
8D28:F0 0B A5 32 C5 37 D0 05 08
8D30:A9 07 4C 70 8A A5 36 F0 BA
8D38:0D A9 00 38 E5 13 85 13 61

```

```

8D40:A9 00 E5 14 85 14 60 A2 0E
8D48:00 86 1E 20 C3 8C B0 05 81
8D50:20 D0 8C 90 09 9D D1 97 44
8D58:E8 20 4D 8C D0 ED C9 25 59
8D60:D0 05 20 4D 8C F0 2A C9 44
8D68:24 D0 09 A9 20 85 1E 20 F8
8D70:4D 8C F0 1D C9 28 D0 19 EE
8D78:A5 1F D0 0B A9 20 C5 1E 70
8D80:D0 0B A9 07 4C 70 8A A9 4F
8D88:20 C5 1E F0 04 A9 10 85 64
8D90:1E E0 10 90 02 A2 0F 86 3D
8D98:1D A5 06 85 22 A5 07 85 FF
8DA0:23 20 E1 FF D0 03 4C 84 41
8DA8:88 A0 00 B1 22 F0 40 29 C9
8DB0:0F 85 20 C5 1D D0 28 B1 43
8DB8:22 29 F0 C5 1E D0 20 C8 E6
8DC0:B1 22 85 0C C8 B1 22 85 85
8DC8:13 C8 B1 22 85 14 A2 00 B9
8DD0:C8 B1 22 DD D1 97 D0 07 74
8DD8:E8 E4 1D 90 F3 B0 11 A5 78
8DE0:20 18 69 04 65 22 85 22 60
8DE8:90 02 E6 23 4C A1 8D 18 F7
8DF0:60 20 80 8C A9 00 85 3F B4
8DF8:20 9B 8E B0 25 C9 22 F0 6D
8E00:19 C9 C4 F0 15 C9 C7 90 B3
8E08:04 C9 CB 90 0D C9 80 B0 5D
8E10:11 20 47 8D A5 1E C9 20 D8
8E18:D0 08 20 8F 8C E6 3F 4C 67
8E20:86 91 20 8F 8C A9 02 85 76
8E28:28 20 A6 8E A5 33 F0 4D 48
8E30:C9 AE D0 05 A9 07 4C 70 BA
8E38:8A C9 AA 90 40 C9 B4 B0 AE
8E40:3C E9 A9 85 FE A4 32 B9 2C
8E48:00 02 C9 B1 90 0F C9 B4 43
8E50:B0 0B 38 E9 AF 18 65 FE D5
8E58:85 FE 20 4D 8C A9 04 85 69
8E60:28 20 4D 8C 20 A6 8E A5 6A
8E68:FE 0A AA 20 5A 91 BD 7E F1
8E70:8E 20 E3 8B BD 7F 8E 20 3B
8E78:E3 8B 4C 2C 8E 60 3D 09 30
8E80:4B 09 59 09 84 09 3B 0A 0A
8E88:D3 09 E0 09 25 0A 1B 0A 10
8E90:31 0A 45 0A 3B 0A 4F 0A BC
8E98:A6 3E 60 20 D0 8C B0 05 C5
8EA0:C9 AB F0 01 18 60 A9 01 52
8EA8:2C A9 00 85 3D 86 3E 20 3F
8EB0:9B 8E 90 30 20 DD 8C 20 06
8EB8:5D 91 A5 13 20 E3 8B A9 20
8EC0:85 20 E3 8B A5 28 20 E3 CF

```

8EC8:8B A5 3D F0 CB 20 5D 91 F6
8ED0:A5 14 20 E3 8B A9 85 20 36
8ED8:E3 8B A5 28 18 69 01 A6 11
8EE0:3E 4C E3 8B C9 80 90 06 DC
8EE8:20 A7 8F 4C 77 91 20 C3 BC
8EF0:8C B0 05 A9 0B 4C 70 8A B0
8EF8:20 47 8D A5 1E D0 27 A9 30
8F00:AD 20 E3 8B 20 03 8C A9 03
8F08:85 20 E3 8B A5 28 20 E3 19
8F10:8B A5 3D F0 B6 A9 AD 20 EC
8F18:E3 8B E6 13 D0 02 E6 14 8A
8F20:20 03 8C 4C D5 8E C9 20 03
8F28:D0 05 A9 10 4C 70 8A 20 80
8F30:B9 8B 20 5B 8F 20 CD 8B EC
8F38:A9 BD 20 E3 8B 20 03 8C 4D
8F40:A9 85 20 E3 8B A5 28 20 3B
8F48:E3 8B A9 BD 20 E3 8B E6 DB
8F50:13 D0 02 E6 14 20 03 8C 8F
8F58:4C D5 8E 20 74 91 20 9B AC
8F60:8E 90 18 20 DD 8C 20 5D AE
8F68:91 A5 13 20 E3 8B A9 0A C8
8F70:20 E3 8B 20 77 91 A9 AA 0C
8F78:4C E3 8B 20 C3 8C B0 05 E0
8F80:A9 0B 4C 70 8A 20 47 8D B8
8F88:A5 1E C9 00 F0 0E C9 10 9E
8F90:D0 05 A9 07 4C 70 8A A9 E1
8F98:10 4C 70 8A A9 AD 20 E3 B1
8FA0:8B 20 03 8C 4C 6E 8F A2 94
8FA8:00 DD 54 8B F0 0A E8 E0 E4
8FB0:1F 90 F6 A9 07 4C 70 8A D1
8FB8:E0 12 B0 05 A9 0E 4C 70 C1
8FC0:8A 8A E9 12 0A AA BD D2 6F
8FC8:8F 48 BD D1 8F 48 4C 74 41
8FD0:91 EA 8F EA 8F ED 8F ED 55
8FD8:8F EA 8F 1B 90 B7 90 3B DD
8FE0:90 59 90 7F 90 D9 90 FA B0
8FE8:90 20 91 20 A6 8E 20 5A 96
8FF0:91 A5 28 C9 02 D0 0C BD 0D
8FF8:12 90 20 E3 8B BD 13 90 91
9000:4C E3 8B BD 12 90 18 69 F9
9008:03 20 E3 8B BD 13 90 4C 8F
9010:E3 8B 86 08 4D 08 68 08 BA
9018:76 08 5A 08 20 47 8D A9 25
9020:AD 20 E3 8B 20 03 8C A9 25
9028:85 20 E3 8B A5 28 20 E3 3B
9030:8B 20 5D 91 A9 00 20 E3 55
9038:8B 4C D5 8E A9 00 85 29 57
9040:85 FC 20 FF 91 20 5A 91 BA
9048:A6 28 BD 54 90 20 E3 8B 1C

```

9050:BD 55 90 4C E3 8B 69 0A A6
9058:6C 0A A9 00 85 29 85 FC 40
9060:20 FF 91 20 5A 91 A5 28 52
9068:C9 02 D0 0A A9 2A 20 E3 C3
9070:8B A9 09 4C E3 8B A9 2D 75
9078:20 E3 8B A9 09 4C E3 8B 7B
9080:A9 02 85 28 20 A9 8E A9 98
9088:A5 20 E3 8B A5 28 20 E3 AB
9090:8B A9 8D 20 E3 8B A9 00 36
9098:20 E3 8B A9 7E 18 65 29 17
90A0:20 E3 8B 20 5D 91 A9 01 C3
90A8:20 E3 8B A9 85 20 E3 8B DE
90B0:A9 08 18 65 29 4C E3 8B CF
90B8:A9 02 85 28 20 A6 8E 20 3B
90C0:5A 91 A5 29 D0 0A A9 E6 A3
90C8:20 E3 8B A9 0A 4C E3 8B D3
90D0:A9 E9 20 E3 8B A9 0A 4C E6
90D8:E3 8B A9 02 85 28 20 80 B1
90E0:8C 20 68 91 20 A9 8E 20 5B
90E8:18 91 A5 32 85 41 20 8F 53
90F0:8C A9 04 85 FC 20 FF 91 95
90F8:4C 44 91 A9 02 85 28 20 B4
9100:80 8C 20 68 91 20 A9 8E FF
9108:20 18 91 20 8F 8C A9 08 7F
9110:85 FC 20 FF 91 4C 44 91 11
9118:A5 33 85 72 A5 32 85 41 F4
9120:60 A9 02 85 28 A9 0C 85 FB
9128:FC 20 80 8C 20 68 91 20 90
9130:A9 8E 20 4D 91 20 68 91 14
9138:20 A9 8E 20 18 91 20 8F 80
9140:8C 20 FF 91 A5 41 85 32 3A
9148:A5 72 85 33 60 20 5A 91 88
9150:A9 FD 20 E3 8B A9 0B 4C 6F
9158:E3 8B A9 20 2C A9 A9 2C 0F
9160:A9 A5 2C A9 8D 4C E3 8B D2
9168:A9 2C 20 A4 8C B0 14 A9 B2
9170:12 4C 70 8A A9 28 2C A9 56
9178:29 20 A4 8C B0 05 A9 13 95
9180:4C 70 8A 4C 4D 8C A9 00 EB
9188:85 FC 85 29 20 FF 91 A5 BA
9190:33 F0 5D C9 AC 90 09 C9 55
9198:B1 B0 05 A9 07 4C 70 8A D0
91A0:C9 AA 90 4C C9 B4 B0 48 F4
91A8:E9 A9 F0 07 A2 00 86 3F 1B
91B0:38 E9 06 85 FE A4 32 B9 2C
91B8:00 02 C9 B1 90 0E C9 B4 B5
91C0:B0 0A 38 E9 B0 65 FE 85 03
91C8:FE 20 4D 8C A9 01 85 29 6B
91D0:A9 00 85 FC 20 4D 8C 20 B8

```

```
91D8:FF 91 A5 FE 0A AA 20 5A 9A
91E0:91 BD F1 91 20 E3 8B BD F8
91E8:F2 91 20 E3 8B 4C 8F 91 6A
91F0:60 EC 08 22 0A 18 0A 2E 95
91F8:0A 42 0A 38 0A 4C 0A A5 B1
9200:33 C9 22 D0 22 20 59 92 59
9208:20 5D 91 A5 1D 20 E3 8B DD
9210:A9 A2 20 E3 8B A5 13 20 2E
9218:E3 8B A9 A0 20 E3 8B A5 9E
9220:14 20 E3 8B 4C BD 92 C9 D4
9228:80 90 06 20 A7 8F 4C 77 FF
9230:91 20 C3 8C B0 05 A9 0B 5F
9238:4C 70 8A 20 47 8D A5 1E CC
9240:C9 20 F0 05 A9 10 4C 70 57
9248:8A A9 AD 20 E3 8B 20 03 65
9250:8C E6 13 D0 02 E6 14 D0 89
9258:B7 20 4D 8C C9 22 F0 04 90
9260:A9 22 85 34 A2 00 A5 33 6A
9268:F0 0D C9 22 F0 09 9D D1 5D
9270:97 20 4D 8C E8 D0 EF 86 CC
9278:1D 20 4D 8C A5 06 85 22 19
9280:A5 07 85 23 A0 00 B1 22 A7
9288:C9 30 D0 21 C8 B1 22 C5 E1
9290:1D D0 18 C8 B1 22 85 13 3C
9298:C8 B1 22 85 14 A0 FF C8 17
92A0:C4 1D F0 18 B1 13 D9 D1 6E
92A8:97 F0 F4 A9 30 29 0F 18 6B
92B0:69 04 65 22 85 22 90 CC FC
92B8:E6 23 D0 C8 60 20 5A 91 8A
92C0:A6 FC A5 29 D0 0C BD DE D0
92C8:92 20 E3 8B BD DF 92 4C 53
92D0:E3 8B BD E0 92 20 E3 8B F8
92D8:BD E1 92 4C E3 8B A8 08 13
92E0:C0 08 06 0C 0B 0C 10 0C 9E
92E8:16 0C 2B 0C 31 0C 0B 08 1A
92F0:0A 00 9E 32 30 36 31 00 CE
92F8:00 00 A9 FF 8D 0E D4 8D 2F
9300:0F D4 A9 80 8D 12 D4 A9 29
9308:00 85 02 A9 00 85 03 A0 28
9310:00 98 91 02 E6 02 D0 02 92
9318:E6 03 A5 03 C9 7E D0 04 46
9320:A5 02 C9 00 90 E9 20 E2 23
9328:0B 4C FF FF A9 00 38 F5 9B
9330:02 95 02 A9 00 F5 03 95 0C
9338:03 60 A2 00 2C A2 02 B5 F2
9340:03 10 03 20 3F 08 60 A2 CC
9348:00 2C A2 02 A1 02 95 02 31
9350:A9 00 95 03 60 A2 00 2C E8
9358:A2 02 A9 00 95 03 A5 D3 5E
```

```

9360:95 02 60 A2 00 2C A2 02 01
9368:AD 1B D4 95 02 AD 1B D4 F2
9370:95 03 60 A2 00 2C A2 02 51
9378:B5 03 D0 05 B5 02 D0 01 FD
9380:60 B5 03 30 09 A9 00 95 2D
9388:03 A9 01 95 02 60 A9 FF F9
9390:95 02 95 03 60 86 06 84 93
9398:07 8D B2 08 A0 00 C0 00 04
93A0:F0 08 B1 06 99 00 7E C8 6B
93A8:D0 F4 84 08 60 86 06 84 34
93B0:07 8D CA 08 A0 00 C0 00 1F
93B8:F0 08 B1 06 99 00 7F C8 85
93C0:D0 F4 84 09 60 86 06 84 5C
93C8:07 A0 00 C4 08 F0 08 B9 B5
93D0:00 7E 91 06 C8 D0 F4 98 36
93D8:60 A2 00 A4 08 E4 09 F0 F9
93E0:0A BD 00 7F 99 00 7E E8 27
93E8:C8 D0 F2 84 08 60 A9 00 64
93F0:85 03 85 05 A5 08 85 02 F6
93F8:A5 09 85 04 A2 FF E8 E4 F1
9400:08 F0 13 E4 09 F0 0F BD 02
9408:00 7E DD 00 7F F0 EF 85 B1
9410:02 BD 00 7F 85 04 60 A2 41
9418:00 2C A2 02 A9 00 95 03 3C
9420:A5 08 F0 03 AD 00 7E 95 6C
9428:02 60 A5 04 18 65 02 85 3F
9430:02 A5 05 65 03 85 03 60 4F
9438:A5 02 38 E5 04 85 02 A5 F9
9440:03 E5 05 85 03 60 A9 00 4A
9448:85 0A 85 0B A2 0F 06 0A 7F
9450:26 0B 06 02 26 03 90 0D 9B
9458:A5 0A 18 65 04 85 0A A5 20
9460:0B 65 05 85 0B CA 10 E6 EB
9468:A5 0A 85 02 A5 0B 85 03 1F
9470:60 A2 00 86 0A 86 0B A0 FB
9478:00 A5 03 10 04 C8 20 3F 2F
9480:08 A2 02 A5 05 10 04 88 E9
9488:20 3F 08 A2 0F 06 02 26 77
9490:03 26 0A 26 0B A5 0B C5 33
9498:05 D0 04 A5 0A C5 04 90 53
94A0:12 A5 0A E5 04 85 0A A5 CB
94A8:0B E5 05 85 0B E6 02 D0 92
94B0:02 E6 03 CA 10 D7 C0 00 03
94B8:F0 05 A2 00 20 3F 08 60 5E
94C0:A5 02 25 04 85 02 A5 03 A4
94C8:25 05 85 03 60 A5 02 05 49
94D0:04 85 02 A5 03 05 05 85 B3
94D8:03 60 A5 03 30 11 A5 05 96
94E0:30 0B A5 03 C5 05 D0 04 B1

```

94E8:A5 02 C5 04 60 38 60 A5 A8
94F0:05 30 EF A9 01 18 60 A9 14
94F8:FF 85 02 85 03 60 A9 00 09
9500:85 02 85 03 60 20 01 09 DD
9508:20 ED 09 F0 EA D0 EF 20 89
9510:01 09 20 ED 09 F0 E7 90 4D
9518:E5 B0 DC 20 01 09 20 ED 5A
9520:09 B0 DB 90 D2 20 01 09 A2
9528:20 ED 09 F0 D1 D0 C8 20 92
9530:01 09 20 ED 09 90 C7 B0 CB
9538:BE 20 01 09 20 ED 09 F0 37
9540:B6 90 B4 B0 B9 A5 05 85 80
9548:17 4C 3D 09 A5 17 30 03 37
9550:4C 4F 0A 4C 45 0A A2 00 13
9558:2C A2 02 A5 08 85 0F E0 32
9560:02 D0 04 A5 09 85 0F A0 B8
9568:00 98 95 02 95 03 85 0C 5C
9570:85 0E C4 0F F0 3E B9 00 5F
9578:7E C9 2D D0 04 E6 0E D0 B0
9580:30 C9 30 90 2F C9 3A B0 0B
9588:2B E9 2F 85 0D B5 03 85 CC
9590:0C B5 02 0A 26 0C 0A 26 AB
9598:0C 75 02 95 02 A5 0C 75 F4
95A0:03 95 03 16 02 36 03 B5 19
95A8:02 65 0D 95 02 90 02 F6 76
95B0:03 C8 D0 BE A5 0E F0 03 DF
95B8:20 3F 08 60 01 00 0A 00 E6
95C0:64 00 E8 03 10 27 20 E6 AF
95C8:0A A6 08 A9 20 9D 00 7E 34
95D0:E6 08 60 A9 7E 2C A9 7F 8F
95D8:8D 47 0B 38 E9 76 8D 50 16
95E0:0B A0 00 84 0C A9 20 8D D6
95E8:00 7E A5 03 10 0A A9 2D C1
95F0:8D 00 7E A2 00 20 3F 08 E3
95F8:C8 A2 08 A9 00 85 0D A5 A2
9600:03 DD D0 0A D0 05 A5 02 C8
9608:DD CF 0A 90 13 A5 02 38 CD
9610:FD CF 0A 85 02 A5 03 FD 74
9618:D0 0A 85 03 E6 0D D0 DF FD
9620:E0 00 F0 0A A5 0D C9 00 71
9628:D0 04 A5 0C F0 0A A5 0D 3C
9630:09 30 99 00 7E C8 E6 0C 12
9638:CA CA 10 BF 84 00 60 46 A6
9640:3C 32 28 1E 14 0A A2 06 13
9648:A5 D3 DD 52 0B 90 06 CA 8F
9650:10 F6 4C B3 0B BD 52 0B 07
9658:85 02 A5 02 38 E5 D3 AA 49
9660:E0 00 F0 08 A9 20 20 D2 7D
9668:FF CA 10 F4 60 A5 02 4C 83


```

9670:5F 96 A9 00 85 08 20 E4 99
9678:FF F0 05 8D 00 7E E6 08 2B
9680:60 A5 B8 F0 0A A9 3F 20 03
9688:D2 FF A9 20 20 D2 FF A2 45
9690:00 20 CF FF C9 0D F0 06 2A
9698:9D 00 7E E8 D0 F3 86 08 5E
96A0:A9 0D 4C D2 FF 6C 02 00 52
96A8:A4 12 A2 00 B1 10 F0 07 B7
96B0:9D 00 7E C8 E8 D0 F5 86 06
96B8:08 C8 B1 10 D0 0E C8 B1 55
96C0:10 48 C8 B1 10 85 11 68 5D
96C8:85 10 A0 00 84 12 60 A9 A7
96D0:00 85 10 A9 00 85 11 4C 80
96D8:DD 0B A2 00 E4 08 F0 09 3E
96E0:BD 00 7E 20 D2 FF E8 D0 F7
96E8:F3 60 A5 02 85 14 A5 03 C7
96F0:85 15 60 A5 02 4C A8 08 27
96F8:A5 02 4C C0 08 20 1C 0C 14
9700:4C A8 08 20 1C 0C 4C C0 EC
9708:08 86 14 38 E5 02 18 65 AF
9710:14 AA 90 01 C8 A5 02 60 57
9718:20 37 0C 4C A8 08 20 37 48
9720:0C 4C C0 08 C6 14 8A 18 B4
9728:65 14 AA 90 01 C8 A5 02 E5
9730:60 00 C7 00 00 00 00 88

```

Program 2. 64 Doodler Demo

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

```

AC 100 POKE53280,0:POKE53281,2:PRINT"{CLR}{3 DOWN}
      {WHT}"TAB(15)"64 DOODLER"
KB 110 PRINT"{2 DOWN}{BLK}"TAB(8)"PLUG JOYSTICK IN
      TO PORT 2"
EP 120 PRINT"{DOWN}"TAB(9)"USE IT TO DRAW PICTURES
      "
CR 130 PRINT"{DOWN}"TAB(10)"TRIGGER CLEARS SCREEN"
      :PRINT"{4 DOWN}{WHT}"TAB(14)"PRESS ANY KEY"
ER 140 GETA$:IFA$=" "THEN140
EG 150 PRINT"{CLR}":FORI=0TO7:READP(I):NEXT:DATA1,
      2,4,8,16,32,64,128
EH 160 FORI=0TO10:READDX(I),DY(I):NEXT
RR 170 DATA0,0,0,-1,0,1,0,0,-1,0,-1,-1,-1,1,0,0,1,
      0,1,-1,1,1
GF 180 POKE53272,PEEK(53272)OR8:POKE53265,PEEK(532
      65)OR32
PH 190 GOSUB300:FORI=1024TO2023:POKEI,2:NEXT
CA 200 X=160:Y=100
JS 210 GOSUB280:J=PEEK(56320):K=JAND16:IFK=0THENGOSUB300

```

```
CB 220 J=JAND15:J=15-J:X=X+DX(J):Y=Y+DY(J)
SS 230 IFX>31.9THENX=0
AX 240 IFX<0THENX=31.9
CA 250 IFY>1.99THENY=0
QE 260 IFY<0THENY=1.99
BC 270 GOTO210
ME 280 RO%=Y/8:CH%=X/8:LI=YAND7:BI=XAND7:BI=7-BI:BI=RO%*320:B2=CH%*8
DD 290 BY=81.92+B1+B2+LI:POKEBY,PEEK(BY)ORP(BI):RETURN
GF 300 FORI=81.92TO16383:POKEI,0:NEXT:RETURN
```

The Fast Assembler

Yves Han

Here's a truly amazing machine language assembler. "Fast Assembler" supports multiple statement lines, labels, and macro-like "include" files. It can assemble to memory or to disk. Written very compactly, it occupies only about 2600 bytes, leaving the rest of memory for your source code. It also adds to the BASIC editor several new features useful to both BASIC and machine language programmers.

Symbolic label-based assemblers are the most convenient way to write machine language (ML) programs. The instructions are entered as *source code* and later assembled into object code (the actual ML program—the numbers in memory). And rather than always having to specify memory addresses, you can name routines with meaningful labels. It's as if you could enter GOSUB JOYSTICK in BASIC.

Saving Memory by Using the BASIC Editor

You write the source code for your ML programs for "The Fast Assembler" (FA) with the BASIC editor. You save the source code to tape or disk as you would a BASIC program, and list it to a printer exactly as you would list BASIC.

The FA is an extension of the BASIC interpreter especially designed for writing programs in machine language. Writing it as a BASIC extension kept the program short (under 2600 bytes) because many subroutines of the BASIC interpreter could be used. Some modifications have been made to BASIC to make writing programs easier. To do this, BASIC ROM was copied into underlying RAM.

Even if you don't write programs in machine language, you can still use the assembler because of the new features added to BASIC and the extra BASIC commands it adds. The assembler will execute a BASIC program just as normal BASIC would.

Typing Fast

You'll need a copy of "MLX" (found in Appendix D) to type in Fast Assembler. First enter MLX and save a copy or two. When you are ready to enter the data for Fast Assembler, run MLX and answer the starting and ending address prompts as follows:

Starting Address: 0801

Ending Address: 1220

After entering all the data for Fast Assembler, save a copy on tape or disk. To start up FA, first load it as if it were a BASIC program (don't use an extra ,1 in the LOAD command; just type **LOAD "filename",8** for disk or **LOAD"filename"** for tape). Then type **RUN**. The enabling SYS is built into the first line of the program. The screen will clear, and a message will appear at the top of the screen, indicating FA has been enabled. You can now start programming—in BASIC or machine language.

BASIC Modifications and Enhancements

The following changes have been made to the BASIC interpreter:

- **Structured listings.** Spaces between the line number and the first character on the line are not deleted. This makes it possible to indent lines and make listings easier to read.
- **List pause.** You can freeze a listing by holding down the SHIFT key or pressing SHIFT-LOCK. Listing can be continued by releasing the SHIFT key.
- **ASCII translations and hexadecimal/binary numbers.** In arithmetic expressions, you can use hexadecimal and binary numbers. Hexadecimal numbers should be preceded by \$ and binary numbers by %. You can also use a character preceded by a single quote ('A is the same as ASC("A")). You can also use this to find the value of a BASIC token. For example, **PRINT 'END** will print the value 128, which is the BASIC code for END. If you put a space between the quote and the character, the ASCII value of the space will be used instead of the character value.
- **Variable and function names.** The rules for variable and function names have been changed a bit. Instead of the first two, the first *eight* characters are recognized. FA recognizes NUMBER1 and NUMBER2 as separate variables, while ordinary BASIC would consider them the same variable (NU).

Variables may contain but not be equal to BASIC/assembler commands or mnemonics: LAND is a legitimate variable name, even though it contains the keyword AND. But variable labels starting with *TI* or *ST* (reserved keywords) are not automatically set to zero the first time you use them. An exception to the eight character names is that only the first two characters of array variables are significant.

- **Keywords.** Because variable and function names may contain keywords, FA has to be able to decide whether a keyword is a keyword or part of a variable or function name. The assembler recognizes a keyword if it's followed by a space or non-alphabetic character. For example, in PRINT "OK" the keyword PRINT will be recognized as a PRINT command, but in A\$="OK":PRINTA\$, the keyword PRINT is interpreted as part of the variable name PRINTA\$. You would have to insert a space (PRINT A\$) if you wanted to print the variable A\$. In keywords like PRINT# or GET#, you must remember to include a space between the keyword and the logical file number value. For example, you must use PRINT# 1; a statement like PRINT#1 will cause a syntax error.
- **REM and DATA.** Capital letters in REM and DATA lines are listed as capital letters and not as tokenized BASIC keywords. For example, **10 rem AB** lists as it's entered and not as **10 rem atnpeek** as normal BASIC would do.

New BASIC Commands

AUTO step value

This command turns on automatic line numbering and defines the step value between the line numbers. To enter AUTO mode, type AUTO followed by the step value, and press RETURN. Then enter a line with a line number. The next line number prints automatically. To leave auto mode, move the cursor to an empty line and press RETURN. To turn automatic line numbering off altogether, enter AUTO only.

You can also use this command to delete part of a program. Turn automatic line numbering on with a step value of one. Type the number of the first line you want to delete and press RETURN. Keep pressing RETURN until you've reached the end of the section you want to delete. Instead of pressing RETURN again and again, you can enter **POKE 650,128** and hold RETURN down until you've reached the last line to be erased.

OLD

If you accidentally type **NEW**, you can restore your program with this command. It can also be used if you've installed a reset button. If you've assembled a program and are testing it, sometimes your computer locks up. Use the reset button and then enter **SYS 4408** to restart the assembler and type **OLD** to restore the source program. If your program has not destroyed the assembler or the source program, everything will be there.

Semicolon (;)

This has the same function as the **REM** statement. It need not be separated with a colon from the preceding command. For example:

```
10 X=0:REM SET X TO ZERO
```

is the same as

```
10 X=0;SET X TO ZERO
```

The semicolon in the commands **PRINT** and **INPUT** is not treated as a **REM** statement but as a separator.

Using Labels as Variables and Addresses

Label names follow the same rules as variable names. They can be used in arithmetic expressions like normal variables. You can define a label in two ways:

You can place the label name just before the command to which you want to refer. If more commands are on the same line, you must separate the label from the commands with a colon.

Or you can label the current program counter:

LABELNAME=*. The asterisk (*) is a special variable which gives the value of the program counter. The counter is the address where the next instruction or datum will be placed. You can only read the variable *. You cannot assign a value to it with the statement ***=expr**.

Here's an example of using labels to mark routines in a program (don't type this in, it's only a fragment of a program):

```
50 JSR DISPLAY1; JUMP TO LABELED SUBROUTINE (LINE 90)
60 LDA $FF: BNE SKIPIT ; CONDITIONAL BRANCH AHEAD
  TO SKIPIT
70 TYA
80 SKIPIT: LDX #4: STA $8000,X: RTS;TARGET OF BRANCH IN
```

```

60
90 DISPLAY1=* ; THIS LABELS THE CURRENT PROGRAM
  COUNTER
100 ;
110 LDA #65: JSR $FFD2: RTS

```

Remember that in the lines above, the semicolon marks the beginning of a comment which, like a REM, is ignored by FA. The technique in line 90 is valuable if you think you may be adding some code at the beginning of the routine. As listed, the subroutine called DISPLAY1 starts with LDA #65, but later you could go in and add some lines between 90 and 110.

Three Passes to Assemble

Three passes are required to assemble source code (what you write) into object code (an executable ML program that the computer can follow). But FA doesn't do it by itself. You have to insert a loop that repeats three times with BASIC commands:

```

10 FOR PASS=1 TO 3
  .
  . (Insert source code)
  .
90 NEXT PASS:END

```

If you use an invalid addressing mode such as *LSR (expr),y* you'll see *ILLEGAL ADDRESSING MODE ERROR*. Mnemonics can only be used in program mode—that is, in a program you execute with RUN. If you enter a mnemonic in direct mode, you'll see *ILLEGAL DIRECT ERROR*.

Also note that for immediate addressing, the argument can be an actual number or an arithmetic expression which evaluates to a value in the range 0–255. Or you can substitute a string expression, in which case the assembler takes the ASCII value of the first character as the argument. If the string length is zero, the argument becomes zero.

Assembler Commands

Assembler commands which write data to the output device can only be used in program mode; otherwise you'll get *ILLEGAL DIRECT ERROR*. All assembler commands must be included in every pass.

ORG *address,mode,device,name*

This command must be used at the start of each pass. It does several things. First, it sets the origin (ORG), the memory address for the beginning of the ML program. It assigns an initial value to the program counter. It also sets the assembler mode, which should be 0 on the first two passes and 1 on the third and last. ORG also sets the output device and filename (if necessary).

Not all arguments are necessary. Also permitted are

ORG

ORG *address*

ORG *address,mode*

Default values for the arguments are

address = 49152 (\$C000)

mode = 0

device = 0 and no name

If you use a mnemonic or assembler command before you've used the command ORG, you'll see *UNDEF'D LOCATION COUNTER ERROR*.

The address assigns a value to the program counter. Usually, you use more than one pass to assemble the source program. Only during the last pass should the object code be written to memory or to the output device. Mode tells the assembler when the last pass is reached. Zero means it's not the last pass, so no object code should be produced, and there's no range checking for arguments and no checking for too large branches.

On the final pass, you should set the mode to 1, which signals the last pass, when object code is written to the output device.

Finally, you set the device number of the output device and a string expression which contains the filename if the object code is not written to memory. Zero means the output device is memory. Be careful not to write to memory locations where the assembler is placed (\$0801-\$121B) or where the BASIC interpreter is placed (\$A000-\$BFFF).

A device number in the range 8-11 means the output device is a disk drive. If mode is equal to 1, the assembler will open a PRG file with the name specified in the argument name. The logical file number will be 8.

BYTE *expression,expression,...*

This command writes numbers or characters to memory or the selected output device. It can have one or more arithmetic or string expressions separated by commas. Arithmetic expressions must give a positive value less than 256. The value will be placed in one byte. Each character of a string expression will be placed in one byte.

WORD *expression,expression,...*

This has the same function as BYTE except that values of arithmetic expressions must be positive and less than 65536. The value will be placed in two bytes in low/high format.

INCLUDE *name,device*

This command assembles a file from disk and inserts the resulting object code into memory or the output device. The file must be a normal PRG file and may not contain BASIC commands which cause a branch to another line or stop the program. Also not permitted are the BASIC commands DEF, RETURN, CLR, NEW, and the assembler commands SEND and INCLUDE.

The file is opened with a logical file number of 9. The file is closed when the end of the file is reached. The name is the filename you're including, and the device number can be 8-11 (use 8 if you have a single drive). If you have only one disk drive and you assemble to disk, the file(s) for the command INCLUDE must be on the same disk to which you assemble.

All variables and labels are *global*, which means you can pass parameters to INCLUDE files so they can work like macro-instructions. Let's say you're writing a program that needs to access several different disk files, and there are several points in the program that use the Kernal routines SETLFS, SETNAM, and OPEN. You could write the source code that performs these Kernal calls and save it to disk under the program name "OPEN" to be used later. Then, in the main program, use INCLUDE "OPEN",8. When the source code is compiled, the series of commands from the OPEN file are automatically inserted in the proper place in the object code.

SEND *stringexpr*

The command SEND may be used only if the object program is written to disk. It's used to link object code to a BASIC program. *Stringexpr* must contain a BASIC line with line number.

If you forget the line number, you'll get *MISSING LINE NUMBER ERROR*. If you want to send more than one line, you must use *SEND* for each line, and you have to send the lines in the right order. You must send the lines before the actual object code is written to disk. The address in the *ORG* command must be the start of BASIC RAM (2049).

UNSEND

If you load a program which consists of both BASIC and ML, the interpreter has to know where the BASIC part ends. *UNSEND* places a mark which the computer recognizes as the end of the BASIC part.

Example Programs

```
100 FOR PASS=1 TO 3:PRINT "PASS"PASS,
110 ORG $C000
120 IF PASS=3 THEN ORG $C000,1
130 START: LDX #0
140 LOOP: LDA TEXT,X:PRINT TEXT
150 BEQ EXIT
160 JSR $FFD2
170 INX
180 BNE LOOP
190 EXIT: RTS
200 PRINT *
210 TEXT: BYTE "EXAMPLE 1",0
220 NEXT PASS:END
```

Lines 110 and 120 show how to use the command *ORG*. In every pass, line 110 sets mode 0. But in pass 3, line 120 sets mode 1. The object code will start at 49152 (hexadecimal \$C000). Line 200 prints the current value of the location counter (*).

You can assemble the program with the command *RUN*. The program will give the following output:

PASS 1	0	49165
PASS 2	49165	49166
PASS 3	49166	49166

The first column is the pass number. The second column is the value of the label *TEXT* in the instruction *LDA TEXT,X* in line 140. The third column is the value the label should have when the source code is assembled. You can see that

only in pass 3 are these values equal to each other. This is because the assembler defaults to zero-page addressing. In pass 1, TEXT has a value less than 256, so zero-page addressing is assumed. This means a two-byte instruction instead of three. The value assigned to TEXT will be too low, as you can see in pass 1. In pass 2, this value, which is too low, will be used in assembling line 140. The assembler decides not to use zero-page addressing, so TEXT is assigned the correct value. In pass 3, the correct value replaces the previously incorrect values during assembly.

```

5 ;EXAMPLE PROGRAM 2
6 ;
10 PRINT CHR$(147)
11 DEF FN H(X)=INT(X/256)
12 DEF FN L(X)=X-256*FN H(X)
20 PRINT:PRINT" LOADER MAKER"
30 PRINT:PRINT" ENTER THE NAME OF THE PROGRAM T
   HAT"
40 PRINT" HAS TO BE LOADED BY THE LOADER."
50 INPUT" >";NAME$
60 PRINT:PRINT" ENTER THE NAME OF THE LOADER."
70 INPUT" >";N$
80 PRINT:PRINT" ENTER THE ADDRESS TO EXECUTE TH
   E"
90 PRINT" PROGRAM."
100 INPUT" >";ADDRESS:ADDRESS=ADDRESS-1
105 ;
110 FOR PASS=1 TO 3
115 ;
120 ORG 2049
130 IF PASS=3 THEN ORG 2049,1,8,N$
135 ;
140 SEND "10 SYS"+STR$(LOADER)
150 UNSEND
160 LOADER: LDA #8:TAX:LDY #1
170         JSR $FFBA
180         LDX #FN L(NAME)
190         LDY #FN H(NAME)
200         LDA #LEN(NAME$)
210         JSR $FFBD
220         LDA #FN H(ADDRESS)      :PHA
230         LDA #FN L(ADDRESS)      :PHA
240         LDA #0:JMP $FFD5
250 NAME:   BYTE NAME$
255 ;
260 NEXT PASS:CLOSE 8:END

```

The above example program shows how to use the commands SEND and UNSEND to write a program which includes a SYS within a BASIC line.

The main routine at lines 160–250 illustrates how to load another program from an ML program. Note that the lines up to 100 are BASIC—they prepare the variables and defined functions for use in the source code. If you assemble the program with the command RUN, you'll get a program that can load another ML program from disk and execute it. The object code will be written to disk.

In line 140, the command SEND writes a BASIC line to the output device by which you can load and run the program as if it were a normal BASIC program. Line 150 marks the end of the BASIC part of the object code.

The INPUTs in lines 50, 70, and 100 permit you to enter the parameters for the object program when the source program is assembled. In this way you can make different object programs with one source program.

Another advantage of writing the assembler as a BASIC extension is that you can assemble a program to the top of memory. Use the following construction to do this:

```
100 POKE 56,PEEK(56)-4:CLR
110 TOPOFMEM=PEEK(55)+256*(PEEK(56)+4)
120 ADDRESS=0:MODE=0
130 FOR PASS=1 TO 3
140 ORG ADDRESS
150 IF PASS=3 THEN ORG ADDRESS, MODE
.
.
. Source code
.
.
900 NEXT PASS
910 IF MODE=1 THEN END
920 ADDRESS=TOPOFMEM-*
930 MODE=1:GOTO 130
```

In this program, the source code goes through six passes. During the first three passes the location counter remains at 0. Mode 0 is used so the object program will not be written to the output device. The length of the program is calculated and subtracted from TOPOFMEM. This address is used in the second three passes to assemble to the top of memory. MODE is

set to 1 so the assembler will write the object code to the output device during the sixth pass (actually pass 3 of the second time around). Line 100 is used to reserve 1K at the top of memory for the object program.

Large Programs

If your source program won't fit into memory, you can split your program and use the command INCLUDE. For example:

```
10 FOR PASS=1 TO 3
20 ORG ADDRESS
30 IF PASS=3 THEN ORG ADDRESS,1
```

. Part 1 of source code

```
90 INCLUDE "PART 2",8
100 INCLUDE "PART 3",8
110 NEXT PASS:END
```

The labels and variables used in the INCLUDE files will be global variables, which means you can use them in arithmetic expressions everywhere in the program.

Another possibility is chaining the programs, but then you can't use a FOR-NEXT loop for the passes. You must use another way to define the passes. For example:

```
FIRSTPROGRAM
10 PASS=PASS+1:IF PASS=4 THEN
END
20 ORG ADDRESS
30 IF PASS=3 THEN ORG ADDRESS,1
```

. Source code

```
90 LOAD"SECONDPROGRAM",8
```

```
SECONDPROGRAM
```

. Source code

```
90 LOAD"FIRSTPROGRAM",8
```

Note that these are just examples. You'd have to insert your own source code as indicated. To chain programs, you would load and execute the first program. It controls the number of passes and loads the next program. The next program

loads the following program and so on until the last program, which must load the first again.

The Fast Assembler

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```
0801:0C 08 0A 00 9E 20 34 34 6C
0809:30 38 00 00 00 A9 36 85 D7
0811:01 A6 7A CA 30 0F BD 00 80
0819:02 C9 20 F0 F6 E8 BD 00 86
0821:02 C9 20 D0 01 E8 A0 04 A6
0829:84 0F BD 00 02 10 0A C9 25
0831:FF F0 06 E9 9F D0 16 A9 F2
0839:20 C9 20 F0 0C 85 08 C9 2F
0841:22 F0 4A C9 3F D0 4F A9 0A
0849:99 C8 99 FB 01 E8 D0 DA 73
0851:A4 71 24 0B 10 06 C8 A9 19
0859:90 99 FB 01 A5 0B 09 80 93
0861:C8 99 FB 01 24 0B 30 C2 3C
0869:C9 83 F0 05 49 8F D0 BA 92
0871:2C A9 3A 85 08 BD 00 02 DA
0879:F0 1A 24 0F 70 04 C5 08 25
0881:F0 C7 C9 22 D0 07 0A 45 53
0889:0F 85 0F A9 22 C8 99 FB 62
0891:01 E8 D0 E1 F0 5E 84 71 10
0899:86 7A A0 9E 84 22 A0 A0 18
08A1:84 23 A0 00 84 0B 51 22 E5
08A9:D0 07 E8 C8 BD 00 02 D0 50
08B1:F5 C9 80 D0 26 E8 BD 00 9C
08B9:02 20 1D 09 B0 92 A5 0B 2D
08C1:30 19 84 08 A0 0B D9 2E 54
08C9:09 F0 F1 88 10 F8 A4 08 17
08D1:C9 34 90 07 BD 00 02 C9 12
08D9:28 F0 E1 88 C8 B1 22 10 60
08E1:FB E6 0B A6 7A C8 B1 22 F1
08E9:D0 C2 C8 F0 0A A4 71 BD BE
08F1:00 02 D0 11 4C C9 A5 A9 2C
08F9:3A 85 22 A9 09 85 23 A9 B5
0901:80 85 0B D0 A7 E8 C8 99 2F
0909:FB 01 20 1D 09 B0 0A BD 0C
0911:00 02 F0 E0 20 1D 09 90 E7
0919:EC 4C 2B 08 C9 30 90 0B D5
0921:C9 3A 90 08 C9 41 90 03 B0
0929:C9 5B 24 38 60 04 18 23 65
0931:26 31 32 33 2E 2D 2C 2B C5
0939:2A 42 59 54 C5 57 4F 52 DD
0941:C4 4F 52 C7 41 55 54 CF 28
0949:4F 4C C4 49 4E 43 4C 55 B0
0951:44 C5 53 45 4E C4 55 4E 34
```

```

0959:53 45 4E C4 42 43 C3 42 65
0961:43 D3 42 45 D1 42 4D C9 A2
0969:42 4E C5 42 50 CC 42 56 9D
0971:C3 42 56 D3 41 44 C3 41 E1
0979:4E C4 43 4D D0 45 4F D2 2E
0981:4C 44 C1 4F 52 C1 53 42 7A
0989:C3 53 54 C1 41 53 CC 44 2E
0991:45 C3 49 4E C3 4C 53 D2 0E
0999:52 4F CC 52 4F D2 53 54 28
09A1:D8 43 50 D8 43 50 D9 4C E3
09A9:44 D8 4C 44 D9 53 54 D9 80
09B1:4A 53 D2 4A 4D D0 42 49 38
09B9:D4 42 52 CB 43 4C C3 43 E3
09C1:4C C4 43 4C C9 43 4C D6 23
09C9:44 45 D8 44 45 D9 49 4E 21
09D1:D8 49 4E D9 4E 4F D0 50 AB
09D9:48 C1 50 48 D0 50 4C C1 31
09E1:50 4C D0 52 54 C9 52 54 31
09E9:D3 53 45 C3 53 45 C4 53 2C
09F1:45 C9 54 41 D8 54 41 D9 2C
09F9:54 53 D8 54 58 C1 54 58 36
0A01:D3 54 59 C1 00 48 AD 8D 65
0A09:02 C9 01 F0 F9 68 10 16 67
0A11:C9 FF F0 12 24 0F 30 0E 15
0A19:50 0F C5 08 D0 08 48 A5 2F
0A21:0F 49 40 85 0F 68 4C F3 16
0A29:A6 C9 90 F0 18 48 C9 83 1D
0A31:F0 05 49 8F D0 0B 2C A9 D5
0A39:3A 85 08 A5 0F 49 40 85 CA
0A41:0F 68 4C 24 A7 C8 B1 5F E5
0A49:30 05 88 A9 90 D0 F3 84 96
0A51:49 AA A0 FF CA 10 08 C8 38
0A59:B9 3A 09 10 FA 30 F5 C8 48
0A61:B9 3A 09 10 03 4C EF A6 D2
0A69:20 47 AB D0 F2 20 73 00 E0
0A71:A2 00 86 0D C9 AC F0 16 71
0A79:86 62 86 63 C9 24 F0 3F 70
0A81:C9 25 F0 28 C9 27 F0 17 48
0A89:20 79 00 4C 8D AE 20 73 AB
0A91:00 A5 FB A4 FC 84 62 85 1D
0A99:63 A2 90 38 4C 49 BC 20 BE
0AA1:AC 0B 48 20 73 00 68 D0 17
0AA9:EE 4C 08 AF 20 73 00 B0 C3
0AB1:E8 C9 32 B0 E4 4A 26 63 FD
0AB9:26 62 90 F0 4C 48 B2 20 A3
0AC1:73 00 90 0A C9 47 B0 D1 E0
0AC9:C9 41 90 CD E9 07 29 0F CE
0AD1:A2 04 06 63 26 62 B0 E4 30
0AD9:CA D0 F7 05 63 85 63 90 5F
0AE1:DE 8D A8 02 85 45 20 B2 32

```

0AE9:0B 20 13 B1 B0 03 4C 08 3B
0AF1:AF A2 00 86 0D 86 0E 20 AD
0AF9:AC 0B 90 05 20 13 B1 90 CA
0B01:0A E0 07 B0 F2 E8 9D A8 5F
0B09:02 D0 EC E8 E0 08 B0 04 0D
0B11:A9 00 F0 F2 AD A9 02 85 E6
0B19:46 20 B2 0B C9 24 D0 06 E7
0B21:A9 FF 85 0D D0 13 C9 25 19
0B29:D0 1B A5 10 D0 C0 A9 80 81
0B31:85 0E 05 45 85 45 8D A8 87
0B39:02 A5 46 09 80 85 46 8D 47
0B41:A9 02 20 AC 0B 05 10 49 51
0B49:28 D0 03 4C D1 B1 84 10 3B
0B51:A5 2D A6 2E 86 60 85 5F 5D
0B59:E4 30 90 04 C5 2F B0 1A A6
0B61:A0 00 B9 A8 02 51 5F D0 6E
0B69:08 C8 C0 08 90 F4 4C 85 C4
0B71:B1 A5 5F 69 0D 90 DF E8 9F
0B79:D0 DA 68 48 C9 2A D0 03 DB
0B81:4C 23 B1 AD AA 02 D0 03 99
0B89:4C 28 B1 A5 2F A4 30 20 EC
0B91:47 B1 A0 07 B9 A8 02 91 42
0B99:5F 88 10 F8 A0 07 4C 74 41
0BA1:B1 10 05 A0 08 4C C7 B5 F5
0BA9:4C F6 B5 E6 7A D0 02 E6 CA
0BB1:7B A0 00 B1 7A C9 3A B0 E8
0BB9:03 4C 84 00 60 A6 9D 30 FD
0BC1:29 E0 21 F0 0B E0 20 F0 E4
0BC9:1A A9 0E A0 0E 4C 27 11 24
0BD1:A6 FD F0 0D A2 08 20 18 37
0BD9:E1 20 0C E1 20 CC FF F0 AD
0BE1:02 81 FB E6 FB D0 02 E6 55
0BE9:FC 60 4C AB B3 A5 66 30 0C
0BF1:12 A5 61 C9 91 B0 0C 20 CA
0BF9:9B BC A4 64 A5 65 85 14 C9
0C01:84 15 60 A5 9D 4A 90 03 41
0C09:4C 48 B2 A9 00 A8 F0 EE BD
0C11:20 73 00 F0 04 C9 3B D0 B3
0C19:07 A9 00 85 15 85 FE 60 94
0C21:C9 23 D0 04 A2 01 D0 08 04
0C29:A2 00 C9 28 D0 05 A2 80 AE
0C31:20 73 00 86 FE 20 9E AD 02
0C39:24 0D 30 0A 20 EE 0B A5 C5
0C41:FE 30 1C F0 51 60 A5 FE CD
0C49:C9 01 F0 03 4C 08 AF 20 D6
0C51:A6 B6 A0 00 84 15 C9 00 8A
0C59:F0 02 B1 22 85 14 60 A2 A2
0C61:00 86 0B A0 00 BD 9F 0C C8
0C69:51 7A D0 04 E8 C8 D0 F5 25
0C71:0A D0 0D A6 0B BD AC 0C 83


```

0C79:85 FE 20 FB A8 4C 73 00 35
0C81:E8 BD 9E 0C 10 FA E6 0B 57
0C89:BD 9F 0C D0 D6 A5 FE 30 72
0C91:BB A9 02 85 FE 60 A9 03 5A
0C99:85 0B A2 08 D0 C5 2C 58 5A
0CA1:A9 29 2C D9 A9 00 2C D8 7A
0CA9:2C D9 00 05 06 0F 03 04 15
0CB1:20 73 00 20 C4 0C 20 79 C8
0CB9:00 C9 3B D0 03 20 3B A9 71
0CC1:4C AE A7 F0 05 C9 3B D0 46
0CC9:05 60 38 4C ED A7 C9 AF 99
0CD1:D0 F8 A2 09 8A D0 72 D0 C2
0CD9:03 4C 2F A8 E9 80 B0 03 AC
0CE1:4C 08 AF C9 40 B0 F9 C9 37
0CE9:08 B0 0D 0A A8 B9 C7 0D 3D
0CF1:48 B9 C6 0D 48 4C 73 00 A0
0CF9:E9 08 AA C9 08 B0 39 BD 2E
0D01:D6 0D 20 BE 0B 20 73 00 79
0D09:F0 D6 C9 3B F0 D2 20 8A DB
0D11:AD A5 9D 4A 90 18 20 F7 E0
0D19:B7 18 A5 14 E5 FB AA 08 87
0D21:A5 15 E5 FC 28 10 02 49 AE
0D29:FF C9 00 D0 04 8A 4C BE 64
0D31:0B A9 26 A0 0E 4C 27 11 0B
0D39:C9 1F 90 0D C9 38 B0 A0 14
0D41:BD D6 0D 20 BE 0B 4C 73 C1
0D49:00 48 BD D6 0D 85 02 20 3D
0D51:11 0C A5 15 20 9D 0D F0 7F
0D59:0C 20 9D 0D F0 07 A9 46 43
0D61:A0 0E 4C 27 11 A5 02 29 97
0D69:0F A8 B9 39 0E 10 06 A5 62
0D71:FE 09 10 85 FE 68 38 E9 9B
0D79:08 A6 FE A8 18 B9 5D 0E 1C
0D81:7D 74 0E 20 BE 0B A5 FE A7
0D89:29 0F F0 38 29 08 08 A5 BC
0D91:14 20 BE 0B 28 F0 2D A5 4B
0D99:15 4C BE 0B F0 06 A5 FE C3
0DA1:09 08 85 FE A5 FE 4A 4A EA
0DA9:4A 4A A5 02 B0 04 29 0F 47
0DB1:90 04 4A 4A 4A 4A A8 A5 75
0DB9:FE 29 07 AA B9 39 0E 49 41
0DC1:FF 3D 94 0E 60 D4 0F D1 E4
0DC9:0F A7 0E 42 0F 11 10 2A 42
0DD1:10 B8 10 B0 10 90 B0 F0 44
0DD9:30 D0 10 50 70 05 05 05 ED
0DE1:05 05 05 05 06 17 11 11 70
0DE9:17 17 17 23 28 28 39 1A D8
0DF1:2B 2C 4C 24 00 18 D8 58 E2
0DF9:B8 CA 88 E8 C8 EA 48 08 4D
0E01:68 28 40 60 38 F8 78 AA AA

```

0E09:A8 BA 8A 9A 98 55 4E 44 1E
0E11:45 46 27 44 20 4C 4F 43 9E
0E19:41 54 49 4F 4E 20 43 4F D1
0E21:55 4E 54 45 D2 42 52 41 DF
0E29:4E 43 48 20 4F 55 54 20 E0
0E31:4F 46 20 52 41 4E 47 C5 47
0E39:1C 0C 04 14 84 6E 6C 0D EB
0E41:86 96 8E 8C 80 49 4C 4C EE
0E49:45 47 41 4C 20 41 44 44 99
0E51:52 45 53 53 49 4E 47 20 B9
0E59:4D 4F 44 C5 60 20 C0 40 1A
0E61:A0 00 E0 80 01 C1 E1 41 06
0E69:21 61 81 E0 C0 A2 A0 80 FE
0E71:14 40 20 09 09 05 15 15 D7
0E79:01 11 80 09 80 0D 1D 19 86
0E81:80 80 80 80 00 04 14 14 62
0E89:80 80 80 80 80 0C 1C 1C A6
0E91:80 80 2C 01 02 04 08 10 E3
0E99:20 40 80 20 79 00 F0 04 99
0EA1:C9 3B D0 02 68 68 60 A9 FA
0EA9:00 85 FB 85 FD 85 B7 A9 1E
0EB1:C0 85 FC A5 9D 09 20 85 60
0EB9:9D 20 9C 0E 20 8A AD 20 C7
0EC1:F7 B7 84 FB 85 FC 20 9C 15
0EC9:0E 20 00 E2 E0 02 90 03 56
0ED1:4C 48 B2 8A 05 9D 85 9D 6C
0ED9:20 9C 0E 20 00 E2 86 FD 87
0EE1:20 0E E2 20 9E AD 20 A3 7F
0EE9:B6 20 BD FF A6 FD F0 B6 E6
0EF1:A9 08 A0 01 20 BA FF A5 9A
0EF9:9D 4A 90 AA A5 B7 D0 06 E7
0F01:A5 FD C9 08 B0 15 20 C1 07
0F09:E1 A2 08 20 18 E1 A5 FB 53
0F11:20 0C E1 A5 FC 20 0C E1 3B
0F19:4C CC FF 20 10 F7 4C 3B C6
0F21:A4 20 8B B0 85 49 84 4A BA
0F29:20 79 00 F0 07 C9 3B F0 8B
0F31:03 4C AC A9 A6 3A E8 D0 D4
0F39:03 4C 08 AF 20 92 0A 4C 93
0F41:D0 BB F0 10 20 6B A9 A5 7D
0F49:14 8D B1 02 A5 15 8D B2 7A
0F51:02 A9 80 2C A9 00 8D B0 C6
0F59:02 60 AD B0 02 29 BF 2C B1
0F61:A9 00 8D B0 02 A6 14 8E 72
0F69:FE 01 A6 15 8E FF 01 2C 10
0F71:B0 02 70 24 20 60 A5 86 0D
0F79:7A 84 7B 20 73 00 AA F0 49
0F81:D9 A2 FF 86 3A 90 03 4C 04
0F89:96 A4 AD B0 02 09 40 8D 1F
0F91:B0 02 20 79 00 4C 9C A4 33

```

0F99:10 C6 18 AD FE 01 6D B1 D7
0FA1:02 85 63 AD FF 01 6D B2 FA
0FA9:02 85 62 B0 B3 A2 90 38 03
0FB1:20 49 BC 20 DF BD 78 A2 55
0FB9:00 BD 00 01 C9 30 90 06 8D
0FC1:9D 77 02 E8 D0 F3 A9 20 25
0FC9:9D 77 02 E8 86 C6 58 D0 34
0FD1:A3 A9 08 2C A9 00 85 02 4A
0FD9:20 9E AD 24 0D 30 1F 20 2F
0FE1:EE 0B A5 02 D0 0D A6 15 2C
0FE9:F0 09 A6 9D E0 20 F0 03 DD
0FF1:4C 48 B2 20 8D 0D 20 9C 1E
0FF9:0E 20 0E E2 D0 DA 20 A6 EF
1001:B6 85 64 A0 00 C4 64 F0 41
1009:ED B1 22 20 BE 0B C8 D0 57
1011:F4 A0 01 A9 08 91 2B 20 8B
1019:33 A5 A5 22 A4 23 65 2D BC
1021:85 2D 90 01 C8 84 2E 4C 72
1029:5C A6 20 9E AD 20 A3 B6 FA
1031:20 BD FF 20 00 E2 A9 09 BA
1039:A0 00 20 BA FF 20 C1 E1 3F
1041:A2 09 20 1E E1 20 12 E1 70
1049:A5 90 29 02 F0 05 A9 04 98
1051:4C 3B A4 20 12 E1 20 CC 22
1059:FF A5 7A 48 A5 7B 48 A2 05
1061:09 20 1E E1 20 12 E1 20 1D
1069:12 E1 AA F0 34 20 12 E1 97
1071:20 12 E1 A2 FF E8 20 12 82
1079:E1 9D 00 02 A8 D0 F6 20 A8
1081:CC FF A9 FF 85 7A A9 01 A7
1089:85 7B 20 73 00 20 C4 0C 9C
1091:20 79 00 AA F0 C9 C9 3A 47
1099:F0 F0 C9 3B F0 C1 4C 08 8A
10A1:AF 20 CC FF A9 09 20 C3 B0
10A9:FF 68 85 7B 68 85 7A 60 FA
10B1:20 B4 10 A9 00 4C BE 0B 65
10B9:20 9E AD 20 A3 B6 A8 A9 3C
10C1:00 C0 58 90 03 4C 71 A5 F7
10C9:99 00 02 88 30 04 B1 22 96
10D1:90 F6 A5 7A 48 A5 7B 48 6C
10D9:A9 FF 85 7A A9 01 85 7B FE
10E1:20 73 00 90 07 A9 14 A0 9F
10E9:11 4C 27 11 20 6B A9 20 BD
10F1:7C A5 A5 14 8D FE 01 A5 BF
10F9:15 8D FF 01 8C FC 01 8C FE
1101:FD 01 84 0B A0 00 B9 FC 19
1109:01 20 BE 0B C8 C4 0B 90 3C
1111:F5 B0 96 4D 49 53 53 49 89
1119:4E 47 20 4C 49 4E 45 20 2B
1121:4E 55 4D 42 45 D2 85 22 30

```

1129:84 23 4C 47 A4 66 0F 0E 3F
1131:08 06 0A B1 0C 6E 0A A9 0D
1139:37 85 01 A9 00 85 58 85 5F
1141:5A 85 5F A9 C0 85 59 85 CC
1149:5B A9 A0 85 60 20 BB A3 8E
1151:A2 09 BD 2E 11 9D 02 03 A7
1159:CA 10 F7 A9 4C 8D C4 B5 56
1161:8D 92 B0 8D 43 B1 8D A5 7F
1169:A9 A9 0D 8D 4E B5 8D 55 FE
1171:B1 A9 08 8D 0F B6 8D 89 A8
1179:B1 A9 EA 8D 10 B6 A9 60 24
1181:8D 69 B1 A9 8C A0 0B 8D 20
1189:44 B1 8C 45 B1 A9 E2 A0 BA
1191:0A 8D 93 B0 8C 94 B0 A9 5B
1199:A2 A0 0B 8D C5 B5 8C C6 54
11A1:B5 A9 D7 A0 0C 8D 2C A0 9D
11A9:8C 2D A0 A9 22 A0 0F 8D 4B
11B1:A6 A9 8C A7 A9 A9 C4 A0 BB
11B9:0C 8D 49 A9 8C 4A A9 A9 93
11C1:FB A0 F0 8D 20 D0 8C 21 7F
11C9:D0 A9 1B A0 12 8D 81 02 F7
11D1:8C 82 02 20 BF E3 A9 E4 E2
11D9:A0 11 20 1E AB 20 30 E4 99
11E1:4C 9D E3 93 08 0E 9B 11 08
11E9:20 20 2A 20 C1 D3 D3 C5 36
11F1:CD C2 CC C5 D2 20 56 33 98
11F9:20 2A 20 20 20 42 59 20 99
1201:D9 56 45 53 20 C8 41 4E 7A
1209:20 20 31 39 38 35 0D 0D BC
1211:20 20 20 20 20 20 20 20 35
1219:20 00 65 00 00 00 00 00 F9

Window Wizard

Ronald Carnell

You'll be amazed at your screen display when you use this truly powerful utility. Loaded with features, "Window Wizard" adds 19 new commands that can give your programs a brand-new, professional look. An impressive demo program is included.

You've probably seen windows in action. The Amiga, Atari ST, and Macintosh all make extensive use of them. If you want a menu on one of these machines, just pull it down from the top of the screen. And when you're finished with it, it quietly disappears—leaving the screen exactly as it found it.

A window is basically a miniature screen that overlays the primary screen. You can print anything you like within a window, and when it disappears, the primary screen is restored, undisturbed and unchanged. Ideally, a window can be any size (even as large as the primary screen) and can be placed anywhere. Its principal function is to offer menus or other information while not destroying what's beneath it. It doesn't hurt if it looks good, too.

"Window Wizard" is a machine language program that gives you windowing capability. Now you can have pull-down menus, magic messages, and professional-looking prompts.

Machine Language Windows

Program 1, "Window Wizard," is written entirely in machine language, so you'll need to use the "MLX" machine language entry program, found in Appendix D, to type it in. When you run MLX, you'll be asked for starting and ending addresses. The proper values for Window Wizard are

Starting address: C000

Ending address: CCC7

After typing in all the data, save a copy to tape or disk before leaving MLX. If you plan to test it with Program 2, "WINDOW DEMO," save the data with the filename WINDOW WIZARD.

To run the program, enter

LOAD "WINDOW WIZARD",8,1 (for disk)

LOAD "WINDOW WIZARD",1,1 (for tape)

Then type

NEW

and

SYS 49152

Program 2 is a demonstration written in BASIC. Type it in and save it to tape or disk. The first thing it does is load Window Wizard, so before running Program 2, make sure the disk containing Program 1 is in your disk drive. If you're using a Datasette, change the load command in line 10 to **LOAD "WINDOW WIZARD",1,1**.

Windowing is Wizard's primary task, but the 19 new statements it adds to BASIC also cover many other areas of screen manipulation.

Wizard Commands

The following is a list of the new statements added by Window Wizard. In the discussion below, parameters shown in ***bold*** are mandatory, while those shown in *normal* are optional.

!DIM *row, column, width, height*

This command tells Wizard where to put your window and how big to make it. *Row* and *column* define the upper left corner of the window. For example, the command **!DIM(10)** would tell Wizard you want your next window to be positioned on row 10—but *column, width, and height* would remain unchanged from the values of the last **!DIM** command.

!DEFINE *tone, 3-D, hue, bchr, ichr, lrate, brate*

!DEFINE is a mouthful, but is one of the most powerful commands in Wizard. The seven parameters (remember, they don't all have to be included each time) represent the options you may select to customize your windows. Let's discuss each separately.

- *tone*. As each letter of your windowed message is printed, Wizard sounds a random tone. It can draw attention to your window. **!DEFINE** allows you to toggle this option on

or off by placing the proper value as the first parameter. A value of 0 turns off the sound and 1 turns it back on.

- *3-D*. One of the first things you may notice about Wizard's windows is that they appear to be three-dimensional. This is accomplished by simulating a shadow effect under the window. A value of 0 turns off the shadow and 1 turns it on.

- *hue*. This parameter controls the color of the window (not the color of the text, though). Place any value from 0 to 15 here.

- *bchr*. This refers to the border character used for the window. Wizard defaults to a 0 value here, which provides a special, thin-lined border. But by supplying a different value, you can request a new character for the border. *Don't use the ASCII code*. ASCII is used for printing, but the screen codes are used for POKEing. Wizard POKes your window onto the screen. For example, 65 is the ASCII code for the letter A. But the screen code for A is 1, and that's the value you would use in the !DEFINE command if you wanted to use an A for the border.

- *ichr*. This is similar to *bchr* and represents the interior character of the window. Normally, this would be the screen code for a space character (32), but you might wish to change it to a reversed space (160) or something else entirely.

- *lrate*. For aesthetic reasons, your message is printed at a controlled rate. The effect is interesting, and it also allows the tone option time to make a sound. You control the speed of printing by supplying a value between 1 (very fast) and 255 (sit-back-and-relax time). The default value for *lrate* is 50.

- *brate*. Windows start out on the screen as 1 × 1 boxes and grow to the size you !DIMensioned them to be. You control the rate of growth with this parameter. Like *lrate*, *brate* has a default value of 50.

!OPEN, message

When you !OPEN a window, the obvious thing that happens is a window appears and your message (this can be a literal string in quotes or a variable) is very neatly printed into it using word-wrap functions. But !OPEN is actually doing much more. Among other things, it is "memorizing" the current screen. Later, when you !CLOSE the window, Wizard reverses this process by putting back the same screen.

Let's look at an example. You have just !OPENed your first window. When you later !CLOSE it, the screen beneath is

undisturbed. Such, after all, is the way of windows. But let's say you don't !CLOSE it right away. Instead, you !OPEN another window. And let's suppose this second window overlaps the first. What happens when you !CLOSE the second window? The first window is still there. And of course, beneath that is our original screen. Just how many windows can you have on the screen before you start destroying what's underneath?

Wizard remembers your original screen plus seven windowed screens, for a total of eight. But, when you call that eighth window, only the seventh window is destroyed. The eighth replaces the seventh. The first six windows, along with the original screen of course, are saved forever.

!CLOSE

!CLOSE makes a window go away. The previous screen (underneath the window) is restored. See !OPEN above.

!RESTORE *screen*

When Wizard memorizes a screen, it pushes the screen onto a last-in-first-out (LIFO) stack. All this means is that when Wizard replaces a screen it always pulls off the last screen it memorized. As long as all of your !OPEN commands are matched by !CLOSE commands, you can let Wizard manage the details. But there may be instances when you want more direct control, and !RESTORE gives you that control.

!RESTORE by itself resets Wizard's LIFO stack to 0; there will be zero memorized screens. By following !RESTORE with a parameter from 1 to 8, you can control where Wizard pulls the next screen from the stack. !RESTORE(5), for example, tells Wizard that the next !CLOSE command should pull in the fifth saved screen. PEEK(49155) tells you which screen is currently being pointed to.

!SAVE

Wizard's !SAVE and !LOAD commands work with screen memorization more directly. !SAVE instructs Wizard to memorize the current screen without !OPENing a window, and !LOAD tells it to replace a screen.

!LOAD

If you wish to restore a !SAVED screen, use this command. See !SAVE above.

!CLR

!CLR erases the contents of the current window.

!AGAIN, *message*

This would be the command to use if you wanted to print something into the current window again. This command doesn't memorize the screen, so reusing a window doesn't count as one of the eight screens Wizard can remember. !AGAIN does an automatic !CLR.

Incidentally, whenever Wizard is handling text, as opposed to borders and shadows, it will print rather than POKE. This means that all the many PRINT command options are still available. (Of course, printing a CLR/HOME in a window might be a little strange.)

!KINETIC *direction, width, height*

!KINETIC allows you to pass dynamic control of the current window directly to the user. The *direction* parameter tells Wizard to move the window one character in that direction. (This command and the next one use numeric values for directions: 0 = do nothing, 1 = up, 2 = right, 3 = down, and 4 = left.) *Width* and *height*, if used, replace the old values, and allow changes in the window's size. For technical reasons, !KINETIC temporarily turns off any 3-D effect.

!SCROLL *direction, amount*

!SCROLL can be used with or without a window. In either case, the area to be !SCROLLED is determined by the last !DIM command. You can !SCROLL in any of four directions (see the !KINETIC command) and you can !SCROLL any number of lines. If a number isn't specified, the default is one line.

!MENU, *menu string, return variable*

The only difference between !MENU and !OPEN is the return string. Like !OPEN, !MENU draws a window and prints a message (it's up to you to insure that the first parameter holds a meaningful menu). Unlike !OPEN, !MENU doesn't immediately jump back to BASIC, but waits for the user to press a key. And whatever key is pressed is put into the string variable specified in the second parameter. For example, !MENU, M\$, K\$ would print the menu held in the variable M\$, wait for a keypress, and put the letter pressed into the variable K\$.

!BOX

This is a pseudo-window command. !BOX draws what looks like a window, based on !DIM and !DEFINE, but it doesn't memorize the screen. A !BOX cannot be !CLOSED.

!DISPLAY, *message*

This one is like !BOX (see above), but allows you to add a message. !DISPLAY prints a message, based on !DIM, but again will not memorize the screen.

!TONE, *duration*

!TONE produces a single random tone. You provide the duration of the sound.

!HUE *bkgnd, border, text*

This command, and those remaining, have little to do with windows, but do provide more control over the screen. The first two parameters of !HUE replace the POKes usually necessary to change screen colors, and work identically to those POKes. The third parameter is slightly different. Changing text color normally changes only future text; what was previously printed on the screen won't change. But if you use the third parameter of !HUE to change text color, everything currently printed on the screen also changes to that color. Use values between 0 and 15 to change the color of background, border, and text.

!ERASE, *row*

You can !ERASE any row from 0 to 24 with this command. Use it in a loop to clear larger portions of the screen.

!PRINT@, *row, column, message*

This command (read it as *print at*) is in many versions of BASIC, and for good reason. It's powerful. If you execute the BASIC statement PRINT A\$, BASIC prints the string where the cursor was last sitting. !PRINT@, however, positions the cursor where you specify through the row and column setting before printing the string.

!CENTER, *row, message*

This command prints a message on a specified row, and Wizard centers the message on the 40-column screen.

If you multiply those 19 new commands by all the options available, you come up with a lot to remember, right? Wrong. Reach up and tap the RESTORE key anytime Wizard

is activated, and a full review of all Wizard commands and options appears.

Error Trapping

Wouldn't it be great if you could write a program and never make a mistake? Unfortunately, a lot of time and effort always goes into finding out the many errors that so insidiously creep into programs. Wizard tries to make that task a little easier.

You will, of course, run across the infamous *SYNTAX ERROR* if you misspell a Wizard command. You'll see the same error if you leave off a necessary parameter (one following a comma). You might even pull out a *STRING TOO LONG ERROR* if you try to *!CENTER* a string over 40 characters. But the most probable danger is trying to use a window too large for the screen.

Consider. If you start your window on row 18 and make the height more than 6, the bottom of your window goes off-screen. Where does it go? Since Wizard *POKEs* the window border into screen memory, it would *POKE* the bottom of your window into the memory following screen memory. What normally sits there? Your BASIC program. Disaster ensues.

Of course, that can't be allowed to happen. So, Wizard responds with an *ILLEGAL QUANTITY ERROR* if your window is too large for the screen. There are three commands in Wizard where this might happen. *!DIM* and *!KINETIC* are obvious; change your values and try again. But why does *!DEFINE* keep causing an error? Check to see if you've asked for a shadow. Then check to see if your window has left room on the screen for a shadow. After all, we can't let a window's shadow bleed offscreen either.

Windows add a significant touch of professionalism to your BASIC programs. There are few things more impressive than quality-looking windows appearing and disappearing when you want. And yet, windows wouldn't be making such an impression on the programming scene if their only function was to look good.

I've been using Window Wizard for several months, and am amazed at the freedom that windows can give a programmer. Screen layouts are now a snap, menus are instantly accessible, prompts no longer need to be short and cryptic, and programs are significantly shorter and easier to follow when so much less of the code is devoted to screen manipulation.

Program 1. Window Wizard

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```
C000:4C 1D C0 00 20 CB 00 00 37
C008:05 14 12 0E 01 01 01 00 42
C010:20 32 32 9C DC 00 00 00 25
C018:00 00 00 00 00 A5 7C C9 F2
C020:4C D0 03 4C 3F C0 A9 4C BD
C028:85 7C A9 7A 85 7D A9 C0 9E
C030:85 7E 78 A9 58 8D 18 03 E9
C038:A9 C0 8D 19 03 58 60 A9 E5
C040:C9 85 7C A9 3A 85 7D A9 BE
C048:B0 85 7E 78 A9 47 8D 18 78
C050:03 A9 FE 8D 19 03 58 60 5C
C058:48 8A 48 98 48 78 20 BC 54
C060:F6 20 E1 FF D0 08 68 A8 C1
C068:68 AA 68 4C 66 FE 58 20 9A
C070:24 CB 68 A8 68 AA 68 78 C5
C078:40 00 48 C9 21 D0 18 A5 E1
C080:7B C9 02 F0 12 A5 D4 D0 23
C088:0E AD 79 C0 C9 A7 D0 04 4A
C090:A5 61 F0 03 68 D0 10 68 9A
C098:C9 20 F0 03 8D 79 C0 C9 F2
C0A0:3A 90 01 60 4C 80 00 20 0E
C0A8:73 00 48 20 73 00 F0 0C 78
C0B0:C9 2C F0 08 C9 28 F0 04 95
C0B8:C9 3A D0 EF 68 C9 9F F0 61
C0C0:49 C9 A0 F0 4B C9 44 F0 77
C0C8:4D C9 99 F0 52 C9 86 F0 5D
C0D0:51 C9 96 F0 50 C9 A4 F0 33
C0D8:4F C9 9C F0 4E C9 8C F0 BA
C0E0:50 C9 45 F0 4F C9 41 F0 C9
C0E8:4E C9 94 F0 4D C9 93 F0 4F
C0F0:1F C9 48 F0 4B C9 53 F0 A5
C0F8:59 C9 43 F0 46 C9 4D F0 F5
C100:45 C9 4B F0 44 C9 42 F0 CF
C108:43 60 20 DA CA 4C 55 C1 EA
C110:20 BE C7 4C 55 C1 20 40 43
C118:C5 20 22 C8 4C 55 C1 4C DE
C120:CA C1 4C 6A C1 4C 8E C1 C7
C128:4C B2 C1 20 F8 C1 4C 55 75
C130:C1 4C 58 C2 4C 93 C2 4C 61
C138:43 C2 20 67 C7 4C 55 C1 64
C140:4C AD C2 4C FC C2 4C 25 23
C148:C3 4C 62 C3 20 C6 C5 4C 3D
C150:55 C1 20 E5 C2 20 79 00 DA
C158:F0 0D C9 3A F0 09 20 73 D3
C160:00 F0 04 C9 3A D0 F7 4C 8E
C168:79 00 A9 00 8D 18 C0 20 4C
C170:40 C5 AD 1A C0 D0 11 98 E0
```

```

C178:AC 18 C0 99 08 C0 EE 18 43
C180:C0 AD 18 C0 C9 04 90 E7 46
C188:20 2A C4 4C 55 C1 A9 00 09
C190:8D 18 C0 20 40 C5 AD 1A 89
C198:C0 D0 11 98 AC 18 C0 99 3D
C1A0:0C C0 EE 18 C0 AD 18 C0 67
C1A8:C9 07 90 E7 20 2A C4 4C E2
C1B0:55 C1 AD 0C C0 48 A9 01 41
C1B8:8D 0C C0 20 40 C5 98 AA 15
C1C0:20 9D C4 68 8D 0C C0 4C 45
C1C8:55 C1 20 40 C5 98 48 20 B0
C1D0:40 C5 68 AA 18 20 F0 FF C0
C1D8:20 40 C5 AD 04 C0 85 FB 3A
C1E0:AD 05 C0 85 FC A0 FF C8 20
C1E8:CC 06 C0 B0 08 B1 FB 20 96
C1F0:D2 FF 4C E7 C1 4C 55 C1 91
C1F8:EE 08 C0 EE 09 C0 AE 0A AF
C200:C0 CA CA 8E 0A C0 AE 0B 96
C208:C0 CA CA 8E 0B C0 AD 0F A8
C210:C0 48 AD 10 C0 8D 0F C0 D9
C218:AD 0D C0 48 A9 00 8D 0D C9
C220:C0 20 C6 C5 68 8D 0D C0 97
C228:68 8D 0F C0 AE 0B C0 E8 3F
C230:E8 8E 0B C0 AE 0A C0 E8 43
C238:E8 8E 0A C0 CE 09 C0 CE 0E
C240:08 C0 60 20 60 C4 20 82 E0
C248:C4 20 F8 C1 20 40 C5 20 21
C250:22 C8 20 EC CA 4C 55 C1 DF
C258:A9 9C 8D 13 C0 A9 DC 8D B0
C260:14 C0 A0 00 8C 03 C0 20 46
C268:40 C5 AD 1A C0 D0 21 C0 23
C270:09 90 05 A2 0E 4C 37 A4 1E
C278:8C 03 C0 AD 13 C0 18 69 2D
C280:04 8D 13 C0 AD 14 C0 18 31
C288:69 04 8D 14 C0 88 D0 EB 6C
C290:4C 55 C1 20 92 C4 20 40 F3
C298:C5 98 AA E0 19 90 05 A2 42
C2A0:0E 4C 37 A4 20 FF E9 20 66
C2A8:19 CB 4C 55 C1 20 40 C5 61
C2B0:8C 21 D0 20 40 C5 AD 1A 6F
C2B8:C0 D0 27 8C 20 D0 20 40 45
C2C0:C5 AD 1A C0 D0 1C 8C 86 7A
C2C8:02 A9 00 85 FB A9 D8 85 CF
C2D0:FC A2 04 A0 00 AD 86 02 CD
C2D8:91 FB C8 D0 FB E6 FC CA 8C
C2E0:D0 F6 4C 55 C1 20 40 C5 40
C2E8:98 48 20 40 C5 AD 1A C0 AE
C2F0:F0 02 A0 01 8C 1C C0 68 52
C2F8:20 5F C9 60 20 40 C5 98 CB
C300:48 20 40 C5 AD 06 C0 C9 E8

```

C308:29 90 06 68 A2 17 4C 37 D0
C310:A4 38 A9 28 ED 06 C0 4A 03
C318:A8 68 AA 18 20 F0 FF 4C F5
C320:DB C1 4C 55 C1 20 60 C4 F8
C328:20 82 C4 20 67 C7 20 40 D5
C330:C5 20 C6 C5 20 22 C8 A9 9C
C338:00 85 C6 20 E4 FF F0 FB 01
C340:48 20 40 C5 A0 00 A9 01 B1
C348:91 47 C8 B1 47 85 FB C8 AF
C350:B1 47 85 FC A0 00 68 91 6A
C358:FB 20 EC CA 20 BE C7 4C 08
C360:55 C1 A9 00 8D 76 03 8D 12
C368:77 03 8D 78 03 AD 06 C0 41
C370:48 20 40 C5 AD 1A C0 D0 B0
C378:19 8C 76 03 20 40 C5 AD E9
C380:1A C0 D0 0E 8C 77 03 20 A8
C388:40 C5 AD 1A C0 D0 03 8C D4
C390:78 03 AD 77 03 F0 03 8D B1
C398:0A C0 AD 78 03 F0 03 8D 02
C3A0:0B C0 AD 76 03 F0 1C C9 D8
C3A8:01 D0 03 CE 08 C0 C9 02 0B
C3B0:D0 03 EE 09 C0 C9 03 D0 D3
C3B8:03 EE 08 C0 C9 04 D0 03 8D
C3C0:CE 09 C0 68 8D 06 C0 20 B6
C3C8:2A C4 20 60 C4 20 82 C4 11
C3D0:AD 13 C0 48 AD 14 C0 48 18
C3D8:AD 03 C0 48 20 BE C7 68 88
C3E0:8D 03 C0 68 8D 14 C0 68 35
C3E8:8D 13 C0 AD 11 C0 48 AD B8
C3F0:0D C0 48 AD 0C C0 48 A2 A9
C3F8:01 8E 11 C0 CA 8E 0C C0 3C
C400:8E 0D C0 8E 0D C0 AD 0A E5
C408:C0 8D 15 C0 AD 0B C0 8D AC
C410:16 C0 20 09 C6 20 22 C8 2D
C418:68 8D 0C C0 68 8D 0D C0 1B
C420:68 8D 11 C0 20 EC CA 4C 06
C428:55 C1 18 AD 08 C0 6D 0B D3
C430:C0 6D 0D C0 C9 1A B0 23 5E
C438:AD 09 C0 6D 0A C0 C9 29 D9
C440:B0 19 AD 0B C0 C9 02 90 90
C448:12 AD 0A C0 C9 02 90 0B 16
C450:AD 0D C0 F0 05 AD 09 C0 CC
C458:F0 01 60 A2 0E 4C 37 A4 85
C460:A2 00 B5 FB 9D 5D 03 E8 03
C468:E0 04 D0 F6 B5 41 9D 5D 38
C470:03 E8 E0 08 D0 F6 A5 14 14
C478:9D 5D 03 E8 A5 15 9D 5D 31
C480:03 60 A2 00 B5 D9 9D DE 27
C488:03 09 80 95 D9 E8 E0 19 8C
C490:D0 F2 38 20 F0 FF 8E 6F 5C

```

C498:03 8C 70 03 60 AD 0C C0 97
C4A0:F0 2D A9 F0 8D 06 D4 A9 0A
C4A8:08 8D 0F D4 A9 81 8D 12 49
C4B0:D4 A9 0F 8D 18 D4 A9 17 48
C4B8:8D 04 D4 AD 1B D4 8D 01 C7
C4C0:D4 20 CF C4 A2 18 A9 00 CB
C4C8:9D 00 D4 CA D0 FA 60 20 BB
C4D0:B3 EE CA D0 FA 60 A9 00 03
C4D8:85 FB AD 88 02 85 FC A9 2C
C4E0:00 85 FD A9 D8 85 FE AC AD
C4E8:08 C0 C8 88 F0 09 20 10 44
C4F0:C5 20 1C C5 4C EB C4 18 F8
C4F8:A5 FB 6D 09 C0 85 FB 90 37
C500:02 E6 FC 18 A5 FD 6D 09 70
C508:C0 85 FD 90 02 E6 FE 60 28
C510:18 A5 FB 69 28 85 FB 90 07
C518:02 E6 FC 60 18 A5 FD 69 C0
C520:28 85 FD 90 02 E6 FE 60 F3
C528:18 A5 45 69 28 85 45 90 DA
C530:02 E6 46 60 18 A5 47 69 94
C538:28 85 47 90 02 E6 48 60 C7
C540:A0 00 8C 06 C0 8C 1A C0 3B
C548:B1 7A C9 2C F0 09 C9 28 AE
C550:F0 05 C8 8C 1A C0 60 20 2C
C558:73 00 B1 7A C9 22 D0 32 26
C560:20 73 00 A6 7A A4 7B 8E 2F
C568:04 C0 8C 05 C0 A0 FF A2 33
C570:00 C8 B1 7A C9 22 F0 F9 BE
C578:C9 00 F0 08 C9 3A F0 04 A4
C580:E8 4C 71 C5 8E 06 C0 20 4C
C588:73 00 F0 04 C9 3A D0 F7 FC
C590:38 60 C9 3A B0 08 20 9E B1
C598:AD 20 AA B1 18 60 20 8B 81
C5A0:B0 A5 0D F0 15 A0 00 B1 7B
C5A8:47 8D 06 C0 C8 B1 47 8D 31
C5B0:04 C0 C8 B1 47 8D 05 C0 DD
C5B8:38 60 A5 47 A4 48 20 A2 CA
C5C0:BB 20 AA B1 18 60 A9 01 39
C5C8:8D 15 C0 8D 16 C0 A9 00 58
C5D0:8D 19 C0 A9 00 8D 6D 03 30
C5D8:AD 15 C0 CD 0A C0 B0 06 30
C5E0:EE 15 C0 EE 6D 03 AD 16 19
C5E8:C0 CD 0B C0 B0 06 EE 16 47
C5F0:C0 EE 6D 03 AD 6D 03 F0 90
C5F8:0F 20 09 C6 AE 12 C0 20 01
C600:CF C4 20 07 C7 4C CE C5 ED
C608:60 20 D6 C4 A2 01 A0 00 4F
C610:E0 01 F0 66 38 8A ED 16 B0
C618:C0 C9 FF D0 05 A9 00 8D E1
C620:19 C0 EC 16 C0 90 03 4C 04

```

C628:C2 C6 C0 00 F0 1C 38 98 E1
 C630:ED 15 C0 C9 FF F0 24 AD 68
 C638:19 C0 D0 0A AD 10 C0 91 FD
 C640:FB AD 0E C0 91 FD C8 4C 67
 C648:2A C6 AD 0F C0 D0 02 A9 3A
 C650:74 91 FB AD 0E C0 91 FD 6B
 C658:C8 D0 CF 88 AD 10 C0 91 C1
 C660:FB C8 AD 0F C0 D0 02 A9 BB
 C668:6A 91 FB AD 0E C0 91 FD 7E
 C670:E8 20 10 C5 20 1C C5 4C 22
 C678:0E C6 C0 00 F0 1B 38 98 D3
 C680:ED 15 C0 C9 FF F0 23 AD B6
 C688:0F C0 D0 02 A9 77 91 FB 52
 C690:AD 0E C0 91 FD C8 4C 7A CF
 C698:C6 AD 0F C0 D0 02 A9 4F 14
 C6A0:91 FB AD 0E C0 91 FD C8 9D
 C6A8:D0 D0 AD 0F C0 D0 02 A9 70
 C6B0:50 91 FB AD 0E C0 91 FD B9
 C6B8:E8 20 10 C5 20 1C C5 4C 6A
 C6C0:0E C6 C0 00 F0 1B 38 98 1C
 C6C8:ED 15 C0 C9 FF F0 23 AD FE
 C6D0:0F C0 D0 02 A9 6F 91 FB 7A
 C6D8:AD 0E C0 91 FD C8 4C C2 60
 C6E0:C6 AD 0F C0 D0 02 A9 4C 59
 C6E8:91 FB AD 0E C0 91 FD C8 E5
 C6F0:D0 D0 AD 0F C0 D0 02 A9 B8
 C6F8:7A 91 FB AD 0E C0 91 FD 17
 C700:20 10 C5 20 1C C5 60 CE E5
 C708:15 C0 AD 0D C0 F0 54 38 83
 C710:A5 FB E9 01 85 45 A5 FC 48
 C718:E9 00 85 46 38 A5 FD E9 EF
 C720:01 85 47 A5 FE E9 00 85 F9
 C728:48 A0 00 B1 45 09 80 91 FF
 C730:45 A9 00 91 47 C8 CC 15 F1
 C738:C0 90 F0 F0 EE 38 A5 45 62
 C740:E9 28 85 45 A5 46 E9 00 ED
 C748:85 46 38 A5 47 E9 28 85 45
 C750:47 A5 48 E9 00 85 48 A0 DB
 C758:00 B1 45 09 80 91 45 A9 0C
 C760:00 91 47 EE 15 C0 60 AD 46
 C768:13 C0 C9 BC B0 12 18 69 1E
 C770:04 8D 13 C0 AD 14 C0 18 2B
 C778:69 04 8D 14 C0 EE 03 C0 39
 C780:A9 00 85 FB AD 88 02 85 6E
 C788:FC A9 00 85 FD AD 13 C0 E6
 C790:85 FE A9 00 85 45 A9 D8 45
 C798:85 46 A9 00 85 47 AD 14 6A
 C7A0:C0 85 48 A2 04 A0 00 B1 79
 C7A8:FB 91 FD B1 45 91 47 C8 3D
 C7B0:D0 F5 E6 FC E6 FE E6 46 1A


```

C7B8:E6 48 CA D0 EA 60 AD 13 7B
C7C0:C0 C9 A0 90 5C A9 00 AE 78
C7C8:88 02 85 FD 86 FE A9 00 31
C7D0:A2 D8 85 47 86 48 A9 00 B5
C7D8:AE 13 C0 85 FB 86 FC AE 97
C7E0:14 C0 85 45 86 46 A2 04 46
C7E8:A0 00 78 A9 34 85 01 B1 DD
C7F0:FB 91 FD B1 45 48 A9 37 93
C7F8:85 01 58 68 91 47 C8 D0 29
C800:E9 E6 FC E6 FE E6 46 E6 55
C808:48 CA D0 DE AD 13 C0 38 EB
C810:E9 04 8D 13 C0 AD 14 C0 20
C818:38 E9 04 8D 14 C0 CE 03 DD
C820:C0 60 AE 0B C0 CA CA 8E 06
C828:16 C0 AE 0A C0 CA CA CA FC
C830:8E 15 C0 20 FB C8 A9 00 BE
C838:8D 6B 03 AD 04 C0 85 FB D0
C840:8D 74 03 AD 05 C0 85 FC 24
C848:8D 75 03 20 15 C9 AD 6B F6
C850:03 F0 01 60 A2 00 8E 18 10
C858:C0 E8 8E 6C 03 AC 18 C0 D8
C860:CC 15 C0 B0 3A A0 00 B1 C6
C868:FB C9 0D D0 07 A2 00 86 62
C870:C7 4C D6 C8 48 20 D2 FF C8
C878:AE 11 C0 20 9D C4 68 C9 5A
C880:20 90 07 C9 80 B0 03 EE 7F
C888:18 C0 20 3E C9 AD 6B 03 1D
C890:F0 CB AD 74 03 8D 04 C0 A1
C898:AD 75 03 8D 05 C0 60 A0 24
C8A0:00 B1 FB C9 20 F0 1F 90 4E
C8A8:09 C9 80 B0 05 A9 9D 20 76
C8B0:D2 FF 20 2A C9 CE 18 C0 CC
C8B8:D0 E5 AE 15 C0 CA 20 3E 03
C8C0:C9 CA D0 FA F0 10 20 D2 8E
C8C8:FF A9 A0 EE 18 C0 AC 18 FC
C8D0:C0 CC 15 C0 90 F0 A9 00 40
C8D8:8D 18 C0 AE 6C 03 EC 16 99
C8E0:C0 90 06 EE 6B 03 4C 8A 31
C8E8:C8 EE 6C 03 A6 C7 A9 0D 0D
C8F0:20 D2 FF 86 C7 20 08 C9 48
C8F8:4C 8A C8 18 AE 08 C0 E8 ED
C900:AC 09 C0 C8 20 F0 FF 60 F5
C908:38 20 F0 FF 18 AC 09 C0 24
C910:C8 20 F0 FF 60 18 AD 04 F0
C918:C0 6D 06 C0 85 FD AD 05 B8
C920:C0 6D 07 C0 85 FE 20 51 16
C928:C9 60 38 A5 FB E9 01 85 29
C930:FB 8D 04 C0 A5 FC E9 00 A6
C938:85 FC 8D 05 C0 60 18 A5 2D

```

C940:FB 69 01 85 FB 8D 04 C0 83
C948:A5 FC 69 00 85 FC 8D 05 5B
C950:C0 A5 FB C5 FD A5 FC E5 EF
C958:FE 90 03 EE 6B 03 60 8D 94
C960:1B C0 EE 08 C0 EE 09 C0 A4
C968:AE 0B C0 CA CA 8E 0B C0 42
C970:AE 0A C0 CA CA 8E 0A C0 08
C978:20 D6 C4 AD 0B C0 8D 16 D1
C980:C0 AD 0A C0 8D 15 C0 AD 1D
C988:1C C0 F0 1C CE 1C C0 AD 50
C990:1B C0 C9 01 F0 29 C9 02 EC
C998:F0 71 C9 03 D0 03 4C 31 C6
C9A0:CA C9 04 D0 03 4C 5B CA 64
C9A8:AE 0A C0 E8 E8 8E 0A C0 13
C9B0:AE 0B C0 E8 E8 8E 0B C0 5D
C9B8:CE 08 C0 CE 09 C0 60 CE 95
C9C0:16 C0 A5 FB 85 45 A5 FC 8D
C9C8:85 46 A5 FD 85 47 A5 FE D8
C9D0:85 48 20 10 C5 20 1C C5 EA
C9D8:A2 00 A0 00 B1 FB 91 45 B7
C9E0:B1 FD 91 47 C8 CC 15 C0 D7
C9E8:D0 F2 20 10 C5 20 1C C5 53
C9F0:20 28 C5 20 34 C5 E8 EC D0
C9F8:16 C0 D0 DE A9 20 A0 00 DE
CA00:91 45 C8 CC 15 C0 D0 F8 DB
CA08:4C 78 C9 AE 16 C0 AC 15 28
CA10:C0 88 88 B1 FB 48 B1 FD B6
CA18:C8 91 FD 68 91 FB 88 D0 1B
CA20:F1 A9 20 91 FB 20 10 C5 7C
CA28:20 1C C5 CA D0 E0 4C 78 55
CA30:C9 20 85 CA AE 16 C0 CA 2A
CA38:AC 15 C0 88 B1 45 91 FB CB
CA40:B1 47 91 FD 88 10 F5 20 23
CA48:AD CA CA D0 EB AC 15 C0 CA
CA50:88 A9 20 91 FB 88 10 FB CF
CA58:4C 78 C9 CE 15 C0 AE 16 77
CA60:C0 A0 00 C8 B1 FB 48 B1 CA
CA68:FD 88 91 FD 68 91 FB C8 7B
CA70:CC 15 C0 D0 EE A9 20 91 C6
CA78:FB 20 10 C5 20 1C C5 CA 3A
CA80:D0 DF 4C 78 C9 AE 16 C0 7D
CA88:CA 20 10 C5 20 1C C5 CA B1
CA90:D0 F7 38 A5 FB E9 28 85 4B
CA98:45 A5 FC E9 00 85 46 38 53
CAA0:A5 FD E9 28 85 47 A5 FE DB
CAA8:E9 00 85 48 60 38 A5 FB 93
CAB0:E9 28 85 FB B0 02 C6 FC CD
CAB8:38 A5 FD E9 28 85 FD B0 36
CAC0:02 C6 FE 38 A5 45 E9 28 AA
CAC8:85 45 B0 02 C6 46 38 A5 0E

```

CAD0:47 E9 28 85 47 B0 02 C6 A9
CAD8:48 60 20 60 C4 20 82 C4 25
CAE0:20 67 C7 20 40 C5 20 C6 7B
CAE8:C5 20 22 C8 A2 00 BD 5D 28
CAF0:03 95 FB E8 E0 04 D0 F6 2B
CAF8:BD 5D 03 95 41 E8 E0 08 F5
CB00:D0 F6 BD 5D 03 85 14 E8 8A
CB08:BD 5D 03 85 15 A2 00 BD 7F
CB10:DE 03 95 D9 E8 E0 19 D0 F5
CB18:F6 18 AE 6F 03 AC 70 03 AC
CB20:20 F0 FF 60 A9 3D 85 FB 53
CB28:A9 CB 85 FC A0 00 B1 FB 6C
CB30:F0 0A 20 D2 FF C8 D0 F6 AF
CB38:E6 FC D0 F2 60 93 20 20 7D
CB40:20 20 20 57 49 4E 44 4F C4
CB48:57 20 57 49 5A 41 52 44 D3
CB50:20 A0 A0 20 A0 A0 A0 20 1F
CB58:A0 A0 A0 A0 A0 A0 A0 0D 5C
CB60:21 4F 50 45 4E 2C 53 24 A8
CB68:0D 21 43 4C 4F 53 45 0D 5B
CB70:21 44 49 4D 28 52 2C 43 CD
CB78:2C 57 2C 48 29 0D 21 44 0A
CB80:45 46 28 54 4F 4E 45 2C 01
CB88:33 44 2C 48 55 45 2C 42 2F
CB90:43 48 52 2C 49 43 48 52 23
CB98:2C 4C 52 41 54 45 2C 42 0A
CBA0:52 41 54 45 29 0D 20 5B A9
CBA8:54 4F 4E 45 20 20 31 3D 7D
CBB0:4F 4E 20 4F 3D 4F 46 46 76
CBB8:5D 20 5B 33 44 20 20 20 A8
CBC0:20 31 3D 4F 4E 20 30 3D E1
CBC8:4F 46 46 5D 0D 21 52 45 0F
CBD0:53 54 4F 52 45 28 53 43 EA
CBD8:52 45 45 4E 29 0D 21 43 7B
CBE0:4C 52 0D 21 41 47 41 49 D9
CBE8:4E 2C 53 24 0D 21 44 49 1E
CBF0:53 50 4C 41 59 2C 53 24 2A
CBF8:0D 21 42 4F 58 0D 21 50 26
CC00:52 49 4E 54 40 2C 52 2C A7
CC08:43 2C 53 24 0D 21 43 45 B3
CC10:4E 54 45 52 2C 52 2C 53 0A
CC18:24 0D 21 48 55 45 28 42 02
CC20:41 43 4B 47 52 4F 55 4E D1
CC28:44 2C 42 4F 52 44 45 52 AC
CC30:2C 54 45 58 54 29 0D 21 A5
CC38:54 4F 4E 45 2C 44 55 52 5D
CC40:41 54 49 4F 4E 0D 21 53 E9
CC48:41 56 45 0D 21 4C 4F 41 AB
CC50:44 0D 21 45 52 41 53 45 4B
CC58:2C 52 4F 57 0D 21 53 43 D2

```

```

CC60:52 4F 4C 4C 28 44 49 52 7C
CC68:45 43 54 49 4F 4E 2C 41 E1
CC70:4D 4F 55 4E 54 29 0D 20 95
CC78:5B 55 50 3D 31 20 52 49 EA
CC80:47 48 54 3D 32 20 44 4F 18
CC88:57 4E 3D 33 20 4C 45 46 3F
CC90:54 3D 34 5D 0D 21 4B 49 CC
CC98:4E 45 54 49 43 28 44 49 56
CCA0:52 45 43 54 49 4F 4E 2C B2
CCA8:4E 45 57 20 57 2C 4E 45 F4
CCB0:57 20 48 29 0D 21 4D 45 66
CCB8:4E 55 2C 4D 24 2C 4B 24 B5
CCC0:0D 00 2C 28 00 00 00 00 E8

```

Program 2. Window Demo

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

```

FB 10 IF TE=0 THEN TE=49152:LOAD"WINDOW WIZARD",8,
1
MD 20 SYS TE:PRINT"{CLR}";:!HUE(6,6,14):FORI=0TO23
MD 30 PRINT"{4 SPACES}WINDOW WIZARD{5 SPACES}WINDO
W WIZARD":NEXT
XF 40 PRINT"{4 SPACES}WINDOW WIZARD{5 SPACES}WINDO
W WIZARD{CYN}";:!RESTORE
GA 50 !DEF(1,0,1,0,32,50,50):!DIM(4,20,19,15)
QJ 60 A$="WELCOME TO THE WONDERFUL WORLD OF WINDOW
WIZARD"+CHR$(13)+CHR$(13)
CB 70 A$=A$+"[8]PREPARE TO BE BEWILDERED AND AMAZE
D BY THE POWER OF YOUR C-64...{CYN}"
PJ 80 A$=A$+CHR$(13)+CHR$(13)+" PRESS THE SPACE
{2 SPACES}BAR TO CONTINUE "
XX 90 !OPEN,A$
RH 100 GOSUB1110:!DIM(8,3,30,10)
RX 110 A$="WITH WIZARD, YOU CAN PRINT ANY MESSAGE
{SPACE}(UP TO 255 CHARACTERS) IN A "
QJ 120 A$=A$+"MAGIC WINDOW. AND WAIT TIL YOU SEE W
HAT CAN BE DONE WITH THE WINDOW"
CJ 130 !OPEN,A$
GB 140 GOSUB1110:!DEF(1,0,10,102,32):!DIM(14,12,25
,9)
BR 150 A$="{YEL}YOU CAN, FOR EXAMPLE, CHANGE EITHE
R THE COLOR OF THE WINDOW OR THE"
QK 160 A$=A$+" CHARACTER USED FOR THE BORDER OF TH
E WINDOW.{CYN}":!OPEN,A$
PF 170 GOSUB1110:!DEF(0,0,1,0,32,35,35):!DIM(1,18,
14,14)
PA 180 A$="AS YOU CAN HEAR, THE PRINTING CAN BE DO
NE WITH OR WITHOUT SOUND."
MR 190 !OPEN,A$

```

```

RK 200 GOSUB1110: !DEF(1,0,1,0,32,1,1): !DIM(10,5,25
,6)
MQ 210 A$="AND YOU CAN VARY THE SPEED BOTH FOR THE
WINDOW AND THE PRINTING."
KS 220 !OPEN,A$
HA 230 GOSUB1110: !DEF(1,1,1,0,32,25,25): !DIM(2,2,2
4,10)
QS 240 A$="NOTICE THE THREE DIMENSIONAL EFFECT OF
{SPACE}THIS WINDOW? THAT'S JUST ONE "
DE 250 A$=A$+"OF THE MANY EFFECTS AT YOUR COMMAND.
": !OPEN,A$
FS 260 GOSUB1110:A$="AND YOU CAN REUSE A WINDOW WH
ENEVER YOU WISH...": !AGAIN,A$
BM 270 GOSUB1110: !DEF(1,1,1,160,160,25,25): !DIM(6,
20,18,12)
QK 280 A$="{RVS}{WHT}YOU CAN CHANGE THE INTERIOR O
F THE WINDOW, TOO.{OFF}{CYN}": !OPEN,A$
JA 290 GOSUB1110: !DEF(1,0,0,0,32,5,50): !DIM(0,0,40
,25)
AJ 300 A$=CHR$(13)+"AND FOR THE REAL KICKER --"+CH
R$(13)+CHR$(13)
XC 310 A$=A$+"{CYN} VERY SLOWLY NOW, PRESS THE SPA
CE BAR EIGHT TIMES "
DX 320 A$=A$+"TO STEP {8}BACKWARDS{CYN} THROUGH TH
E WINDOWS. ": !OPEN,A$
MX 330 FOR I=1 TO 8
MG 340 GOSUB1110
GM 350 !CLOSE:NEXTI
EC 360 !DIM(6,15,22,12): !DEF(1,1,1,0,32,35,35)
AB 370 A$="OF COURSE, THERE ARE MANY MORE COMMANDS
AVAILABLE THROUGH WINDOW "
MD 380 A$=A$+"WIZARD. FOR EXAMPLE, PRESS THE SPACE
BAR TO '!SAVE' "
QB 390 A$=A$+"THE SCREEN...": !OPEN,A$
RA 400 GOSUB1110: !SAVE:A$="PRESS IT AGAIN TO CLEAR
THE SCREEN...": !AGAIN,A$
QA 410 GOSUB1110:PRINT"{CLR}"
AD 420 A$="AND NOW PRESS IT ONCE MORE TO '!LOAD' T
HE SCREEN WE SAVED BEFORE..."
JF 430 !AGAIN,A$
SM 440 GOSUB1110: !LOAD:FOR I=1TO500:NEXT
DD 450 A$="AND WE'RE RIGHT BACK WHERE WE STARTED F
ROM..."
HR 460 !AGAIN,A$:FORI=1TO2000:NEXT: !CLOSE
KC 470 !RESTORE: !DEF(0,1,1,0,32,10,20): !DIM(4,4,22
,12)
DD 480 A$="WHAT ARE SOME OF THE OTHER COMMANDS AVA
ILABLE?": !OPEN,A$
QQ 490 FORI=1TO2000:NEXT
RK 500 A$="WELL, YOU CAN USE WIZARD'S '!TONE' COMM
AND FOR YOUR OWN PURPOSES."

```

```
KQ 510 A$=A$+CHR$(13)+"PRESS THE SPACE BAR TO HEAR
      IT (PRESS [8]RETURN{CYN} TO MOVE ON)
HP 520 !AGAIN,A$
RC 530 GETK$:IFK$=""THEN530
BB 540 IFK$=CHR$(13) THEN570
XX 550 IFK$<>CHR$(32)THEN530
FK 560 !TONE,80:GOTO530
FC 570 A$="AND YOU HAVE TOTAL SCROLLING OF ANY ARE
      A OF THE SCREEN, AS WELL."
QM 580 A$=A$+"PRESS THE NUMBERS 1 TO 4 TO SEE WHAT
      I MEAN (OR PRESS [8]RETURN{CYN} TO"
FG 590 A$=A$+" MOVE ON).":!AGAIN,A$
BQ 600 GETK$:IFK$=""THEN600
BR 610 IFK$=CHR$(13)THEN660
FS 620 IFK$<"1"ORK$>"4"THEN600
PS 630 K=VAL(K$)
RX 640 T=1.2:IF K/2=INT(K/2) THEN T=22
KA 650 FORI=1TOT: !SCROLL,K,1:FORJ=1TO50:NEXT:NEXT:
      GOTO570
DJ 660 A$="{CYN}AND THE 'ERASE' COMMAND WILL ERAS
      E ANY ROW YOU DIRECT."+CHR$(13)
XH 670 A$=A$+"PRESS A NUMBER KEY AND WE'LL ERASE T
      HAT ROW..."
RF 680 A$=A$+" (OR PRESS [8]RETURN{CYN} TO MOVE ON
      )
KE 690 !AGAIN,A$
AE 700 K$="":GETK$:IFK$=""THEN700
DM 710 IFK$=CHR$(13)THEN740
DR 720 IFK$<"0"ORK$>"9"THEN700
ER 730 L=VAL(K$):!ERASE,L:GOTO700
GH 740 !CLOSE: !DIM(1.2,4,30,7)
XB 750 A$="AND NOW WE NEED TO MOVE OUR WINDOW DOWN
      HERE TO SHOW YOU THE NEXT"
JC 760 A$=A$+" WIZARD COMMAND...":!OPEN,A$:FORI=1T
      O1500:NEXT
HD 770 A$="LET'S CLEAR THE TOP OF THE SCREEN WITH
      {SPACE}OUR 'ERASE' COMMAND..."
FP 780 !AGAIN,A$:FOR I=0 TO 11:!ERASE,I:NEXT:FORI=
      1TO2000:NEXT
SJ 790 A$="AND NOW WE'LL EXAMINE YET ANOTHER COMMA
      ND...":!AGAIN,A$
SS 800 FORI=1TO2000:NEXT: !CENTER,3,"NOTICE HOW"
PJ 810 !CENTER,4,"THESE THREE LINES ARE ALL": !CENT
      ER,5,"CENTERED?"
QM 820 A$="IT WAS DONE WITH WIZARD'S ' !CENTER' COM
      MAND.": !AGAIN,A$
RD 830 FORI=1TO3000:NEXT: !CLOSE: !RESTORE: !DIM(0,1,
      38,5)
XQ 840 A$="YOU MAY DISCOVER THAT WIZARD'S ' !BOX' C
      OMMAND WILL COME IN HANDY..."
```

```

JQ 850 IOPEN,A$:FORI=1TO1500:NEXT:IDEF(0,0,0,0,32,
20,15):IDIM(7,1,18,10):IBOX
DJ 860 IDIM(7,21,18,10):IBOX:IDIM(18,1,12,6):IBOX:
IDIM(18,14,12,6):IBOX
GA 870 IDIM(18,27,12,6):IBOX:FORI=1TO2000:NEXT:ICL
OSE:IRESTORE
RE 880 IDEF(1,1,1,0,32,35,35):IDIM(1,20,19,8)
CE 890 A$="AND YOU CAN USE THE 'IKINETIC' COMMAND
{SPACE}TO MOVE WINDOWS...":IOPEN,A$
DJ 900 FORI=1TO1500:NEXT:FORI=1TO12:IKINETIC(3,0,0
):IKINETIC(4,0,0):NEXT
FA 910 A$="AND, USING THE SAME COMMAND, YOU CAN CO
NTROL THE SIZE OF ANY OF "
QA 920 A$=A$+"YOUR MAGIC WINDOWS...":IAGAIN,A$:FOR
I=1TO1500:NEXT
SR 930 FORI=1TO12:J=19+I:K=8+I:IKINETIC(1,J,K):NEX
T:IDIM(10,5,30,9)
JC 940 A$="THERE ARE ALSO COMMANDS LIKE 'IRESTORE'
,'IPRINT@', AND 'IDISPLAY'."
PF 950 A$=A$+CHR$(13)+CHR$(13)+"BUT THE LAST ONE W
E'LL DEMONSTRATE "
RK 960 A$=A$+"IS THE 'IMENU' COMMAND. IT'S A HANDY
ONE. ":IOPEN,A$
CF 970 FOR I=1TO1500:NEXT:R$=CHR$(13):M$="
{5 SPACES}MENU"+R$+R$
GH 980 M$=M$+"1.{2 SPACES}THIS IS AN"+R$:M$=M$+"2.
{2 SPACES}AN EXAMPLE OF"+R$+"3.{2 SPACES}A
{SPACE}WINDOW"+R$
DS 990 M$=M$+"4.{2 SPACES}GENERATED"+R$+"5.
{2 SPACES}MENU":M$=M$+R$+"6.{2 SPACES}PRESS
A KEY"
RB 1000 IDEF(1,1,0,0,32,10,30):IDIM(5,16,22,11):IM
ENU,M$,K$:IDIM(16,5,35,6)
QB 1010 IDEF(1,1,1,0,32,10,10)
SK 1020 A$="YOU JUST PRESSED THE '"+K$+"' KEY."+CH
R$(13)
KR 1030 IFK$<"1"ORR$>"6"THENA$=A$+"THAT WASN'T EVE
N ON THE MENU"
FS 1040 IOPEN,A$:FORI=1TO2000:NEXT:IDIM(3,1,20,20)
CE 1050 A$="F8]SYS 49152{CYN} ACTS AS A TOGGLE SWI
TCH. IF WIZARD IS OFF, IT WILL "
QC 1060 A$=A$+"TURN IT ON AND IF WIZARD IS ON, IT
{SPACE}WILL TURN IT OFF."+R$+R$
BF 1070 A$=A$+"TAPPING F8]RESTORE{CYN} WILL GIVE Y
OU A FULL PAGE REVIEW OF ALL 19 OF "
RA 1080 A$=A$+"WIZARD'S COMMANDS"+R$+R$
BG 1090 A$=A$+"F8]PRESS THE SPACE BAR TO END THIS
{SPACE}DEMONSTRATION.{CYN}":IOPEN,A$
CP 1100 GOSUB1110:PRINT"{CLR}":END
XM 1110 K$="":GETK$:IFK$=""THEN1110
HH 1120 RETURN

```

Fill-64

Hubert Cross

This exciting graphics utility adds 15 new commands to BASIC. "Fill-64" allows you to paint any solid shape on the hi-res graphics screen at a speed that's fast enough for animation. Once an animated sequence is complete, you can store the entire animation on disk or tape, to be replayed at a future time. No machine language knowledge is needed to use this program.

If you're interested in high-resolution graphics, you may have used programs that add commands such as PLOT and DRAW to BASIC. Most of these programs let you draw a closed shape on the screen with lines, then paint it with a command called something like FILL. Fill commands of this type often suffer from two drawbacks—they can't draw over existing dots on the screen, and they're much too slow for animation.

"Fill-64" lets you paint a solid shape anywhere on the hi-res screen, even over existing shapes. The program works so fast that it can even be used for animation. Since it adds new commands to BASIC, Fill-64 is easy enough for anyone to use. With simple commands—such as FILL, CLEAR, RECORD, and PLAY—you can draw complete animated sequences, store them on disk or tape, and redisplay them at your convenience.

Fill It Up

Fill-64 is written in machine language and must be typed in with "MLX," the machine language entry program found in Appendix D. However, no machine language expertise is required to use the program. Read the MLX instructions carefully before you type Program 1, and be sure to save a copy when you're done. Here are the addresses required for Fill-64:

Starting address: 83FF

Ending address: 9826

After you have finished entering all the data for Fill-64, be sure to save a copy to disk or tape before leaving MLX.

Before loading Fill-64, you must enter the following line in immediate mode (with no line numbers) and press RETURN.

POKE 55,255:POKE 56,63:NEW

This statement moves the top of BASIC memory down to a point 24,577 bytes lower than usual. That leaves 14K for BASIC programs, more than enough for most applications. The reserved 24K area is used to store Fill-64 and its data. *It's very important that you perform this step every time you wish to use Fill-64; if you don't, the program can't work properly.*

Next, load Fill-64 with **LOAD "PROGRAM",8,1** for disk or **LOAD "PROGRAM",1,1** for tape. Replace the name PROGRAM with the name you used when you saved the Fill-64 data with MLX. When the program has loaded, type **NEW** and press RETURN to reset the computer's BASIC pointers.

To activate Fill-64, type **SYS 33846** and press RETURN. Fill-64 makes two hi-res screens available at all times. When you first activate the program with SYS, it clears both screens and displays screen 1. Use the function keys to switch from one screen to the other. The f5 key displays screen 2. To return to screen 1, press f3. The f7 key clears the current screen, and f1 returns you to the normal text screen.

Fill-64 Commands

Fill-64 adds 15 new graphics commands to BASIC. A summary of each command follows. Later sections of this article contain programming examples and additional information about these commands.

FILL *number, color, x1, y1...,xnumber,ynumber*

The FILL command paints a polygon of the specified color at the designated screen location. The first parameter (*number*) indicates how many vertices (corners) the shape has. For example, a triangle has 3 vertices; a square has 4; and so on. Use a vertex number of 1 to draw a dot. A vertex value of 2 draws a line. The second parameter (*color*) sets the shape's color, using the same color numbers explained in your user's manual. After these values you must supply an appropriate number of *x* (horizontal) and *y* (vertical) coordinate pairs. To draw a triangle, for instance, you need three pairs of *x,y* coordinates, one pair to define each corner. This statement, for instance, draws a triangle.

FILL 3,2,10,10,50,10,30,50

Notice that the FILL command automatically displays the

hi-res screen. No special command is needed to switch from the text screen to hi-res.

SWITCH

SWITCH moves from one hi-res screen to the other. If you're in screen 1, SWITCH flips you to screen 2, and vice versa. SWITCH ordinarily erases the screen that it previously displayed. If you perform SWITCH from the text screen, Fill-64 displays screen 1 without erasing its contents.

FLIP

FLIP works exactly like SWITCH but does not erase the previous screen.

CLEAR

This command erases both hi-res screens. It's useful at the beginning of a program.

LOWRES

The LOWRES statement switches to the text screen under program control. In immediate mode (when you're not running a program) you can also do this by pressing the f1 key.

SETCOLOR *border, background, hi-res1, hi-res2, hi-res3*

Sets the screen border, background, and three hi-res colors. The default colors for Fill-64 are white, blue, black, yellow, and white, respectively.

RECORD

This command records animated graphics sequences. It tells Fill-64 to "memorize" every graphics command that follows, until it encounters a REWIND or PLAY statement (see below). As a reminder, Fill-64 sets the border color to red whenever RECORD is active.

PLAY

Replays an animation previously recorded with RECORD or stored with DPUT or CPUT (see below). When Fill-64 is PLAYing an animation, it can be interrupted only by pressing RUN/STOP-RESTORE.

REWIND

Exits RECORD mode. In effect, this command "rewinds" Fill-64's recorder to the beginning, erasing any previously recorded animation.

MEMORY

Prints the number of bytes available for RECORDing animations.

DPUT "filename"

Saves an animation to disk. Replace *filename* with any legal Commodore filename.

DGET "filename"

Loads a previously saved animation from disk.

CPUT "filename"

Saves an animation to tape (the filename is optional).

CGET "filename"

Loads an animation from tape (the filename is optional).

Fill-64 Techniques

Program 2, "Animation Demo," demonstrates just a few of the effects you can achieve with Fill-64, from detailed kaleidoscopic patterns to three-dimensional animation. Program 3, "Animation Player," loads and replays the animations created by Program 2, using the full speed of Fill-64's built-in machine language routines.

Note: Since these programs use Fill-64 BASIC commands, you must install Fill-64 as described above before you type them in (naturally, you must also install Fill-64 whenever you wish to run these programs). Once Fill-64 is active, press f1 to return to the text screen; then enter Programs 2 and 3 as you would any BASIC program.

Program 2 creates several different animations. For each animated sequence, the program switches into RECORD mode, then performs a series of FILL commands. Then it replays the animation with PLAY and optionally saves the animation. When you run Program 2, it asks whether you wish to save the animations. Answer yes (enter Y) the first time you run the program, and be patient—some of the more complex shapes take considerable time to calculate. Notice how much faster the animation proceeds when replayed with PLAY. When Program 2 has finished, you can run Program 3 to load and replay all the animations in rapid sequence. Together, these two programs demonstrate the complete process of drawing, recording, reloading, and replaying an animation.

Fill-64 has two limitations when painting. First, you can draw any solid polygon with as many as 18 vertices, but the polygon must be convex—that is, its inside angles must be equal to, or less than, 90 degrees. To draw a shape containing obtuse angles (angles greater than 90 degrees), break it down into two or more convex polygons. For instance, a star shape can be built from two triangles, one placed upside down over the other.

Second, although you may start defining the polygon at any vertex, you must proceed in a clockwise direction. The FILL command ignores any vertex written in a counterclockwise direction. This feature is extremely useful for eliminating hidden surfaces in three-dimensional objects.

Let's look at a simple example. This command paints a small triangle in color 2 (yellow):

FILL 3,2,10,10,50,10,30,50

The first value (3) indicates the number of vertices in the shape, and the second (2) indicates the color. The remaining values contain the coordinates for the triangle. This program shows how SWITCH flips from one hi-res screen to the other.

```
100 FILL 3,2,10,10,50,10,30,50
110 SWITCH
120 FILL 3,2,11,10,51,10,31,50
130 SWITCH
```

The triangle moves one pixel to the right. When an animation requires more than a few frames, you'll probably want to use a program loop. This example moves the triangle all the way across the screen with a FOR-NEXT loop:

```
100 FOR X=50 TO 150
110 FILL 3,2,10+X,10,50+X,10,30+X,50
120 SWITCH:NEXT:LOWRES
```

Notice that the triangle moves off the screen without painting any unwanted areas. Fill-64 automatically clips the edge of any shape that moves across a screen border. It faithfully paints only the visible portion of a shape as long as its coordinates don't exceed the range -32768-32767.

Program 1. Fill-64

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```

83FF:FF 4C 24 88 4C 92 87 4C 2F
8407:78 88 4C 93 88 4C 49 8F C8
840F:4C 86 84 4C 47 88 4C 65 8F
8417:88 4C FB 88 4C FF 88 4C 3F
841F:C9 88 4C CF 88 4C E1 88 77
8427:4C F3 88 4C F7 88 4C F0 94
842F:88 4C F1 88 4C F2 88 A9 3F
8437:8C 8D 04 03 A9 84 8D 05 1A
843F:03 A9 51 8D 06 03 A9 85 4C
8447:8D 07 03 A9 ED 8D 08 03 8C
844F:A9 85 8D 09 03 A9 00 8D 1D
8457:DF 9F 8D DE 9F 8D 17 9F D8
845F:8D 05 40 A9 00 8D 1F 9F 27
8467:A9 40 8D 20 9F A9 01 8D 3C
846F:00 40 8D 04 40 A9 06 8D BC
8477:01 40 A9 00 8D 02 40 A9 E4
847F:07 8D 03 40 4C A9 8E 20 1A
8487:F5 96 4C 61 97 A6 7A A0 BD
848F:04 84 0F BD 00 02 10 07 A8
8497:C9 FF F0 3E E8 D0 F4 C9 C5
849F:20 F0 37 85 08 C9 22 F0 D0
84A7:56 24 0F 70 2D C9 3F D0 AD
84AF:04 A9 99 D0 25 C9 30 90 A6
84B7:04 C9 3C 90 1D 84 71 A0 44
84BF:00 84 0B 88 86 7A CA C8 50
84C7:E8 BD 00 02 38 F9 9E A0 5C
84CF:F0 F5 C9 80 D0 30 05 0B 6C
84D7:A4 71 E8 C8 99 FB 01 B9 B1
84DF:FB 01 F0 38 38 E9 3A F0 97
84E7:04 C9 49 D0 02 85 0F 38 18
84EF:E9 55 D0 9F 85 08 BD 00 1F
84F7:02 F0 DF C5 08 F0 DB C8 1B
84FF:99 FB 01 E8 D0 F0 A6 7A 95
8507:E6 0B C8 B9 9D A0 10 FA 87
850F:B9 9E A0 D0 B4 F0 0F BD 05
8517:00 02 10 BC 99 FD 01 C6 FD
851F:7B A9 FF 85 7A 60 A0 00 41
8527:B9 87 85 D0 02 E8 C8 BD B1
852F:00 02 38 F9 87 85 F0 F5 8B
8537:C9 80 D0 04 05 0B D0 98 30
853F:A6 7A E6 0B C8 B9 86 85 89
8547:10 FA B9 87 85 D0 E0 4C 46
854F:16 85 10 0F C9 FF F0 0B F4
8557:24 0F 30 07 C9 CC B0 06 97
855F:4C 24 A7 4C F3 A6 38 E9 E7
8567:CB AA 84 49 A0 FF CA F0 B3
856F:09 C8 B9 87 85 10 FA 4C 8F

```

```

8577:6D 85 C8 B9 87 85 30 05 07
857F:20 47 AB D0 F5 4C EF A6 56
8587:53 45 54 43 4F 4C 4F D2 69
858F:46 49 4C CC 53 57 49 54 45
8597:43 C8 46 4C 49 D0 4C 4F 79
859F:57 52 45 D3 43 4C 45 41 E7
85A7:D2 52 45 43 4F 52 C4 50 2B
85AF:4C 41 D9 44 50 55 D4 44 76
85B7:47 45 D4 52 45 57 49 4E DF
85BF:C4 45 52 41 53 C5 4D 45 6E
85C7:4D 4F 52 D9 43 50 55 D4 10
85CF:43 47 45 D4 54 55 52 42 23
85D7:4F 50 55 D4 54 55 52 42 75
85DF:4F 47 45 D4 52 45 4C 4F E9
85E7:43 41 54 C5 00 00 20 73 7F
85EF:00 20 F6 85 4C AE A7 C9 70
85F7:CC 90 04 C9 DE 90 06 20 10
85FF:79 00 4C ED A7 38 E9 CC EE
8607:0A AA BD 15 86 48 BD 14 B1
860F:86 48 4C 73 00 FF 83 02 3B
8617:84 05 84 08 84 0B 84 0E 20
861F:84 11 84 14 84 17 84 1A 28
8627:84 1D 84 20 84 23 84 26 30
862F:84 29 84 2C 84 2F 84 32 38
8637:84 AD 00 9D D0 01 60 C9 E0
863F:03 10 0C AD 01 9D 2D DF E6
8647:9F D0 03 4C B4 86 60 A0 9E
864F:00 20 7E 86 A0 09 20 94 9A
8657:86 20 89 8D 20 B0 8C A0 37
865F:03 20 7E 86 A0 06 20 94 20
8667:86 20 89 8D 20 BC 8C A9 80
866F:00 8D DF 9F AD 0B 9F 30 DE
8677:3C A9 80 8D DF 9F 60 20 54
867F:AA 86 AD 08 9F 8D 1C 9F C4
8687:AD 09 9F 8D 1D 9F AD 0A 47
868F:9F 8D 1E 9F 60 20 AA 86 EC
8697:AD 1C 9F 8D 02 9F AD 1D 56
869F:9F 8D 03 9F AD 1E 9F 8D EC
86A7:04 9F 60 20 E4 8B 20 0A 4C
86AF:8C 20 56 8C 60 AD 00 9D F5
86B7:C9 02 10 24 AD 02 9D 0D 2C
86BF:04 9D 0D 05 9D 0D 07 9D F4
86C7:D0 EA AD 03 9D 38 E9 A0 20
86CF:B0 E2 AD 06 9D 38 E9 C8 6E
86D7:B0 DA 20 04 88 4C 29 87 87
86DF:C9 03 10 39 20 51 89 20 A1
86E7:04 88 AD 00 9D C9 02 10 F6
86EF:01 60 F0 21 AD 02 9D CD 44
86F7:04 9D D0 14 AD 03 9D CD 4C
86FF:05 9D D0 0C AD 06 9D 8D 20

```

```

8707:04 9D AD 07 9D 8D 05 9D 70
870F:A9 02 8D 00 9D A9 00 8D 46
8717:06 9D 4C 29 87 20 51 89 95
871F:AD 00 9D C9 03 90 18 20 FF
8727:04 88 AD 17 9F F0 03 20 68
872F:3F 87 A9 00 8D 20 98 A9 BC
8737:9D 8D 21 98 20 30 90 60 69
873F:AD 00 9D 0A 69 06 6D 18 CF
8747:9F A9 00 6D 19 9F C9 80 C2
874F:90 06 20 49 8F 4C 35 A4 7D
8757:A9 02 8D 20 D0 AD 18 9F 7C
875F:85 FB AD 19 9F 85 FC AD 32
8767:00 9D 0A 69 02 A8 18 65 FD
876F:FB 8D 18 9F A9 00 65 FC F1
8777:8D 19 9F C8 98 18 65 FB FF
877F:8D 1F 9F A9 00 65 FC 8D C8
8787:20 9F B9 00 9D 91 FB 88 79
878F:10 F8 60 20 8A AD 20 9B D9
8797:BC A5 65 8D 00 9D 20 FD A8
879F:AE 20 8A AD 20 9B BC A5 C8
87A7:65 8D 01 9D A9 00 8D 1B 4A
87AF:9F 8D 1A 9F 20 FD AE 20 A5
87B7:8A AD 20 9B BC 20 CE 87 C0
87BF:EE 1A 9F AD 1A 9F 4A CD 4D
87C7:00 9D 90 E8 4C 38 86 AC DB
87CF:1B 9F A9 00 99 02 9D A5 3F
87D7:65 99 03 9D A5 64 99 04 30
87DF:9D 10 1A A9 FF 99 02 9D A7
87E7:B9 7F 3F 7F 01 41 69 01 74
87EF:99 03 9D B9 04 9D 49 FF 05
87F7:69 00 99 04 9D 98 18 69 18
87FF:03 8D 1B 9F 60 AD 00 9D A8
8807:0A 8D 1B 9F A2 01 A0 00 38
880F:BD 02 9D 99 02 9D E8 E8 0E
8817:E8 C8 CC 1B 9F 90 F1 A9 E6
881F:00 99 02 9D 60 A9 00 8D E7
8827:1B 9F 20 8A AD 20 9B BC 3C
882F:A5 65 AE 1B 9F EE 1B 9F 82
8837:9D 00 40 E0 04 10 06 20 B9
883F:FD AE 4C 29 88 4C 78 8F 0D
8847:A9 80 8D 17 9F A9 02 8D A5
884F:20 D0 A9 05 8D 18 9F A9 DF
8857:40 8D 19 9F 60 A9 00 8D 40
885F:17 9F 20 78 8F 60 20 5C 0A
8867:88 A9 05 8D 20 98 A9 40 97
886F:8D 21 98 20 51 88 4C 30 1A
8877:90 AD 17 9F 10 13 A9 80 B9
887F:20 AE 88 AD 18 9F 18 69 11
8887:01 8D 18 9F 90 03 EE 19 01
888F:9F 4C E3 8F AD 17 9F 10 12

```

8897:13 A9 40 20 AE 88 AD 18 B1
889F:9F 18 69 01 8D 18 9F 90 60
88A7:03 EE 19 9F 4C 0E 90 AC 7B
88AF:18 9F 84 FB AC 19 9F 84 92
88B7:FC A0 00 91 FB 98 C8 91 ED
88BF:FB EE 1F 9F D0 03 EE 20 F8
88C7:9F 60 20 5C 88 4C 51 88 2B
88CF:A9 00 8D 05 40 A9 00 8D ED
88D7:1F 9F A9 40 8D 20 9F 4C 12
88DF:C9 88 A9 00 38 ED 1F 9F 84
88E7:AA A9 80 ED 20 9F 4C CD 8D
88EF:BD 60 60 60 A9 01 D0 0C 09
88F7:A9 01 D0 2A A9 08 D0 04 ED
88FF:A9 08 D0 22 AA A0 01 20 1E
8907:BA FF 20 43 89 A9 00 85 28
890F:FB A9 40 85 FC A9 FB AE 20
8917:1F 9F AC 20 9F 20 D8 FF 68
891F:90 03 4C F9 E0 60 AA A0 E2
8927:01 20 BA FF 20 43 89 A9 E4
892F:00 A2 00 A0 40 20 D5 FF 23
8937:90 03 4C D1 E1 8E 1F 9F 21
893F:8C 20 9F 60 A9 00 20 BD E5
8947:FF 20 06 E2 20 57 E2 4C C1
894F:BD FF A9 00 8D 01 9F 20 46
8957:58 8A A9 A0 8D 00 9F 20 44
895F:72 89 A9 C8 8D 01 9F 20 9F
8967:58 8A A9 00 8D 00 9F 20 4A
896F:72 89 60 20 71 8D A0 00 2F
8977:8C 00 9E AD 01 9D 8D 01 1A
897F:9E 20 A0 8B 20 0A 8C 20 19
8987:56 8C AD 0A 9F 0D 09 9F 22
898F:F0 4B AD 08 9F 8D 16 9F 23
8997:20 30 8C 20 56 8C AD 0A A4
899F:9F 0D 09 9F F0 37 AD 08 A8
89A7:9F CD 16 9F F0 2F 20 6F AE
89AF:8B AD 00 9E 0A 6D 00 9E 82
89B7:0A AA AD 02 9F 9D 05 9E 6C
89BF:AD 03 9F 9D 06 9E AD 04 42
89C7:9F 9D 07 9E A9 00 9D 02 67
89CF:9E 9D 04 9E AD 00 9F 9D 4E
89D7:03 9E EE 00 9E AD 00 9F 3D
89DF:D0 08 B9 08 9D D0 46 4C 1E
89E7:FC 89 B9 08 9D D0 0E B9 99
89EF:0A 9D D0 39 AD 00 9F 38 02
89F7:F9 09 9D 90 30 AD 00 9E DD
89FF:0A 6D 00 9E 0A AA B9 0B D6
8A07:9D 9D 05 9E B9 0C 9D 9D B3
8A0F:06 9E B9 0D 9D 9D 07 9E E6
8A17:B9 08 9D 9D 02 9E B9 09 9F
8A1F:9D 9D 03 9E B9 0A 9D 9D 83


```

8A27:04 9E EE 00 9E AD 00 9D 0D
8A2F:0A 6D 00 9D 0A 38 E9 06 89
8A37:8D 16 9F 98 38 ED 16 9F 5B
8A3F:B0 08 98 18 69 06 A8 4C 44
8A47:80 89 AD 00 9E D0 06 8D 86
8A4F:00 9D 68 68 60 20 5C 8D 29
8A57:60 20 71 8D A0 00 8C 00 C9
8A5F:9E AD 01 9D 8D 01 9E 20 F6
8A67:AF 8B 20 1D 8C 20 56 8C 2B
8A6F:AD 0A 9F 0D 09 9F F0 4B 96
8A77:AD 08 9F 8D 16 9F 20 43 E4
8A7F:8C 20 56 8C AD 0A 9F 0D 58
8A87:09 9F F0 37 AD 08 9F CD 35
8A8F:16 9F F0 2F 20 3E 8B AD 67
8A97:00 9E 0A 6D 00 9E 0A AA A5
8A9F:AD 02 9F 9D 02 9E AD 03 C2
8AA7:9F 9D 03 9E AD 04 9F 9D 98
8AAF:04 9E A9 00 9D 05 9E 9D 7F
8AB7:07 9E AD 01 9F 9D 06 9E DB
8ABF:EE 00 9E AD 01 9F D0 08 2B
8AC7:B9 0B 9D D0 46 4C E2 8A F0
8ACF:B9 0B 9D D0 0E B9 0D 9D 54
8AD7:D0 39 AD 01 9F 38 F9 0C 47
8ADF:9D 90 30 AD 00 9E 0A 6D C4
8AE7:00 9E 0A AA B9 08 9D 9D 57
8AEF:02 9E B9 09 9D 9D 03 9E 7D
8AF7:B9 0A 9D 9D 04 9E B9 0B 13
8AFF:9D 9D 05 9E B9 0C 9D 9D AC
8B07:06 9E B9 0D 9D 9D 07 9E E0
8B0F:EE 00 9E AD 00 9D 0A 6D 44
8B17:00 9D 0A 38 E9 06 8D 16 F2
8B1F:9F 98 38 ED 16 9F B0 08 AA
8B27:98 18 69 06 A8 4C 66 8A EB
8B2F:AD 00 9E D0 06 8D 00 9D 02
8B37:68 68 60 20 5C 8D 60 20 A4
8B3F:AF 8B 20 43 8C 20 56 8C 67
8B47:20 BE 8B 20 89 8D 20 B0 05
8B4F:8C 20 AF 8B 20 1D 8C 20 12
8B57:56 8C 20 E4 8B 20 89 8D 8C
8B5F:20 BC 8C 20 F7 8B 20 1D 94
8B67:8C 20 56 8C 20 E4 8D 60 70
8B6F:20 A0 8B 20 30 8C 20 56 7C
8B77:8C 20 D1 8B 20 89 8D 20 32
8B7F:B0 8C 20 A0 8B 20 0A 8C 9D
8B87:20 56 8C 20 F7 8B 20 89 8F
8B8F:8D 20 BC 8C 20 E4 8B 20 A1
8B97:0A 8C 20 56 8C 20 E4 8D 7C
8B9F:60 A9 00 8D 02 9F 8D 04 D7
8BA7:9F AD 00 9F 8D 03 9F 60 0C
8BAF:A9 00 8D 02 9F 8D 04 9F 48

```

8BB7:AD 01 9F 8D 03 9F 60 B9 C3
8BBF:02 9D 8D 02 9F B9 03 9D 98
8BC7:8D 03 9F B9 04 9D 8D 04 AB
8BCF:9F 60 B9 05 9D 8D 02 9F 1D
8BD7:B9 06 9D 8D 03 9F B9 07 EA
8BDF:9D 8D 04 9F 60 B9 08 9D 3B
8BE7:8D 02 9F B9 09 9D 8D 03 B2
8BEF:9F B9 0A 9D 8D 04 9F 60 7C
8BF7:B9 0B 9D 8D 02 9F B9 0C 49
8BFF:9D 8D 03 9F B9 0D 9D 8D 6E
8C07:04 9F 60 B9 02 9D 8D 05 58
8C0F:9F B9 03 9D 8D 06 9F B9 1E
8C17:04 9D 8D 07 9F 60 B9 05 B2
8C1F:9D 8D 05 9F B9 06 9D 8D B3
8C27:06 9F B9 07 9D 8D 07 9F A3
8C2F:60 B9 08 9D 8D 05 9F B9 3B
8C37:09 9D 8D 06 9F B9 0A 9D E3
8C3F:8D 07 9F 60 B9 0B 9D 8D 9D
8C47:05 9F B9 0C 9D 8D 06 9F 91
8C4F:B9 0D 9D 8D 07 9F 60 AD 3A
8C57:05 9F 49 FF 8D 05 9F CD 91
8C5F:02 9F D0 17 8D 08 9F AD 66
8C67:03 9F 18 6D 06 9F 8D 09 96
8C6F:9F AD 04 9F 6D 07 9F 8D 92
8C77:0A 9F 60 AD 03 9F 38 ED 59
8C7F:06 9F 8D 09 9F AD 04 9F 21
8C87:ED 07 9F 90 0A 8D 0A 9F 90
8C8F:AD 02 9F 8D 08 9F 60 AD F9
8C97:06 9F 38 ED 03 9F 8D 09 3C
8C9F:9F AD 07 9F ED 04 9F 8D 1B
8CA7:0A 9F AD 05 9F 8D 08 9F 96
8CAF:60 A2 04 BD 0B 9F 9D 11 21
8CB7:9F CA 10 F7 60 AD 11 9F 50
8CBF:49 FF 8D 11 9F CD 0B 9F 2A
8CC7:D0 26 AD 0C 9F 18 6D 12 93
8CCF:9F 8D 0C 9F AD 0D 9F 6D E5
8CD7:13 9F 8D 0D 9F AD 0E 9F 54
8CDF:6D 14 9F 8D 0E 9F AD 0F DA
8CE7:9F 6D 15 9F 8D 0F 9F 60 11
8CEF:AD 0C 9F 38 ED 12 9F AD FE
8CF7:0D 9F ED 13 9F AD 0E 9F DD
8CFF:ED 14 9F AD 0F 9F ED 15 CB
8D07:9F 90 26 AD 0C 9F 38 ED F2
8D0F:12 9F 8D 0C 9F AD 0D 9F FA
8D17:ED 13 9F 8D 0D 9F AD 0E 0B
8D1F:9F ED 14 9F 8D 0E 9F AD 93
8D27:0F 9F ED 15 9F 8D 0F 9F B1
8D2F:60 AD 12 9F 38 ED 0C 9F 53
8D37:8D 0C 9F AD 13 9F ED 0D EA
8D3F:9F 8D 0D 9F AD 14 9F ED 14

```

8D47:0E 9F 8D 0E 9F AD 15 9F 61
8D4F:ED 0F 9F 8D 0F 9F AD 11 55
8D57:9F 8D 0B 9F 60 AD 00 9E 59
8D5F:0A 6D 00 9E 0A 69 02 AA 69
8D67:BD 00 9E 9D 00 9D CA 10 2B
8D6F:F7 60 AD 00 9D 0A 6D 00 44
8D77:9D 0A 69 05 A8 A2 05 BD F8
8D7F:02 9D 99 02 9D 88 CA 10 0B
8D87:F6 60 A9 00 8D 0C 9F 8D D4
8D8F:0D 9F 8D 0E 9F 8D 0F 9F 9C
8D97:8D 0B 9F AD 08 9F CD 02 67
8D9F:9F F0 05 A9 FF 8D 0B 9F ED
8DA7:A2 10 0E 0C 9F 2E 0D 9F 0A
8DAF:2E 0E 9F 2E 0F 9F 0E 03 52
8DB7:9F 2E 04 9F 90 23 AD 0C 21
8DBF:9F 18 6D 09 9F 8D 0C 9F D9
8DC7:AD 0D 9F 6D 0A 9F 8D 0D BE
8DCF:9F AD 0E 9F 69 00 8D 0E 56
8DD7:9F AD 0F 9F 69 00 8D 0F 7F
8DDF:9F CA D0 C6 60 AD 09 9F 6F
8DE7:0D 0A 9F D0 06 A9 02 8D 75
8DEF:20 D0 60 AD 0C 9F 0D 0D 3C
8DF7:9F 0D 0E 9F 0D 0F 9F D0 96
8DFF:0C A9 00 8D 02 9F 8D 03 11
8E07:9F 8D 04 9F 60 A9 FF 8D 09
8E0F:02 9F AD 0B 9F CD 08 9F 5F
8E17:D0 05 A9 00 8D 02 9F A9 70
8E1F:00 8D 03 9F 8D 04 9F 8D 43
8E27:10 9F CE 10 9F 0E 0C 9F FB
8E2F:2E 0D 9F 2E 0E 9F 2E 0F D7
8E37:9F 10 EF AD 0A 9F 10 03 F2
8E3F:4C 4D 8E EE 10 9F 0E 09 BA
8E47:9F 2E 0A 9F 10 F5 4E 0F FE
8E4F:9F 6E 0E 9F 6E 0D 9F 6E E8
8E57:0C 9F 4E 0A 9F 6E 09 9F 35
8E5F:A2 10 AD 0E 9F 38 ED 09 2B
8E67:9F 8D 16 9F AD 0F 9F ED 4B
8E6F:0A 9F 90 0A 8D 0F 9F AD C1
8E77:16 9F 8D 0E 9F 38 2E 03 57
8E7F:9F 2E 04 9F 0E 0C 9F 2E 80
8E87:0D 9F 2E 0E 9F 2E 0F 9F 2D
8E8F:CA D0 CF AD 10 9F 30 03 7D
8E97:4C EC 8D EE 10 9F 10 09 DE
8E9F:4E 04 9F 6E 03 9F 4C 9A 89
8EA7:8E 60 78 A9 C6 8D 14 03 65
8EAF:A9 8E 8D 15 03 58 A9 00 15
8EB7:8D DD 9F 20 78 8F 20 5E A9
8EBF:8F A9 A0 8D ED 9F 60 48 F2
8EC7:A5 C5 C9 03 30 0E C9 07 E6
8ECF:10 0A AD DD 9F C9 00 F0 20

```

8ED7:0C 4C E0 8E A9 00 8D DD 59
8EDF:9F 68 4C 31 EA A9 3F 8D 8D
8EE7:02 DD A5 C5 C9 04 D0 06 94
8EEF:20 49 8F 4C 3B 8F C9 05 D6
8EF7:D0 06 20 5E 8F 4C 3B 8F 9C
8EFF:C9 06 D0 06 20 73 8F 4C 38
8F07:3B 8F AD 00 DD C9 97 F0 93
8F0F:2B 8A 48 98 48 A5 FB 48 12
8F17:A5 FC 48 AD ED 9F 48 AD 58
8F1F:00 DD C9 94 F0 1F A9 A0 30
8F27:8D ED 9F 20 DF 96 68 8D 36
8F2F:ED 9F 68 85 FC 68 85 FB 23
8F37:68 A8 68 AA A9 01 8D DD B6
8F3F:9F 68 4C 31 EA A9 E0 4C F0
8F47:27 8F A9 97 8D 00 DD A9 5E
8F4F:C8 8D 16 D0 A9 1B 8D 11 EB
8F57:D0 A9 15 8D 18 D0 60 A9 33
8F5F:95 8D 00 DD A9 D8 8D 16 6C
8F67:D0 A9 3B 8D 11 D0 A9 08 C0
8F6F:8D 18 D0 60 A9 94 4C 60 14
8F77:8F A9 02 AE 17 9F E0 80 6D
8F7F:F0 03 AD 00 40 8D 20 D0 D6
8F87:AD 01 40 8D 21 D0 A2 83 B3
8F8F:A9 E8 8D DC 9F A9 80 85 67
8F97:FC A9 00 85 FB AD 02 40 D2
8F9F:0A 0A 0A 0A 0D 03 40 8D AA
8FA7:D9 9F 20 C6 96 A2 C3 A9 7C
8FAF:E8 8D DC 9F A9 C0 85 FC 94
8FB7:A9 00 85 FB AD D9 9F 20 50
8FBF:C6 96 A2 DB A9 E8 8D DC E2
8FC7:9F A9 D8 85 FC A9 00 85 A8
8FCF:FB AD 04 40 8D 86 02 20 87
8FD7:C6 96 60 20 5E 8F A9 A0 33
8FDF:8D ED 9F 60 AD 00 DD 29 8D
8FE7:FE C9 94 F0 06 20 DA 8F 90
8FEF:4C DF 96 AD ED 9F C9 E0 3D
8FF7:F0 07 A0 95 A9 E0 4C 04 2C
8FFF:90 A0 94 A9 A0 8C 00 DD D1
9007:8D ED 9F 20 DF 96 60 AD 28
900F:00 DD 29 FE C9 94 D0 C3 C2
9017:AD ED 9F C9 E0 F0 07 A0 94
901F:95 A9 E0 4C 29 90 A0 94 B7
9027:A9 A0 8C 00 DD 8D ED 9F 77
902F:60 AD 00 DD C9 94 F0 0A 56
9037:C9 95 F0 06 20 A9 8E 20 06
903F:DF 96 AD 20 98 8D CD 9F E3
9047:AD 21 98 8D CC 9F A9 34 DF
904F:8D E1 9F A9 FB 8D E0 9F B5
9057:A9 00 85 FB A9 03 85 FC 1F
905F:AD CD 9F 85 FD AD CC 9F F6

```
9067:85 FE A0 01 B1 FD 8D DE AE
906F:9F 29 03 8D DA 9F 20 76 EF
9077:96 88 B1 FD 8D E7 9F 0E 75
907F:E7 9F AC E7 9F C9 00 D0 85
9087:01 60 10 14 20 E3 8F AD E1
908F:CD 9F 18 69 01 8D CD 9F 92
9097:90 BE EE CC 9F 4C 57 90 C8
909F:C9 40 30 06 20 0E 90 4C C2
90A7:8E 90 C9 01 D0 03 4C 35 DD
90AF:95 C9 02 D0 03 4C 73 93 1F
90B7:AD CD 9F 18 6D E7 9F 90 73
90BF:03 EE CC 9F 18 69 02 8D A9
90C7:CD 9F 90 03 EE CC 9F A5 89
90CF:FD 18 69 01 85 FD 90 02 7A
90D7:E6 FE B1 FD 8D E6 9F 8C 16
90DF:E4 9F 88 88 B1 FD 38 ED D8
90E7:E6 9F B0 08 B1 FD 8D E6 82
90EF:9F 8C E4 9F 88 88 D0 EC 8F
90F7:AC E7 9F B1 FD 8D E5 9F 0A
90FF:8C E3 9F 88 88 B1 FD 38 1C
9107:ED E5 9F 90 08 B1 FD 8D 28
910F:E5 9F 8C E3 9F 88 88 D0 DD
9117:EC AC E4 9F 8C E2 9F A5 47
911F:FD 38 E9 01 85 FD B0 02 24
9127:C6 FE A2 02 AC E2 9F B1 C3
912F:FD EE E2 9F AC E1 9F 91 21
9137:FB EE E1 9F CA D0 ED CD 8D
913F:E5 9F F0 14 AD E2 9F 38 0D
9147:E9 04 8D E2 9F D0 DB AD E5
914F:E7 9F 8D E2 9F 4C 29 91 40
9157:AC E1 9F A9 FF 91 FB AC C2
915F:E4 9F 8C E2 9F A2 02 AC D4
9167:E2 9F B1 FD EE E2 9F AC E8
916F:E0 9F 91 FB CE E0 9F CA E0
9177:D0 ED CD E5 9F F0 10 AD 25
917F:E7 9F CD E2 9F 10 DE A9 0B
9187:02 8D E2 9F 4C 64 91 A0 1D
918F:00 84 FB A0 34 8C E1 9F 94
9197:A0 FB 8C E0 9F A0 03 84 B3
919F:FC 20 F3 95 AD FE 9F 38 02
91A7:E9 C8 90 03 4C 4D 90 AD 9A
91AF:FE 9F CD FC 9F F0 EA 20 7A
91B7:36 96 AD FA 9F CD F8 9F C6
91BF:F0 F5 20 C5 97 AD FF 9F 4C
91C7:4A 4A 8D DB 9F AD FB 9F 5D
91CF:4A 4A 38 ED DB 9F B0 03 52
91D7:4C BE 96 8D E8 9F AD FB 99
91DF:9F 29 03 8D E9 9F A9 00 77
91E7:8D F0 9F 8D EE 9F 20 8A 9B
91EF:92 AD F1 9F 18 6D F7 9F 06
```

91F7:8D F1 9F 90 03 EE F0 9F B0
91FF:38 ED F6 9F AA AD F0 9F 21
9207:E9 00 90 0F 8D F0 9F 8E 22
920F:F1 9F 20 68 95 AD F1 9F 86
9217:4C FF 91 AD EF 9F 18 6D 0B
921F:F4 9F 8D EF 9F 90 03 EE 8B
9227:EE 9F 38 ED F3 9F AA AD B2
922F:EE 9F E9 00 90 0F 8D EE BB
9237:9F 8E EF 9F 20 A7 95 AD 40
923F:EF 9F 4C 29 92 AD FE 9F 49
9247:C9 C7 D0 03 4C B6 96 20 18
924F:CA 95 AD FE 9F CD FC 9F B2
9257:D0 18 20 F3 95 AD FC 9F 2B
925F:38 ED FE 9F F0 09 90 07 C9
9267:AD FD 9F C9 FF D0 03 4C 09
926F:B6 96 AD FE 9F CD F8 9F 01
9277:D0 0E 20 36 96 AD F8 9F EC
927F:38 ED FA 9F F0 E9 90 E7 CD
9287:4C ED 91 AD EC 9F 85 FD 4A
928F:AD EB 9F 85 FE AD E8 9F F2
9297:F0 77 10 05 68 68 4C BE A1
929F:96 20 64 93 AE EA 9F E0 1F
92A7:00 F0 0E CA F0 11 CA F0 C9
92AF:16 29 FC 0D D7 9F 4C CB 3C
92B7:92 AD D4 9F 4C CB 92 29 06
92BF:C0 0D D5 9F 4C CB 92 29 1D
92C7:F0 0D D6 9F A0 00 91 FD A3
92CF:AE E8 9F A5 FD 18 69 08 FF
92D7:85 FD 90 02 E6 FE CA F0 2B
92DF:08 AD D4 9F 91 FD 4C D2 F8
92E7:92 20 64 93 AE E9 9F F0 71
92EF:1F CA F0 0B CA F0 10 29 89
92F7:03 0D D3 9F 4C 0B 93 29 35
92FF:3F 0D D1 9F 4C 0B 93 29 1B
9307:0F 0D D2 9F A0 00 91 FD 73
930F:60 AE EA 9F EC E9 9F 10 C7
9317:4B 20 64 93 E0 00 F0 0B A5
931F:CA F0 19 29 F3 0D D0 9F B2
9327:4C 5F 93 AE E9 9F CA F0 FD
932F:1A CA F0 1F 29 03 0D D3 69
9337:9F 4C 5F 93 AE E9 9F E0 A3
933F:02 F0 18 29 C3 0D CE 9F C8
9347:4C 5F 93 29 3F 0D D1 9F E2
934F:4C 5F 93 29 0F 0D D2 9F 6B
9357:4C 5F 93 29 CF 0D CF 9F 73
935F:A0 00 91 FD 60 78 A9 34 55
9367:85 01 A0 00 B1 FD A0 37 A3
936F:84 01 58 60 AD CD 9F 85 93
9377:FB 18 69 06 8D CD 9F AD C0
937F:CC 9F 85 FC 69 00 8D CC A8

```

9387:9F AD DF 9F 10 08 AD DE BA
938F:9F 10 03 4C 4D 90 A9 02 B1
9397:8D E1 9F 20 F3 95 A9 01 3E
939F:8D CB 9F AD FE 9F 38 ED 24
93A7:FC 9F 90 08 8D F6 9F A9 F8
93AF:FF 8D CB 9F AD F5 9F 10 42
93B7:27 AC FC 9F AD FE 9F 8D 6D
93BF:FC 9F 8C FE 9F AC FD 9F 1A
93C7:AD FF 9F 8D FD 9F 8C FF 1A
93CF:9F AC CB 9F 10 05 C8 C8 54
93D7:4C DC 93 88 88 8C CB 9F 05
93DF:20 C5 97 AD F6 9F CD F7 20
93E7:9F B0 47 AD F7 9F 4A 8D 2F
93EF:F1 9F A9 00 8D F0 9F AE 4B
93F7:F7 9F E0 00 D0 03 4C 4D 97
93FF:90 CA 20 7F 94 AD F1 9F FC
9407:18 6D F6 9F 8D F1 9F 90 74
940F:03 EE F0 9F AD F1 9F 38 3A
9417:ED F7 9F A8 AD F0 9F E9 0E
941F:00 90 09 8C F1 9F 8D F0 70
9427:9F 20 DB 94 20 B9 94 4C 4A
942F:F9 93 AD F6 9F 4A 8D F1 92
9437:9F A9 00 8D F0 9F AE F6 CD
943F:9F E0 00 D0 03 4C 4D 90 F1
9447:CA AD FF 9F C9 A0 F0 03 F0
944F:20 7F 94 AD F1 9F 18 6D 81
9457:F7 9F 8D F1 9F 90 03 EE 69
945F:F0 9F AD F1 9F 38 ED F6 6E
9467:9F A8 AD F0 9F E9 00 90 84
946F:09 8C F1 9F 8D F0 9F 20 08
9477:B9 94 20 DB 94 4C 40 94 4F
947F:AD EC 9F 85 FD AD EB 9F 25
9487:85 FE 20 64 93 AC EA 9F 42
948F:F0 10 88 F0 15 88 F0 1A 1C
9497:29 FC 0D D7 9F A0 00 91 C4
949F:FD 60 29 3F 0D D1 9F 4C 34
94A7:9C 94 29 CF 0D CF 9F 4C 99
94AF:9C 94 29 F3 0D D0 9F 4C E7
94B7:9C 94 EE FF 9F EE EA 9F 60
94BF:AD EA 9F C9 04 F0 01 60 51
94C7:AD EC 9F 18 69 08 8D EC EB
94CF:9F 90 03 EE EB 9F A9 00 6D
94D7:8D EA 9F 60 AD CB 9F 30 88
94DF:29 AD FE 9F 29 07 C9 07 E2
94E7:F0 0B EE EC 9F D0 17 EE 56
94EF:EB 9F 4C 05 95 AD EC 9F AD
94F7:18 69 39 8D EC 9F AD EB B4
94FF:9F 69 01 8D EB 9F EE FE 07
9507:9F 60 AD FE 9F 29 07 C9 39
950F:00 F0 0E AD EC 9F D0 03 9D

```

9517:CE EB 9F CE EC 9F 4C 31 35
951F:95 AD EC 9F 38 E9 39 8D 81
9527:EC 9F AD EB 9F E9 01 8D 59
952F:EB 9F CE FE 9F 60 AD DF BB
9537:9F 10 05 AD DE 9F 30 18 9F
953F:A0 02 B1 FD 8D FF 9F C8 C5
9547:B1 FD 8D FE 9F 38 E9 C8 E6
954F:B0 06 20 C5 97 20 7F 94 85
9557:AD CD 9F 18 69 04 8D CD 86
955F:9F 90 03 EE CC 9F 4C 4D 98
9567:90 AD EA 9F 18 6D F5 9F 9F
956F:30 08 C9 04 F0 1B 8D EA 28
9577:9F 60 AD EC 9F 38 E9 08 C8
957F:8D EC 9F B0 03 CE EB 9F 76
9587:A9 03 8D EA 9F EE E8 9F D2
958F:60 AD EC 9F 18 69 08 8D F1
9597:EC 9F 90 03 EE EB 9F A9 73
959F:00 8D EA 9F CE E8 9F 60 3F
95A7:AD E9 9F 18 6D F2 9F 30 40
95AF:08 C9 04 F0 0D 8D E9 9F F2
95B7:60 A9 03 8D E9 9F CE E8 0B
95BF:9F 60 A9 00 8D E9 9F EE 4A
95C7:E8 9F 60 AD FE 9F 29 07 06
95CF:C9 07 F0 0B EE EC 9F 90 6B
95D7:17 EE EB 9F 4C EF 95 AD BC
95DF:EC 9F 18 69 39 8D EC 9F 7C
95E7:AD EB 9F 69 01 8D EB 9F 25
95EF:EE FE 9F 60 AC E1 9F B1 2A
95F7:FB 8D FF 9F C8 B1 FB 8D 11
95FF:FE 9F C8 B1 FB 8D FD 9F 78
9607:A2 01 8E F5 9F 38 ED FF B0
960F:9F B0 0B A2 FF 8E F5 9F 89
9617:AD FF 9F 38 F1 FB 8D F7 25
961F:9F C8 B1 FB 8D FC 9F 38 1C
9627:ED FE 9F 8D F6 9F 4A 8D 30
962F:F1 9F 88 8C E1 9F 60 AC 12
9637:E0 9F B1 FB 8D FB 9F 88 D6
963F:B1 FB 8D FA 9F 88 B1 FB 24
9647:A2 01 8E F2 9F 38 ED FB BC
964F:9F B0 0B A2 FF 8E F2 9F C3
9657:AD FB 9F 38 F1 FB 8D F4 61
965F:9F 88 B1 FB 8D F8 9F 38 3C
9667:ED FA 9F 8D F3 9F 4A 8D 57
966F:EF 9F C8 8C E0 9F 60 AD 52
9677:DA 9F 29 03 8D D7 9F 0A 64
967F:0A 0D D7 9F 8D D6 9F 0A FA
9687:0A 0D D7 9F 8D D5 9F 0A FE
968F:0A 0D D7 9F 8D D4 9F 0A 03
9697:0A 8D D3 9F 0A 0A 8D D2 08
969F:9F 0A 0A 8D D1 9F 4A 4A 25


```
96A7:8D CF 9F 4A 4A 8D D0 9F F1
96AF:0D CF 9F 8D CE 9F 60 A9 83
96B7:00 8D DF 9F 4C 4D 90 A9 A0
96BF:80 8D DF 9F 4C 4D 90 A0 DF
96C7:00 91 FB C8 D0 FB E6 FC A6
96CF:E4 FC D0 F5 CC DC 9F F0 32
96D7:06 91 FB C8 4C D3 96 60 B7
96DF:AD ED 9F C9 A0 F0 0F C9 A0
96E7:E0 F0 73 20 5E 8F A9 A0 57
96EF:8D ED 9F 4C F5 96 A9 00 75
96F7:A0 00 99 00 A0 99 00 A1 B5
96FF:99 00 A2 99 00 A3 99 00 A9
9707:A4 99 00 A5 99 00 A6 99 FC
970F:00 A7 99 00 A8 99 00 A9 B0
9717:99 00 AA 99 00 AB 99 00 E3
971F:AC 99 00 AD 99 00 AE 99 A9
9727:00 AF 99 00 B0 99 00 B1 13
972F:99 00 B2 99 00 B3 99 00 1D
9737:B4 99 00 B5 99 00 B6 99 56
973F:00 B7 99 00 B8 99 00 B9 75
9747:99 00 BA 99 00 BB 99 00 56
974F:BC 99 00 BD 99 00 BE 99 03
9757:00 BF 88 D0 9D 60 A9 00 56
975F:A0 00 99 00 E0 99 00 E1 61
9767:99 00 E2 99 00 E3 99 00 1C
976F:E4 99 00 E5 99 00 E6 99 0A
9777:00 E7 99 00 E8 99 00 E9 6B
977F:99 00 EA 99 00 EB 99 00 55
9787:EC 99 00 ED 99 00 EE 99 B6
978F:00 EF 99 00 F0 99 00 F1 CD
9797:99 00 F2 99 00 F3 99 00 8E
979F:F4 99 00 F5 99 00 F6 99 63
97A7:00 F7 99 00 F8 99 00 F9 30
97AF:99 00 FA 99 00 FB 99 00 C7
97B7:FC 99 00 FD 99 00 FE 99 10
97BF:00 FF 88 D0 9D 60 A9 00 CE
97C7:85 FE 8D DB 9F AD FE 9F 3A
97CF:29 F8 85 FD 0A 26 FE 0A 53
97D7:26 FE 18 65 FD 90 02 E6 50
97DF:FE 0A 26 FE 0A 26 FE 0A B6
97E7:26 FE 85 FD AD FE 9F 29 4C
97EF:07 18 65 FD 85 FD 90 02 7C
97F7:E6 FE AD FF 9F 29 FC 0A B5
97FF:2E DB 9F 18 65 FD 85 FD DE
9807:8D EC 9F A5 FE 6D DB 9F 8D
980F:6D ED 9F 85 FE 8D EB 9F 64
9817:AD FF 9F 29 03 8D EA 9F 69
981F:60 05 40 00 00 FF FF 00 C9
```

Program 2. Animation Demo

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

```

JC 90 POKE51,255:POKE52,63:POKE55,255:POKE56,63
SE 100 INPUT"{CLR}{4 DOWN}DO YOU WANT TO SAVE (Y/N
    )";NS$:IFNS$="N"THEN140
CH 110 IFNS$<>"Y"THEN100
XX 120 INPUT"DISK OR TAPE (D/T)";DV$
EX 130 IF(DV$<>"D")AND(DV$<>"T")THEN120
XQ 140 CLEAR:SETCOLOR 1,6,0,14,1:SC=9:RECORD:FOR S=
    1 TO 70:AN=AN+10:C=C+1:SC=SC*.94
JE 150 TH=AN*↑/180
KA 160 A(1)=-10:B(1)=10:A(2)=10:B(2)=10:A(3)=10:B(
    3)=-10:A(4)=-10:B(4)=-10
DA 170 NU=4:GOSUB1000:NEXT
KF 180 IFDV$="D"THEN:DPUT"SQUARES"
MB 190 IFDV$="T"THEN:LOWRES:CPUT"SQUARES"
XM 200 PLAY:PLAY:CLEAR
CJ 210 SETCOLOR 1,6,0,7,1:SC=2:RECORD:C=0:AN=0:FOR
    S=0 TO 70:C=C+1:SC=SC*.94
HJ 220 TH=AN*↑/180
JS 230 A(1)=-60:B(1)=40:A(2)=60:B(2)=40:A(3)=0:B(3
    )=-75:NU=3:GOSUB1000
HA 240 A(1)=0:B(1)=75:A(2)=60:B(2)=-40:A(3)=-60:B(
    3)=-40:GOSUB1000:AN=AN+5:NEXT
PD 250 IFDV$="D"THEN:DPUT"STARS"
HE 260 IFDV$="T"THEN:LOWRES:CPUT"STARS"
DQ 270 PLAY:PLAY:CLEAR
DK 280 SC=2:C=0:TH=0:SETCOLOR 1,3,5,7,9:RECORD:FOR
    S=0 TO 70:C=C+1:SC=SC*.94
AM 290 A(1)=-60:B(1)=40:A(2)=60:B(2)=40:A(3)=0:B(3
    )=-75:NU=3:GOSUB1000
JA 300 A(1)=0:B(1)=75:A(2)=60:B(2)=-40:A(3)=-60:B(
    3)=-40:NU=3:GOSUB1000:NEXT
PD 310 IFDV$="D"THEN:DPUT"MORE STARS"
GG 320 IFDV$="T"THEN:LOWRES:CPUT"MORE STARS"
KB 330 PLAY:PLAY:CLEAR
KQ 340 SC=5:C=0:AN=0:SETCOLOR 1,0,7,14,1:RECORD:FO
    RS=0 TO 1040:SC=SC+.1:C=C+1
GX 350 IFC=4 THEN C=1
HA 360 TH=AN*↑/180:A(1)=0:B(1)=0:A(2)=0:B(2)=1:NU=
    2:GOSUB1000:AN=AN+5:NEXT
AS 370 IFDV$="D"THEN:DPUT"RAYS"
GF 380 IFDV$="T"THEN:LOWRES:CPUT"RAYS"
DH 390 PLAY:CLEAR:PLAY:CLEAR
RC 400 OF=135:SC=700:GAP=500:DG=↑/180:SETCOLOR 1,6
    ,0,7,1:RECORD
AP 410 FORAN=0 TO 180 STEP 5:TH=AN*DG:FORBE=0 TO 360 STEP
    5:GM=BE*DG:DL=(BE+OF)*DG

```

```

HB 420 A(1)=-50:B(1)=40*SIN(GM):C(1)=40*COS(GM)
DB 430 A(2)=50:B(2)=40*SIN(DL):C(2)=40*COS(DL)
BB 440 FORN=1TO2:GOSUB4000:GOSUB5000:NEXT:C=3:GOSU
      B1020:NEXTBE:SWITCH:NEXTAN
AM 450 IFDV$="D"THEN:DPUT"WIRES"
ES 460 IFDV$="T"THEN:LOWRES:CPUT"WIRES"
MC 470 PLAY:CLEAR:LOWRES:PRINT"{CLR}PLEASE WAIT..."
      "
GM 480 DIM SN(360),CS(360):DG=1/180:FORX=0TO360STE
      P2:TH=X*DG:SN(X)=SIN(TH)
JH 490 CS(X)=COS(TH):NEXT:OF=135:SC=350:GA=25:GM=3
      0*DG:SETCOLOR 1,0,2,7,5:RECORD
BP 500 FORAN=0TO342STEP2:TH=AN*DG:RESTORE:FORFA=1T
      O6:READC
CG 510 FORVE=1TO4:READA(VE),B(VE),C(VE):NEXTVE
JP 520 FORN=1TO4:GOSUB6000:GOSUB7000
GE 530 A(N)=A(N)+4-AN/50:B(N)=B(N)+4-AN/50:C(N)=C(
      N)-AN/20:GOSUB5000:NEXTN
MA 540 GOSUB1040:NEXTFA:SWITCH:NEXTAN
AH 550 IFDV$="D"THEN:DPUT"CUBE"
DA 560 IFDV$="T"THEN:LOWRES:CPUT"CUBE"
XB 570 PLAY:PLAY:LOWRES:END
ES 1000 FORN=1TONU:GOSUB2000:GOSUB3000:NEXT
CP 1010 ON NU-1 GOSUB 1020,1030,1040:RETURN
MF 1020 FILL 2,C,A(1),B(1),A(2),B(2):RETURN
XQ 1030 FILL 3,C,A(1),B(1),A(2),B(2),A(3),B(3):RET
      URN
CM 1040 FILL 4,C,A(1),B(1),A(2),B(2),A(3),B(3),A(4
      ),B(4):RETURN
SM 2000 Q=A(N)*COS(TH)+B(N)*SIN(TH)
AF 2010 B(N)=-A(N)*SIN(TH)+B(N)*COS(TH)
AE 2020 A(N)=Q:RETURN
BP 3000 A(N)=80+SC*A(N):B(N)=100-1.4*SC*B(N):RETUR
      N
QD 4000 Q=A(N)*COS(TH)+C(N)*SIN(TH)
KB 4010 C(N)=-A(N)*SIN(TH)+C(N)*COS(TH)
SH 4020 A(N)=Q:RETURN
KS 5000 K=GA+C(N):A(N)=80+SC*A(N)/K:B(N)=100-1.4*S
      C*B(N)/K:RETURN
JC 6000 Q=B(N)*CS(30)+C(N)*SN(30)
KH 6010 C(N)=-B(N)*SN(30)+C(N)*CS(30)
KF 6020 B(N)=Q:RETURN
XQ 7000 Q=A(N)*CS(AN)+C(N)*SN(AN)
SD 7010 C(N)=-A(N)*SN(AN)+C(N)*CS(AN)
MP 7020 A(N)=Q:RETURN
JF 8000 DATA 1,-1,+1,-1,+1,+1,-1,+1,-1,-1,-1,-1,-1
HG 8010 DATA 2,+1,+1,-1,+1,+1,+1,+1,-1,+1,+1,-1,-1
DH 8020 DATA 1,+1,+1,+1,-1,+1,+1,-1,-1,+1,+1,-1,+1
AE 8030 DATA 2,-1,+1,+1,-1,+1,-1,-1,-1,-1,-1,-1,+1
PD 8040 DATA 3,-1,+1,+1,+1,+1,+1,+1,+1,-1,-1,+1,-1
BB 8050 DATA 3,+1,-1,-1,+1,-1,+1,-1,-1,+1,-1,-1,-1

```

Program 3. Animation Player

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

```
KK 130 POKE51,255:POKE52,63:POKE55,255:POKE56,63
XC 140 IFA=0THENA=1:LOAD "FILL 64",8,1
JX 150 SYS 33846:CLEAR
SC 160 DGET"SQUARES":PLAY
MX 170 DGET"STARS":PLAY
AX 180 DGET"MORE STARS":PLAY
FE 190 DGET"RAYS":PLAY
HX 220 SETCOLOR 14,6,0,7,14
BG 230 LOWRES:END
```

Sequential File Reader

Gene Nash

If you've ever wanted to read a sequential file while you were working on a BASIC program, you'll appreciate this program. It reads a file off your disk and prints it on your screen or printer without disturbing the BASIC program you're working on. Disk drive required.

"Sequential File Reader" can be an extremely useful tool. It lets you read any sequential file without erasing the BASIC program you're working on. You can print your file to the screen, to a printer, or both. It's especially useful when you are writing a program that creates or modifies sequential files.

BASIC Loader

Sequential File Reader is written entirely in machine language, so it must be entered using the "MLX" machine language entry program found elsewhere in this issue. Be sure you read and understand the instructions for using MLX before you begin entering the data for Sequential File Reader. When you're ready to begin entering data, load and run MLX. You'll be asked for a starting address and an ending address. Respond to these prompts as follows:

Starting address: CD78

Ending address: CFBF

After you finish entering all the data for Sequential File Reader, be sure to save a copy before you exit MLX.

You should load Sequential File Reader at the beginning of each programming session—that way, it will always be ready. Use a command of the form **LOAD "filename",8,1**, where *filename* is the name you used when you saved the data with MLX. After the program has loaded, type **NEW** and press RETURN. If you forget, save the program you're working on, load and RUN Sequential File Reader, type NEW and press RETURN to reset important pointers, and then reload your program.

Whenever you want to read a sequential file, type **SYS 52600**. You're prompted for a filename. Press RETURN alone to go back to BASIC. The computer then asks if you want the output to go to the screen or a printer. If you choose the printer, you'll be asked if you want the output to go to the screen as well.

If an error occurs when the program tries to open the file, control returns to BASIC. Printing can be paused and restarted by pressing the *P* key. Pressing *S* stops the program and returns to BASIC.

Sequential File Reader

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

```
CD78:A9 80 85 FB A9 CF 85 FC 0E
CD80:20 28 CF 20 AD CE AD 7B B1
CD88:CF D0 01 60 A9 0D 20 D2 FA
CD90:FF A9 8C 85 FB A9 CF 85 2C
CD98:FC 20 28 CF 20 E4 FF C9 1B
CDA0:53 D0 15 20 D2 FF A9 0E B6
CDA8:20 D2 FF A9 03 8D 7D CF BC
CDB0:A9 00 8D 7E CF 4C F7 CD 28
CDB8:C9 50 D0 E0 20 D2 FF A9 6B
CDC0:0D 20 D2 FF A9 04 8D 7D 3B
CDC8:CF A9 A6 85 FB A9 CF 85 8F
CDD0:FC 20 28 CF 20 E4 FF C9 53
CDD8:59 D0 10 20 D2 FF A9 0E 51
CDE0:20 D2 FF A9 01 8D 7E CF E6
CDE8:4C F7 CD C9 4E D0 E5 20 A0
CDF0:D2 FF A9 00 8D 7E CF EE 20
CDF8:7B CF EE 7B CF A9 0D 20 3B
CE00:D2 FF 20 D2 FF A9 00 A0 7F
CE08:CF A2 BD 20 BD FF A9 0F 40
CE10:A2 08 A0 0F 20 BA FF 20 12
CE18:C0 FF A9 00 A0 CF A2 BD 92
CE20:20 BD FF AD 7D CF AA C9 62
CE28:04 D0 05 A0 07 4C 32 CE 43
CE30:A0 00 20 BA FF 20 C0 FF CF
CE38:AD 7B CF A0 CF A2 BD 20 34
CE40:BD FF A9 05 A2 08 A0 05 BD
CE48:20 BA FF 20 C0 FF 20 49 36
CE50:CF 20 CC FF 20 E4 FF F0 FC
CE58:0E C9 53 D0 03 4C 99 CE 32
CE60:C9 50 D0 03 20 38 CF A2 65
CE68:05 20 C6 FF 20 E4 FF 8D 8B
CE70:7C CF A5 90 8D 7F CF 20 28
CE78:CC FF AE 7D CF 20 C9 FF BC
CE80:AD 7C CF 20 D2 FF AE 7E 82
```

```

CE88:CF F0 09 20 CC FF AD 7C AB
CE90:CF 20 D2 FF AE 7F CF F0 7C
CE98:B8 A9 05 20 C3 FF A9 0F 20
CEA0:20 C3 FF AD 7D CF 20 C3 49
CEA8:FF 20 CC FF 60 A2 00 8E 04
CEB0:7B CF 20 07 CF 20 E1 FF 37
CEB8:D0 08 A9 00 8D 7B CF 4C 3C
CEC0:01 CF 20 E4 FF F0 EE C9 90
CEC8:A0 D0 02 A9 20 C9 14 F0 07
CED0:41 C9 0D F0 2C C9 20 90 8B
CED8:DC C9 C1 B0 07 C9 80 90 8B
CEE0:0A 4C B5 CE C9 DB 90 03 1C
CEE8:4C B5 CE AE 7B CF E0 10 CB
CEF0:F0 C3 20 D2 FF 9D BF CF EE
CEF8:20 07 CF EE 7B CF 4C B5 BA
CF00:CE A9 20 20 D2 FF 60 A9 78
CF08:3C 20 D2 FF A9 9D 20 D2 FE
CF10:FF 60 AE 7B CF E0 00 F0 48
CF18:9C 20 D2 FF CE 7B CF CA 37
CF20:A9 00 9D BF CF 4C B5 CE 2E
CF28:A0 00 B1 FB AA C8 B1 FB E5
CF30:20 D2 FF C8 CA D0 F7 60 0B
CF38:20 E4 FF C9 50 D0 01 60 E5
CF40:C9 53 D0 F4 68 68 4C 99 1A
CF48:CE A2 0F 20 C6 FF 20 E4 37
CF50:FF C9 30 D0 04 20 CC FF AF
CF58:60 20 D2 FF 20 E4 FF 20 3F
CF60:D2 FF A5 90 F0 F6 A9 0D EA
CF68:20 D2 FF A9 00 20 C3 FF 6F
CF70:69 01 C9 10 D0 F7 68 68 DE
CF78:4C CC FF 00 00 00 00 00 71
CF80:0B 46 49 4C 45 4E 41 4D 58
CF88:45 20 30 3A 19 50 52 49 74
CF90:4E 54 45 52 20 4F 52 20 3D
CF98:53 43 52 45 45 4E 3F 20 53
CFA0:28 50 2F 53 29 20 16 45 BE
CFA8:43 48 4F 20 54 4F 20 53 5B
CFB0:43 52 45 45 4E 3F 20 28 5B
CFB8:59 2F 4E 29 20 30 3A 00 63

```

64 RAM Expander

Alton C. Williams

Would you like to expand your Commodore 64's memory without hardware add-ons? This short utility program opens an extra 16K of memory for storing machine language programs which can be executed with a normal SYS from BASIC. It doesn't really change your 64—the memory has always been there, hidden under the computer's ROM chips. But the program does make this memory easily accessible from BASIC, almost as if you'd gotten a free 16K memory upgrade.

The Commodore 64 has more memory than meets the eye. However, most machine language programs use the 4K section of RAM located at 49152–53247 (\$C000–\$CFFF) or the 191 bytes of cassette buffer space at 828–1019 (\$033C–\$03FB). That's not enough room for long programs. And since many 64 programs use these same areas, it's hard to find a place for a new machine language program without consuming some of BASIC RAM, which in turn leaves less room for a BASIC program and variables.

Fortunately, there's another solution. The 64 has a full 16K (16,384 bytes) of RAM underlying the BASIC and Kernal ROM chip addresses. The BASIC language interpreter is located in the 8K section from 40960–49151 (\$A000–\$BFFF), and the Kernal operating system is located in the 8K section from 57344–65535 (\$E000–\$FFFF).

To use this extra memory from BASIC, however, is not easy—one reason why it's often called *hidden* RAM. If you SYS to an address in this area from BASIC, the computer executes the instructions recorded in the ROM addresses, ignoring the contents of the hidden RAM. For instance, SYS 59626 calls the Kernal ROM routine that scrolls the screen up one line. SYS 58726 calls the ROM routine to home the cursor, and so on.

"64 RAM Expander" opens up new programming space by allowing you to SYS to a machine language program stored in underlying RAM. It doesn't really expand your 64's memory (the RAM has been there all the time), but the effect is the same as if you suddenly gained 16K of extra memory space.

Expand Your RAM

Type in and save a copy of 64 RAM Expander (Program 1). To use the program, simply load and run it. Now you can install any machine language program designed to reside in the hidden RAM. Use the statement **SYS 920** to tell the computer that subsequent SYS statements will use the hidden RAM area. To return to normal, use the statement **SYS 931**. These two commands allow you to switch the hidden RAM in and out at will.

When you SYS to hidden RAM, 64 RAM Expander switches out the ROM overlying the area where the program is located. If your program doesn't call BASIC or Kernal routines, that poses no problems. But BASIC ROM routines are not available when you SYS to a program located under BASIC (however, you may still call Kernal ROM routines). If your program is located under the Kernal ROM, neither BASIC nor Kernal routines are available, since switching out the Kernal ROM switches out BASIC as well.

This program occupies the cassette buffer area from memory locations 920–1018 (\$0398–\$03FA). Do not use those locations when 64 RAM Expander is installed.

A Short Demonstration

Program 2 is a short BASIC loader which demonstrates 64 RAM Expander. Load and run 64 RAM Expander; then load and run Program 2. It POKes a short machine language routine into the RAM underlying BASIC ROM, activates 64 RAM Expander with SYS 920, and then activates the hidden ML program with a SYS to location 41000 (SA in line 90). The program cycles the screen border colors until you press a key.

If you're not familiar with bank switching on the 64, you may well wonder how one writes a program designed to live in the RAM under ROM. Most machine language monitors see only ROM at those addresses, since the monitor itself typically needs ROM routines to function. One way is to write fully relocatable code, which executes the same way no matter where it loads into memory. The longer the program, however, the more difficult it becomes to preserve full relocatability. Another way is to write the program in a more convenient area, then adjust all the absolute addresses by hand. Again, that's a tedious business for all but the simplest programs. The best solution is to use a good machine language assembler which allows you to assemble object code to a disk file rather than to

memory. Once the object file has been created, you can load it with ,8,1. Like POKES from BASIC, a relocating load stores data in underlying RAM rather than ROM.

Program 1. 64 RAM Expander

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

```
KE 10 PRINT "{CLR}":T=0:FOR I=920 TO 1018:READX:T=T
    +X:POKEI,X:NEXT
FC 20 IF T<>9573 THEN PRINT"ERROR IN DATA STATEMEN
    TS!!":END
FM 30 PRINT"EASY 16K IS NOW READY FOR USE!":END
ES 31 DATA 169,174,141,8,3,169,3,141,9,3,96,169,22
    8,141,8,3,169
PG 32 DATA 167,141,9,3,96,32,115,0,8,201,158,240,4
    ,40,76,231,167
EM 33 DATA 40,32,115,0,32,138,173,32,247,183,24,16
    5,21,105,32,144,8
QX 34 DATA 120,169,48,133,1,76,227,3,165,21,105,64
    ,176,10,165,21,105
GD 35 DATA 96,144,4,169,54,133,1,169,243,56,233,1,
    170,169,3,233,0
PD 36 DATA 72,138,72,108,20,0,169,55,133,1,88,76,1
    74,167
```

Program 2. RAM Expansion Demo

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

```
CS 10 REM POKE PROGRAM INTO RAM UNDER BASIC ROM
GB 20 ADR=41000:SA=ADR:CHK=0
EF 30 READ BYTE:IF BYTE<>256 THEN POKE ADR,BYTE:AD
    R=ADR+1:CHK=CHK+BYTE:GOTO 30
RD 40 IF CHK<>1577 THEN PRINT "ERROR IN DATA STATE
    MENTS":END
QE 50 SYS 920:REM TURN ON RAM EXPANDER
BG 60 PRINT:PRINT
DX 70 PRINT "NOW RUNNING ML PROGRAM UNDER BASIC RO
    M. "
ED 80 PRINT "PRESS ANY KEY TO QUIT. "
EA 90 SYS SA:REM START OUR PROGRAM
CD 100 DATA 238,32,208,32,228,255,240
DS 110 DATA 248,96,256
```

Chapter 6

For Everything Else

Turbo Format

Ross Ouwinga

If you're interested in saving time—and wear and tear—on your 1541 disk drive, put this program high on your list of utilities. It formats a disk in only 9½ seconds—a super speed-up from the usual minute and 20 seconds—and it eliminates head knocking. And it's easy to use—just answer a few prompts, and you're seconds away from a formatted disk.

“Turbo Format” is a fast, easy-to-use disk-formatting program. It formats a disk in only 9½ seconds—the standard 1541 format program requires one minute and 20 seconds. That's a speed increase of over eight times. It's easy to use because you don't have to remember the cryptic command line required by the 1541 to format a disk. And you're prompted through each step of the procedure so there's no need to memorize anything.

Machine Language Speed

Turbo Format is written entirely in machine language. “MLX,” the machine language entry program found in Appendix D, is required to type it in. After loading and running MLX, answer the prompts for starting and ending addresses with the following:

Starting address: 0801

Ending address: 0EC8

Type in Turbo Format and save it to disk. To run it, type **LOAD “filename”,8** and then **RUN**. There will be a short delay (while most of the program is transferred to the memory in the disk drive), and a title screen with the first prompt will be displayed.

Three Easy Steps

You're first prompted to enter the disk name (up to 16 characters). Virtually any character may be used in the name, but some may cause undesirable side effects, so caution is recommended whenever characters other than letters or numbers are used.

The second prompt is for the disk ID. This is the number used by the disk drive primarily to determine when a disk has been changed. Normally this number is two characters long, but Turbo Format allows up to five characters. More than five characters may be typed in, but only the first five will be used. Only the first two characters will actually be used by the disk drive to identify the disk after it has been formatted, but all five will appear in the BAM (Block Allocation Map).

Next, you'll be instructed to insert the disk to be formatted. *Don't leave your disk with Turbo Format in the disk drive.* Formatting the disk will erase everything. A recommended safeguard is to put a write-protect tab on your disk immediately after you have a working copy. Turbo Format does check the write-protect tab, and if one is present, the disk won't be formatted. Also, an error message will appear to notify you that the disk is write-protected.

Along with the prompt to insert a disk is an abort option. Press the f1 key to abort the formatting process. This causes the program to start again at the beginning, prompting you to reenter the disk name. With this option, you can change your mind and enter a new disk name or ID or correct any spelling errors before formatting the disk.

Two Kinds of Formatting

The formatting process doesn't actually begin until you press the space bar or RETURN. Note, however, that the space bar and RETURN do not perform the same function. If you press the space bar, the disk will be formatted and all data written to the disk will be checked for errors. If an error occurs, the formatting will stop and the type of error will be displayed on the screen. If you press RETURN, the disk will be formatted as quickly as possible without checking for errors.

The reason for offering the two options is to allow you to use the program in a manner which most closely suits your needs. If the disks are verified, the procedure takes 17 seconds rather than 9½ seconds. Verifying provides more assurance that the disk is formatted correctly. The additional 7 or 8 seconds may not make much difference to you. On the other hand, if speed is important, you may choose not to verify your disks. Errors occur very rarely and most are self-correcting, provided the disk itself is not damaged. Also, if you have a disk which you know is faulty, but wish to format anyway,

you must use the no-verify option. I strongly recommend using the verify feature if you're formatting the back side of a disk certified only for single-side use, or if the disk is well used, or if you use an inexpensive, off-brand disk.

No Knock

After you've pressed the space bar or RETURN to format, the process begins. A feature that you'll notice immediately is that there's no loud knocking noise at the beginning of the format routine. A single short click is heard instead to indicate that the routine is working correctly. This feature is possible since Turbo Format is loaded from a disk at the start and the exact location of the read/write head is saved in memory in the disk drive. The program is then able to calculate the exact position of the first track without the excessive knock. The standard format routine doesn't necessarily know the position of the read/write head; it therefore assumes the worst-case condition and tries to step the head down 45 tracks. If the head reaches track 1 before it has stepped 45 times (which is usually the case), it bangs the head against the stop until the count of 45 is complete. It works, but it makes an awful racket and, in some cases, eventually works the disk drive out of alignment.

When the format process is complete, a message is displayed and will indicate OK if there were no errors. At this point you have the option of formatting another disk or quitting. It's important that you press f1 to quit. Don't reset the computer by turning it off and back on. The reason for this is that the format routine leaves the read/write head on track 35 when it finishes. By using f1 to quit, the disk drive is sent an initialize command and also a reset command to set everything back in order. If you accidentally turn the computer off, you should initialize the disk drive using the IO: command. Failure to do so could cause erratic operation. If you accidentally restarted when you intended to quit, it would be necessary to press RETURN for the *Enter name* and *Enter ID* prompts and to press the space bar with no disk in the disk drive. An error would be detected immediately and you would again be given the prompt to quit by pressing f1.

Using Turbo Format doesn't mean your disks will become less reliable. It's much faster because it eliminates several unnecessary and time-consuming routines. Most of the time used to format a disk is spent by a routine whose only function is

to space the sectors evenly around the disk. Turbo Format sets this space the same for every sector on every track, but skips a time-consuming calculation process used in standard formatting. It's amazing how much time is saved by this one modification. To a lesser extent, time is saved by reducing delays when stepping between tracks and by eliminating unnecessary data-conversion routines. A technical discussion of this is beyond the scope of this article, but it should be noted that only those procedures that don't affect reliability are changed. Disks formatted with Turbo Format work just as well as those formatted the standard way.

Intermediate or advanced machine language programmers might be interested to know that the data recorded to the BAM of the disk is at locations \$0B00 (2816) through \$0BFF (3071). If you have a thorough understanding of the BAM and its contents, you can modify this area of the program to customize or personalize the BAM on your disks. Be aware, however, that modifying the BAM may have undesirable side effects and isn't recommended for the beginner or someone who knows no more than the basics of disk drive operation.

If you have a non-1541 disk drive, Turbo Format will likely work if the drive claims to be 100-percent compatible. Most of the program is transferred to and operates in the disk drive and calls on several routines which are part of the standard program in the drive. If these routines are not at the same addresses as in the 1541, the program won't operate correctly. No damage will occur to your disk drive if this should happen, but it may be necessary to turn it off and on again to regain control.

Note: Turbo Format will not work with the Commodore 1571 disk drive.

Turbo Format

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```
0801:0D 08 0A 00 9E 28 32 30 05
0809:36 33 29 00 00 00 A9 16 8F
0811:85 FF A9 00 85 FB 85 FD 3E
0819:A9 0C 85 FC A9 05 85 FE EC
0821:20 89 09 20 51 09 A2 0B C5
0829:20 56 09 A0 00 20 5F 09 52
0831:A0 12 20 5F 09 A0 64 20 C3
0839:5F 09 A0 FF 20 CF FF C8 58
0841:C9 0D F0 0C C0 10 D0 03 43
0849:88 D0 F1 99 90 0B D0 EC E8
```


0851:C0 10 F0 08 A9 A0 99 90 F7
0859:0B C8 D0 F4 A0 73 20 5F FC
0861:09 A0 FF 20 CF FF C8 C9 F9
0869:0D F0 0C C0 05 D0 03 88 C3
0871:D0 F1 99 A2 0B D0 EC A9 E2
0879:96 8D A7 0B 20 77 09 A0 6F
0881:7F 20 5F 09 20 E4 FF C9 34
0889:00 F0 F9 C9 85 F0 60 C9 2C
0891:0D F0 1E C9 20 D0 ED 20 05
0899:A6 0E A9 A7 20 A8 FF A9 7D
08A1:04 20 A8 FF A9 01 20 A8 0B
08A9:FF A9 26 20 A8 FF 20 AE 1F
08B1:FF A0 46 20 5F 09 20 B5 C9
08B9:0E A0 02 B9 C4 0E 20 A8 1C
08C1:FF 88 10 F7 A9 00 20 A8 AB
08C9:FF A9 05 20 A8 FF 20 AE 1B
08D1:FF A5 BA 20 B4 FF A9 6F 0D
08D9:85 B9 20 96 FF 20 A5 FF 54
08E1:20 D2 FF C9 0D D0 F6 20 0D
08E9:AB FF A0 53 20 5F 09 A0 4A
08F1:C7 20 6B 09 20 E4 FF C9 4A
08F9:00 F0 F9 C9 85 F0 1E C9 18
0901:0D D0 F1 A9 A0 A0 15 99 F1
0909:90 0B 88 10 FA A0 04 A9 44
0911:41 99 A2 0B 88 A9 32 99 18
0919:A2 0B 4C 24 08 20 B5 0E 45
0921:A9 49 20 A8 FF A9 30 20 10
0929:A8 FF A9 3A 20 A8 FF 20 2C
0931:AE FF 20 B5 0E A9 55 20 DB
0939:A8 FF A9 3A 20 A8 FF A9 C5
0941:00 85 A2 A5 A2 C9 40 30 50
0949:FA 20 AE FF 20 E7 FF 60 B7
0951:A9 93 4C D2 FF A9 20 20 DA
0959:D2 FF CA D0 F8 60 B9 00 F7
0961:0A F0 06 20 D2 FF C8 D0 70
0969:F5 60 B9 00 09 F0 06 20 FD
0971:D2 FF C8 D0 F5 60 A9 08 9F
0979:85 FF A9 00 85 FB 85 FD A8
0981:A9 0B 85 FC A9 04 85 FE 13
0989:20 A6 0E A5 FD 20 A8 FF 33
0991:A5 FE 20 A8 FF A9 20 20 CB
0999:A8 FF A0 00 B1 FB 20 A8 7A
09A1:FF C8 C0 20 90 F6 A5 FB A7
09A9:69 1F 85 FB A5 FC 69 00 9C
09B1:85 FC A5 FD 69 20 85 FD 2F
09B9:A5 FE 69 00 85 FE 20 AE A2
09C1:FF C6 FF D0 C3 60 11 50 A4
09C9:52 45 53 53 20 12 52 45 29
09D1:54 55 52 4E 92 20 54 4F 9F
09D9:20 46 4F 52 4D 41 54 20 D4

09E1:41 4E 4F 54 48 45 52 20 73
09E9:44 49 53 4B 0D 50 52 45 23
09F1:53 53 20 12 46 31 92 20 E3
09F9:54 4F 20 51 55 49 54 11 AC
0A01:05 54 55 52 42 4F 20 46 52
0A09:4F 52 4D 41 54 20 92 0D 6C
0A11:00 11 46 4F 52 4D 41 54 C5
0A19:20 54 49 4D 45 20 49 53 E0
0A21:20 39 2E 35 20 53 45 43 C8
0A29:4F 4E 44 53 0D 31 37 20 F1
0A31:53 45 43 4F 4E 44 53 20 E7
0A39:57 49 54 48 20 56 45 52 91
0A41:49 46 59 0D 00 0D 46 4F 97
0A49:52 4D 41 54 54 49 4E 47 F2
0A51:0D 00 46 4F 52 4D 41 54 48
0A59:20 43 4F 4D 50 4C 45 54 9F
0A61:45 0D 00 0D 11 44 49 53 AB
0A69:4B 20 4E 41 4D 45 20 3F 08
0A71:20 00 0D 44 49 53 4B 20 C9
0A79:49 44 20 3F 20 00 0D 11 67
0A81:49 4E 53 45 52 54 20 44 F4
0A89:49 53 4B 0D 50 52 45 53 FA
0A91:53 20 12 46 31 92 20 54 66
0A99:4F 20 41 42 4F 52 54 0D 23
0AA1:50 52 45 53 53 20 12 52 E1
0AA9:45 54 55 52 4E 92 20 54 96
0AB1:4F 20 46 4F 52 4D 41 54 D1
0AB9:20 57 49 54 48 4F 55 54 A0
0AC1:20 56 45 52 49 46 59 0D 6C
0AC9:50 52 45 53 53 20 12 53 0B
0AD1:50 41 43 45 92 20 54 4F 28
0AD9:20 46 4F 52 4D 41 54 20 D6
0AE1:57 49 54 48 20 56 45 52 3A
0AE9:49 46 59 0D 0D 00 00 00 98
0AF1:00 00 00 00 00 00 00 00 06
0AF9:00 00 00 00 00 00 00 12 20
0B01:01 41 00 15 FF FF 1F 15 8C
0B09:FF FF 1F 15 FF FF 1F 15 A7
0B11:FF FF 1F 15 FF FF 1F 15 AF
0B19:FF FF 1F 15 FF FF 1F 15 B7
0B21:FF FF 1F 15 FF FF 1F 15 BF
0B29:FF FF 1F 15 FF FF 1F 15 C7
0B31:FF FF 1F 15 FF FF 1F 15 CF
0B39:FF FF 1F 15 FF FF 1F 15 D7
0B41:FF FF 1F 15 FF FF 1F 11 DB
0B49:FC FF 07 13 FF FF 07 13 11
0B51:FF FF 07 13 FF FF 07 13 9A
0B59:FF FF 07 13 FF FF 07 13 A2
0B61:FF FF 07 12 FF FF 03 12 91
0B69:FF FF 03 12 FF FF 03 12 19

```

0B71:FF FF 03 12 FF FF 03 12 21
0B79:FF FF 03 11 FF FF 01 11 14
0B81:FF FF 01 11 FF FF 01 11 DB
0B89:FF FF 01 11 FF FF 01 A0 73
0B91:A0 A0 A0 A0 A0 A0 A0 A7
0B99:A0 A0 A0 A0 A0 A0 A0 AF
0BA1:A0 A0 A0 A0 32 41 A0 A0 C6
0BA9:A0 A0 00 00 00 00 00 38
0BB1:00 00 00 00 00 00 00 C7
0BB9:00 00 00 00 00 00 00 CF
0BC1:00 00 00 00 00 00 00 D7
0BC9:00 00 00 00 00 00 00 DF
0BD1:00 00 00 00 00 00 00 E7
0BD9:00 00 00 00 00 00 00 EF
0BE1:00 00 00 00 00 00 00 F7
0BE9:00 00 00 00 00 00 00 FF
0BF1:00 00 00 00 00 00 00 08
0BF9:00 00 00 00 00 00 00 78 88
0C01:AD A2 04 85 12 AD A3 04 04
0C09:85 13 AD A7 04 8D FD 05 30
0C11:A9 A0 8D A7 04 AD 00 1C 45
0C19:09 0C 8D 00 1C 29 10 D0 E0
0C21:05 A9 A6 4C 0F 06 38 A5 66
0C29:22 69 01 85 4A 20 32 06 62
0C31:C6 4A D0 F9 A2 00 86 4B 66
0C39:20 39 06 A9 01 85 22 20 8D
0C41:4B F2 85 43 8A 0A 0A 0A 3B
0C49:0A 0A 85 44 AD 00 1C 29 AC
0C51:9F 05 44 8D 00 1C 20 53 DF
0C59:06 A9 55 8D 01 1C A0 00 1C
0C61:84 4D A5 39 99 00 03 C8 F2
0C69:C8 A5 4D 99 00 03 C8 A5 D5
0C71:22 99 00 03 C8 A5 13 99 CD
0C79:00 03 C8 A5 12 99 00 03 BF
0C81:C8 A9 0F 99 00 03 C8 99 1B
0C89:00 03 C8 B9 FA 02 59 FB A5
0C91:02 59 FC 02 59 FD 02 99 21
0C99:F9 02 E6 4D A5 4D C5 43 12
0CA1:90 C0 98 48 A9 00 85 30 52
0CA9:A9 03 85 31 20 30 FE 68 43
0CB1:A8 88 20 E5 FD 20 F5 FD FC
0CB9:A9 00 85 32 20 61 06 C6 D3
0CC1:4D D0 F9 50 FE B8 20 00 14
0CC9:FE A9 00 85 32 A5 22 C9 5A
0CD1:12 D0 28 A9 04 85 31 20 7F
0CD9:E9 F5 85 3A 20 8F F7 20 08
0CE1:FE 06 D0 2A A2 09 50 FE 90
0CE9:B8 CA D0 FA 20 53 06 20 55
0CF1:BB 06 20 C9 06 50 FE B8 32
0CF9:20 00 FE 20 26 07 D0 0E 01

```

```

0D01:E6 22 20 28 06 A5 22 C9 72
0D09:24 F0 05 4C 40 05 85 4B 43
0D11:AD 00 1C 29 F3 8D 00 1C 0A
0D19:A9 EC 8D 0C 1C A5 4B F0 B4
0D21:05 29 7F 4C C8 C1 60 20 EA
0D29:2B 06 AE 00 1C E8 4C 39 86
0D31:06 20 35 06 AE 00 1C CA D5
0D39:8A 29 03 85 44 AD 00 1C 90
0D41:29 FC 05 44 8D 00 1C A0 59
0D49:06 A2 00 CA D0 FD 88 D0 1C
0D51:FA 60 A9 CE 8D 0C 1C A9 A1
0D59:FF 8D 01 1C 8D 03 1C 60 C9
0D61:20 BB 06 A2 0A A4 32 50 FC
0D69:FE B8 B9 00 03 8D 01 1C D4
0D71:C8 CA D0 F3 A2 09 50 FE D4
0D79:B8 A9 55 8D 01 1C CA D0 BC
0D81:F5 20 BB 06 A0 04 50 FE 2B
0D89:B8 B9 9C 07 8D 01 1C 88 A3
0D91:10 F4 A2 40 A0 04 50 FE FD
0D99:B8 B9 A1 07 8D 01 1C 88 54
0DA1:10 F4 CA D0 EF A9 55 A2 DA
0DA9:08 50 FE B8 8D 01 1C CA BA
0DB1:D0 F7 A5 32 18 69 0A 85 0A
0DB9:32 60 A2 05 A9 FF 50 FE 96
0DC1:B8 8D 01 1C CA D0 F7 60 67
0DC9:A0 BB 50 FE B8 B9 00 01 CA
0DD1:8D 01 1C C8 D0 F4 50 FE FC
0DD9:B8 B9 00 04 8D 01 1C C8 70
0DE1:D0 F4 60 A9 D0 8D 05 18 27
0DE9:2C 05 18 10 0D 2C 00 1C 94
0DF1:30 F6 AD 01 1C B8 A0 00 AC
0DF9:98 60 A9 A1 60 A9 5A 85 AB
0E01:4B 20 E4 06 D0 17 A4 32 26
0E09:A2 0A 50 FE B8 AD 01 1C 8D
0E11:D9 00 03 D0 09 C8 CA D0 59
0E19:F1 A9 00 85 4B 60 C6 4B A5
0E21:D0 DF A9 A0 60 A5 43 85 82
0E29:4D A5 22 C9 12 D0 30 20 8A
0E31:FE 06 D0 63 20 B3 06 20 9A
0E39:E4 06 D0 5B A0 BB 50 FE AC
0E41:B8 AD 01 1C D9 00 01 D0 A8
0E49:4F C8 D0 F2 A2 FC 50 FE 31
0E51:B8 AD 01 1C D9 00 04 D0 BE
0E59:3F C8 CA D0 F1 F0 32 20 85
0E61:FE 06 D0 33 20 B3 06 20 C7
0E69:E4 06 D0 2B A0 04 50 FE FA
0E71:B8 AD 01 1C D9 9C 07 D0 57
0E79:1F 88 10 F2 A2 40 A0 04 D3
0E81:50 FE B8 AD 01 1C D9 A1 45
0E89:07 D0 0D 88 10 F2 CA D0 3A

```

```

0E91:ED C6 4D D0 CA A9 00 60 6A
0E99:A9 A5 60 4A 29 A5 D4 55 83
0EA1:4A 29 A5 94 52 20 B5 0E B7
0EA9:A0 02 B9 C1 0E 20 A8 FF 2C
0EB1:88 10 F7 60 A5 BA 20 B1 25
0EB9:FF A9 6F 85 B9 4C 93 FF AC
0EC1:57 2D 4D 45 2D 4D 00 00 71

```

Turbo Copy

Thomas Tempelmann

This utility is a must for all 1541 disk drive owners with Commodore 64s. It copies an entire disk in only 4½ minutes—and formats the disk as it copies.

The Commodore 1541 disk drive is slow—no one questions that. That characteristic is especially aggravating when you're copying disks. You *can* use a copy program, of course. At least that's easier than having to load, switch disks, and save for each program.

But a normal copy program can take up to 20 minutes to copy a single disk. Much too long.

"Turbo Copy" is the answer. It temporarily turbocharges your 1541 disk drive to make a copy in just 4½ minutes. That's a speed increase of over 400 percent.

Turbo Copying the Right Way

The program is written entirely in machine language. You'll need "MLX," found in Appendix D, to type it in. After loading and running MLX, answer the prompts as follows:

Starting address: 0801

Ending address: 1210

Type in Turbo Copy and save it to disk. Although it's written in machine language, you load and run Turbo Copy just as you would a BASIC program, with

LOAD "filename",8

and then

RUN

Note: Turbo Copy copies all sectors from the source disk to the destination disk, whether or not the sectors are used by a program. Thus, if you wish to back up a disk which contains only one or two short files, you might save time by using a file-copy program instead of Turbo Copy.

Before using Turbo Copy, turn off your printer or extra disk drives (if you have them). Turn off your printer interface—or better yet, disconnect it. Check that there are no other programs or cartridges in the computer when you use Turbo Copy. Since the program uses all of the 64's available RAM, any programs, even hidden ones, might make it crash.

This, then, is what you should do *before* using Turbo Copy:

- Shut off the 64.
- Remove any cartridges.
- Disconnect any printer interfaces (like those that use the cassette port).
- Turn off every peripheral on the serial port except for one disk drive.

Copying at Turbo Speed

Run Turbo Copy. The first thing you'll see is a title screen. After a moment, the disk drive knocks to let you know all's well.

Put the source disk—the disk you want to copy *from*—into the drive and press RETURN.

To be safe, make sure you've covered the write-protect notch on the source disk to avoid accidentally writing to it.

Note that the drive keeps spinning. That's normal.

As soon as you press RETURN, Turbo Copy begins reading the disk contents into memory. Each time a disk sector is read into memory, the screen color changes and the red light on the drive flashes. (The flashing red light normally indicates a disk error, but just ignore it while Turbo Copy works.)

After about 30 seconds you'll hear an audio cue. Insert the destination disk (the disk you're copying *to*) and press RETURN. Make sure that the destination disk is blank—unless you don't mind that its contents are erased.

One minute later, you'll need to switch disks again. This happens two more times until the copy is complete, when you're asked if you want to make another copy. Type Y and Turbo Copy restarts. Any other keypress resets the machine.

Knock Knock

It may seem that Turbo Copy knocks the drive for no reason. Don't be concerned. Turbo Copy is simply aligning the read/write head after a different disk is placed in the drive.

You don't even have to format a disk before you use it with Turbo Copy. Each sector is automatically formatted as information is written to it. If you want, use a new disk right out of the box.

Turbo Copy is almost foolproof. Though all normal read errors are read by the program, Turbo Copy goes on when it finds a read error. It doesn't even stop to tell you it's found a read error (although the drive knocks). Turbo Copy can deal with such errors.

If a read error is found when reading the source disk, Turbo Copy skips over the corresponding sectors when it writes to the destination disk. If the destination disk was previously unformatted, this puts a read error on the destination disk in the same spot as on the source disk. If the disk was already formatted, the track or sector is left alone, although the IDs probably won't match. This might (though the chances are very small) cause a read error when the drive tries to access the track or sector and finds a different ID.

Writing to disk is different. The only way a write error could happen is if there were no disk in the drive, if the drive door were left open, or if a write-protect tab were on the disk. Any of these three conditions stop Turbo Copy; the program will ask you either to quit or to start again.

Turbo Work

To speed up data transfer more than four times, Turbo Copy reprograms the disk drive to send two bits of data at a time instead of one.

Normally, just a single bit is sent over the data line of the serial port. Turbo Copy sends and receives two bits of data—one on the data line, one on the clock line. More than half of Turbo Copy is sent to the disk drive and stored in its RAM. It's this data which tells the disk drive what to do.

That means the disk drive, because it's been reprogrammed, won't work correctly in normal operations after

you've quit Turbo Copy. Turn off the drive; then turn it back on again to return your drive to its normal programming.

Note: *Turbo Copy will not work with the Commodore 1571 disk drive.*

Turbo Copy

For mistake-proof program entry, be sure to use "MLX" (Appendix D).

```

0801:0B 08 0A 00 9E 32 30 36 2E
0809:31 00 00 00 A9 00 8D 20 3A
0811:D0 8D 21 D0 A9 80 8D 05 8D
0819:D4 A9 FF 8D 06 D4 A9 86 34
0821:8D 01 D4 A9 1E 8D 00 D4 69
0829:A9 6A 85 FB A9 0B 85 FC 9A
0831:20 CF 09 20 43 0A 20 49 34
0839:0A A9 1D 85 FB A9 0C 85 D8
0841:FC A2 02 20 C9 FF 20 CF 19
0849:09 20 CC FF 20 BE 09 20 AD
0851:18 0A A9 D4 85 FB A9 0A EB
0859:85 FC 20 CF 09 20 A4 09 87
0861:A9 A5 20 29 09 A9 00 A2 D7
0869:10 85 03 86 04 20 5F 09 14
0871:F0 16 08 A0 00 91 03 E6 BD
0879:03 28 30 F1 20 5F 09 91 5C
0881:03 C8 D0 F8 E6 04 D0 E5 BD
0889:20 5F 09 10 03 4C 6E 08 D1
0891:20 9A 09 20 18 0A A9 56 0E
0899:85 FB A9 0A 85 FC 20 CF 71
08A1:09 20 A4 09 20 29 09 A9 C4
08A9:00 85 03 A9 10 85 04 A0 55
08B1:00 20 5F 09 D0 17 B1 03 8F
08B9:08 20 29 09 E6 03 28 30 4F
08C1:F0 B1 03 20 29 09 C8 D0 E8
08C9:F8 E6 04 D0 E4 10 0B A5 C0
08D1:03 A6 04 85 30 86 31 4C 30
08D9:B2 08 C9 03 D0 06 20 9A 28
08E1:09 4C 50 08 C9 09 D0 0B 33
08E9:A5 30 A6 31 85 03 86 04 0A
08F1:4C B2 08 C9 24 D0 03 4C 29
08F9:0F 09 20 9A 09 20 18 0A 84
0901:A9 A9 85 FB A9 0A 85 FC 40
0909:20 CF 09 4C 20 09 20 9A 05
0911:09 20 18 0A A9 23 85 FB 34
0919:A9 0B 85 FC 20 CF 09 20 B5
0921:A4 09 20 9A 09 4C 50 08 97
0929:85 6B A9 35 85 01 A9 0B EF
0931:8D 00 DD AD 00 DD 10 FB 34
0939:A9 03 8D 00 DD A2 04 A9 BD
0941:03 46 6B 6A 46 6B 6A 4A 79

```

0949:4A EA 8D 00 DD CA D0 EF 98
0951:E6 FD A5 FD D0 03 EE 20 7B
0959:D0 A9 34 85 01 60 A9 35 2F
0961:85 01 A9 0B 8D 00 DD AD 32
0969:00 DD 10 FB A9 03 8D 00 29
0971:DD A2 05 CA EA D0 FC A2 9F
0979:04 AD 00 DD 0A 08 0A 26 81
0981:6B 28 26 6B CA D0 F2 E6 35
0989:FD A5 FD D0 03 CE 20 D0 35
0991:A9 34 85 01 A5 6B 49 FF B3
0999:60 A9 36 85 01 A9 1B 8D D7
09A1:11 D0 60 58 20 34 0A C0 A8
09A9:4E D0 03 4C E2 FC A9 0B A5
09B1:8D 11 D0 A0 20 CA D0 FD BE
09B9:88 D0 FA 78 60 A9 0A A2 8B
09C1:00 A0 00 88 D0 FD CA D0 69
09C9:F8 E9 01 D0 F2 60 A0 00 5A
09D1:B1 FB C9 A1 F0 15 C9 87 06
09D9:F0 12 C9 AB F0 2E 20 D2 30
09E1:FF 20 07 0A D0 EA E6 FC 7A
09E9:4C D1 09 60 20 07 0A B1 A0
09F1:FB AA 20 07 0A B1 FB 20 50
09F9:07 0A 84 20 A8 18 20 F0 7B
0A01:FF A4 20 4C D1 09 C8 D0 1C
0A09:E2 E6 FC 60 20 CC FF A2 C4
0A11:02 20 C9 FF 4C E2 09 A9 11
0A19:00 8D 20 D0 60 A9 0F 8D F6
0A21:18 D4 A9 11 8D 04 D4 60 43
0A29:A9 10 8D 04 D4 A9 00 8D E2
0A31:18 D4 60 20 1E 0A A9 00 01
0A39:85 C6 20 CF FF A8 20 29 CE
0A41:0A 60 A9 00 20 BD FF 60 FF
0A49:A9 02 A2 08 A0 0F 20 BA C3
0A51:FF 20 C0 FF 60 93 87 07 EC
0A59:0B 9A 70 4C 45 41 53 45 87
0A61:20 50 4C 41 43 45 20 54 FA
0A69:48 45 87 09 0B 05 44 45 AE
0A71:53 54 49 4E 41 54 49 4F 8F
0A79:4E 9A 20 44 49 53 4B 87 59
0A81:0B 0A 49 4E 54 4F 20 54 20
0A89:48 45 20 44 52 49 56 45 05
0A91:20 41 4E 44 87 0D 0D 50 EE
0A99:52 45 53 53 20 12 98 72 B4
0AA1:65 74 75 72 6E 90 0D A1 CC
0AA9:93 87 09 0D 05 64 69 73 5B
0AB1:6B 20 65 72 72 6F 72 21 AE
0AB9:21 87 0F 0B 9A 74 52 59 77
0AC1:20 41 47 41 49 4E 20 28 1F
0AC9:05 59 9A 2F 05 4E 9A 29 BC
0AD1:3F 05 A1 93 87 07 0B 9A 3D

```

0AD9:70 4C 45 41 53 45 20 50 36
0AE1:4C 41 43 45 20 54 48 45 51
0AE9:87 09 0E 05 53 4F 55 52 EA
0AF1:43 45 9A 20 44 49 53 4B 87
0AF9:87 0B 09 20 49 4E 54 4F 33
0B01:20 54 48 45 20 44 52 49 99
0B09:56 45 20 41 4E 44 87 0D 53
0B11:0D 50 52 45 53 53 20 12 9A
0B19:98 72 65 74 75 72 6E 90 EE
0B21:0D A1 93 87 06 0E 9A 54 03
0B29:48 45 20 43 4F 50 59 27 82
0B31:53 87 08 0E 05 63 6F 6D B6
0B39:70 6C 65 74 65 64 21 87 1D
0B41:09 0E 9A A3 A3 A3 A3 83
0B49:A3 A3 A3 A3 A3 87 0F 0A 2C
0B51:61 4E 4F 54 48 45 52 20 F6
0B59:43 4F 50 59 20 28 05 59 89
0B61:9A 2F 05 4E 9A 29 3F 05 13
0B69:A1 93 0E 87 06 06 12 1C F7
0B71:20 20 20 81 20 20 20 9E 1C
0B79:20 20 20 99 20 20 20 1E 25
0B81:20 20 20 9A 20 20 20 1F 3E
0B89:20 20 20 9C 20 20 20 05 4C
0B91:20 20 20 87 07 06 12 1C CC
0B99:20 74 20 81 20 75 20 9E AE
0BA1:20 72 20 99 20 62 20 1E EA
0BA9:20 6F 20 9A 20 63 20 1F 47
0BB1:20 6F 20 9C 20 70 20 05 89
0BB9:20 79 20 87 08 06 12 1C 53
0BC1:20 20 20 81 20 20 20 9E 6C
0BC9:20 20 20 99 20 20 20 1E 75
0BD1:20 20 20 9A 20 20 20 1F 8E
0BD9:20 20 20 9C 20 20 20 05 9C
0BE1:20 20 20 87 0C 04 92 98 BA
0BE9:34 2D 4D 49 4E 55 54 45 59
0BF1:20 48 49 47 48 2D 53 50 B5
0BF9:45 45 44 20 44 49 53 4B C7
0C01:20 43 4F 50 49 45 52 87 74
0C09:14 0C 90 42 59 20 61 2E A0
0C11:20 6D 2E 20 63 55 54 52 C7
0C19:4F 4E 9A A1 4D 2D 57 00 A7
0C21:03 22 A5 00 29 06 F0 11 4C
0C29:C9 02 D0 03 4C 1A 03 C9 8B
0C31:04 D0 03 4C CA 05 4C 4E F5
0C39:04 4C 9E FD A5 44 4A 4A 37
0C41:4A 4A 4A AA AB 4D 2D 57 49
0C49:22 03 22 BD 4A 04 85 0C CC
0C51:A5 77 D0 04 E6 77 A5 43 18
0C59:85 0A A9 05 85 09 A9 5A 3A
0C61:85 4B A9 52 85 24 20 56 BC

```

```

0C69:F5 50 FE B8 AD AB 4D 2D DF
0C71:57 44 03 22 01 1C C5 24 F0
0C79:F0 09 C6 4B D0 EF A9 0A 7D
0C81:4C 69 F9 A2 00 50 FE B8 7B
0C89:AD 01 1C 95 25 E8 E0 07 2B
0C91:90 F3 20 97 F4 A5 AB 4D 4F
0C99:2D 57 66 03 22 16 45 17 26
0CA1:45 18 45 19 45 1A F0 07 18
0CA9:C6 09 D0 C0 4C 1E F4 A5 F7
0CB1:18 C5 06 F0 03 4C 0B F4 6B
0CB9:85 22 A5 16 A6 17 85 AB 7B
0CC1:4D 2D 57 88 03 22 12 86 8A
0CC9:13 A6 19 E8 E8 E4 43 90 B8
0CD1:02 A2 00 A9 02 85 31 4C 03
0CD9:3D 04 CA D0 15 4C 19 04 08
0CE1:86 07 86 19 8A 45 18 45 40
0CE9:AB 4D 2D 57 AA 03 22 17 03
0CF1:45 16 85 1A 20 34 F9 A2 EC
0CF9:5A 20 56 F5 50 FE B8 AD 0F
0D01:01 1C D9 24 00 D0 DA C8 E1
0D09:C0 08 D0 F0 20 56 F5 50 45
0D11:FE AB 4D 2D 57 CC 03 22 28
0D19:B8 AD 01 1C 91 30 C8 D0 8C
0D21:F5 A0 BA 50 FE B8 AD 01 F1
0D29:1C 99 00 01 C8 D0 F4 20 5B
0D31:E0 F8 A5 38 C5 47 D0 2D 4C
0D39:20 E9 AB 4D 2D 57 EE 03 CF
0D41:22 F5 C5 3A F0 0B A6 07 4E
0D49:BD 50 01 D0 23 A9 05 D0 1E
0D51:21 E6 19 A5 19 20 63 05 48
0D59:B9 00 02 20 63 05 C8 D0 24
0D61:F7 AD 00 AB 4D 2D 57 10 7B
0D69:04 22 1C 49 08 8D 00 1C B8
0D71:A9 FF 2C A9 84 A6 07 09 56
0D79:80 9D 50 01 08 A0 00 8C A4
0D81:FE 02 28 10 07 C6 0A D0 D9
0D89:03 4C 9E FD AB 4D 2D 57 30
0D91:32 04 22 8A 18 65 0C C5 E6
0D99:43 90 02 E5 43 AA BD 50 A8
0DA1:01 10 05 E8 8A 4C 36 04 65
0DA9:4C A2 03 09 09 0A 0B A5 AF
0DB1:44 4A 4A 4A 4A AB 4D 2D 37
0DB9:57 54 04 22 4A AA BD C3 73
0DC1:05 85 0C A0 00 84 0A A2 14
0DC9:00 A5 39 99 00 02 C8 C8 70
0DD1:A5 0A 99 00 02 C8 A5 06 F8
0DD9:99 00 02 C8 A5 13 AB 4D AB
0DE1:2D 57 76 04 22 99 00 02 F0
0DE9:C8 A5 12 99 00 02 C8 A9 F0
0DF1:0F 99 00 02 C8 99 00 02 C8

```

0DF9:C8 B9 FA 01 59 FB 01 59 6C
0E01:FC 01 59 FD 01 99 F9 AB F4
0E09:4D 2D 57 98 04 22 01 E6 1D
0E11:0A A5 0A C5 43 90 C0 98 AF
0E19:48 A9 02 85 31 20 30 FE C5
0E21:68 A8 88 B9 00 02 99 45 C8
0E29:02 88 D0 F7 AD 00 02 8D 01
0E31:AB 4D 2D 57 BA 04 22 45 01
0E39:02 20 F5 FD A9 00 85 32 87
0E41:AD 0C 1C 29 1F 09 C0 8D 79
0E49:0C 1C A9 FF 8D 03 1C A9 02
0E51:55 8D 01 1C A2 1E 20 24 4F
0E59:FE AB 4D 2D 57 DC 04 22 B4
0E61:A9 FF 8D 01 1C A2 05 50 D9
0E69:FE B8 CA D0 FA A2 0A A4 B4
0E71:32 50 FE B8 B9 00 02 8D 85
0E79:01 1C C8 CA D0 F3 A2 09 87
0E81:50 FE AB 4D 2D 57 FE 04 98
0E89:22 B8 A9 55 8D 01 1C CA E2
0E91:D0 F5 A9 FF A2 05 50 FE 91
0E99:B8 8D 01 1C CA D0 F7 A9 8A
0EA1:55 50 FE B8 8D 01 1C A2 33
0EA9:BC 50 FE AB 4D 2D 57 20 C0
0EB1:05 22 B8 A9 DD 8D 01 1C CD
0EB9:E8 D0 F5 A0 00 50 FE B8 3F
0EC1:A9 DD 8D 01 1C 88 D0 F5 86
0EC9:A9 55 A6 0C 50 FE B8 8D 23
0ED1:01 1C CA D0 AB 4D 2D 57 20
0ED9:42 05 22 F7 A5 32 18 69 AB
0EE1:0A 85 32 C6 0A D0 8E 50 18
0EE9:FE B8 50 FE B8 20 00 FE F2
0EF1:A9 00 8D FE 02 85 50 A9 F4
0EF9:01 4C 69 F9 85 AB 4D 2D 19
0F01:57 64 05 22 08 2C 00 18 AF
0F09:10 FB A9 10 8D 00 18 2C 2D
0F11:00 18 30 FB A2 04 A9 00 73
0F19:06 08 2A 0A 06 08 2A 0A D0
0F21:8D 00 18 CA D0 F0 AB 4D A4
0F29:2D 57 86 05 22 A2 01 CA 3D
0F31:D0 00 EA EA A9 0F 8D 00 68
0F39:18 60 2C 00 18 10 FB A9 A3
0F41:10 8D 00 18 2C 00 18 30 0E
0F49:FB A2 04 CA D0 FD 8E AB 82
0F51:4D 2D 57 A8 05 22 00 18 9F
0F59:A2 04 AD 00 18 4A 08 4A C3
0F61:4A 66 08 28 66 08 CA D0 7B
0F69:F1 A9 0F 8D 00 18 A5 08 59
0F71:60 08 0B 10 07 4C 62 06 58
0F79:AB 4D 2D 57 CA 05 22 A5 30
0F81:43 85 0A 20 00 FE A9 00 35

0F89:20 63 05 EA 20 93 05 30 69
0F91:EB AA CA 86 07 A9 02 85 7A
0F99:31 A0 00 20 93 05 99 00 5E
0FA1:02 AB 4D 2D 57 EC 05 22 C2
0FA9:C8 D0 F7 20 E9 F5 85 3A CD
0FB1:20 8F F7 AD FE 02 85 7C 25
0FB9:A9 00 8D FE 02 85 50 A5 BA
0FC1:0F F0 03 4C B1 07 20 10 C2
0FC9:F5 A2 AB 4D 2D 57 0E 06 BE
0FD1:22 09 50 FE B8 CA D0 FA CA
0FD9:A9 FF 8D 03 1C AD 0C 1C 7A
0FE1:29 1F 09 C0 8D 0C 1C A9 08
0FE9:FF A2 05 8D 01 1C B8 50 64
0FF1:FE B8 CA AB 4D 2D 57 30 CF
0FF9:06 22 D0 FA A0 BB B9 00 D4
1001:01 50 FE B8 8D 01 1C C8 92
1009:D0 F4 A5 7C 8D FE 02 B1 69
1011:30 50 FE B8 8D 01 1C C8 3A
1019:D0 F5 50 FE AB 4D 2D 57 5D
1021:52 06 22 20 00 FE AD 00 89
1029:1C 49 08 8D 00 1C A9 00 47
1031:8D FE 02 C6 0A F0 03 4C EA
1039:D1 05 4C 9E FD 78 A9 07 23
1041:85 47 A2 00 CA AB 4D 2D 17
1049:57 74 06 22 D0 FD A9 01 E7
1051:85 06 A9 C0 20 A8 07 A9 52
1059:E0 20 A8 07 20 93 05 A5 76
1061:06 85 0D A9 00 20 63 05 6E
1069:A5 06 09 80 20 63 AB 4D 3A
1071:2D 57 96 06 22 05 20 9D 34
1079:07 A9 00 85 78 85 83 85 46
1081:77 85 10 A9 E2 20 A8 07 4B
1089:C9 02 90 16 A5 10 D0 12 A3
1091:A9 C0 20 A8 07 A9 B0 AB 31
1099:4D 2D 57 B8 06 22 20 A8 C3
10A1:07 E6 10 A9 00 85 78 85 28
10A9:83 A2 00 86 09 BD 50 01 7D
10B1:C9 FF D0 08 E8 E4 43 D0 83
10B9:F4 4C F5 06 A4 78 B9 DB DC
10C1:AB 4D 2D 57 DA 06 22 FE 58
10C9:F0 0A A9 00 9D 50 01 E6 31
10D1:09 4C CE 06 86 07 E8 8A 70
10D9:09 80 20 63 05 A6 07 4C F5
10E1:CE 06 A5 09 F0 17 85 77 96
10E9:A5 AB 4D 2D 57 FC 06 22 21
10F1:83 38 F9 DB FE 85 83 B9 AD
10F9:DB FE 58 20 76 D6 78 E6 BB
1101:78 4C A4 06 A5 83 58 20 73
1109:76 D6 78 E6 06 A5 06 C9 36
1111:0C F0 AB 4D 2D 57 1E 07 C9

1119:22 0B C9 17 F0 07 C9 24 15
1121:F0 03 4C 8B 06 A9 00 20 B5
1129:63 05 A9 01 20 63 05 EA 07
1131:20 93 05 A5 0D 85 06 A9 77
1139:C0 20 A8 AB 4D 2D 57 40 A1
1141:07 22 07 A5 06 09 80 20 20
1149:63 05 A9 05 85 0E A9 09 A4
1151:20 63 05 A9 E6 20 A8 07 A7
1159:A9 E4 A2 00 86 0F 20 A8 37
1161:07 C9 02 B0 AB 4D 2D 57 09
1169:62 07 22 10 A9 09 20 63 D8
1171:05 E6 0F A9 E4 20 A8 07 4C
1179:C9 02 90 09 C6 0E D0 D5 89
1181:A9 05 4C 97 07 E6 06 A5 42
1189:06 C9 0C F0 0B AB 4D 2D 80
1191:57 84 07 22 C9 17 F0 07 17
1199:C9 24 F0 0B 4C 41 07 A9 97
11A1:03 20 63 05 4C 84 06 20 AA
11A9:63 05 4C 71 06 A9 00 A2 D8
11B1:14 9D 50 01 CA 10 AB 4D 9A
11B9:2D 57 A6 07 22 FA 60 85 D0
11C1:00 58 A5 00 30 FC 78 60 75
11C9:20 0A F5 A0 BB B9 00 01 0D
11D1:50 FE B8 4D 01 1C D0 1A FB
11D9:C8 D0 F2 A5 7C 8D FE AB 11
11E1:4D 2D 57 C8 07 22 02 B1 E3
11E9:30 50 FE B8 4D 01 1C D0 1B
11F1:08 C8 C0 FD D0 F1 4C 55 7E
11F9:06 A9 00 8D FE 02 4C C5 C0
1201:F6 00 00 00 00 00 00 A0
1209:AB 4D 2D 45 6C 06 0D A1 87

Directory Extension

Kevin Mykytyn and Mark Tuttle

Have you ever looked at a cryptic filename like ZYM/414/LUB and wondered what it was and where it came from? Have you ever forgotten the SYS for a favorite program? This program will help; it adds up to 55 characters per directory entry—plenty of room for descriptions and information.

Commodore disk drives limit filenames to 16 characters, which sometimes just isn't enough. "Directory Extension" lets you add a 55-character extension to any or all files on a disk. After creating the descriptions, simply save them on the disk they describe. Then, whenever you want to see the descriptions, load and run Directory Extension. You can update or modify this file as often as you like.

Easy Operation

Directory Extension is written entirely in BASIC. Type it in and save a copy. To use it, type **RUN**. The program first looks for the description file for the disk in the drive. If one exists, it's loaded into memory. Next, the directory of the disk is read and you're presented with a menu.

The menu offers five options. Option 1 allows you to view the directory and enter or edit the descriptions. To the left is the file number. In inverse black is the filename, and the inverse white area is where you enter the descriptions. At the bottom of each screen are five additional options, each chosen when you press the appropriate function key. If the disk contains more than six files, use the f5 and f7 keys to page forward and backward, respectively. The f3 key allows you to edit the descriptions.

After pressing f3 to select a file, enter the file number. The cursor moves to the beginning of the white field next to the file you have chosen. (If you mistakenly choose the wrong file, press RETURN and all will remain intact.) You may now type up to 55 characters. Any character is allowed except quotation marks. The cursor automatically wraps around at the

end of a line. Use the INST/DEL key to erase. When you've finished the description or entered the maximum number of characters, press RETURN. You can enter and change descriptions as many times as you like. When you've finished the descriptions, return to the main menu by pressing f1. Choose option 3 and the descriptions will be saved on the disk with the filename DIR FILE. Be sure you have enough room on the disk for this file.

Option 2 lets you perform various disk functions—scratch, rename, initialize, and validate. Option 3 saves the descriptions to the disk. If DIR FILE already exists on the disk, the program will erase the file and replace it with what's in memory. Option 4 allows you to work with other disks without having to rerun Directory Extension. Option 5 ends the program.

Directory Extension

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

```

AB 100 POKE 53280,15:POKE 53281,15:PRINT "{CLR}
      {WHT}{6 DOWN}{10 SPACES}DIRECTORY EXTENSION
      "
XS 110 PRINT "{10 SPACES}{19 T}{5 DOWN}"
KR 120 FOR I=830 TO 902:READ D:POKE I,D:NEXT
KC 130 DN$="{HOME}{21 DOWN}":P$="{P}{DOWN}{2 LEFT}
      {N}{RVS}{OFF}{H}{DOWN}{2 LEFT}{Y}":XX=1
HE 140 MT$="{RVS}{WHT}{19 SPACES}{OFF}{4 SPACES}
      {RVS}{36 SPACES}{BLK}"
PD 150 DM=144:DIM F$(DM),S$(DM),T$(DM),S2$(DM)
RD 160 FOR I=0 TO DM:F$(I)=CHR$(32)+"{18 SPACES}":
      NEXT
BS 170 PRINT "{UP}{11 SPACES}READING DIRECTORY
      {2 SPACES}":OPEN1,8,0,"$":SYS 830:CLOSE 1
FB 180 GOSUB 950
QR 190 POKE 198,0:PRINT "{CLR}{DOWN}{WHT}"SPC(9)P$"
      {UP}{LEFT}{RVS}1{OFF}"SPC(3)"VIEW DIRECTORY
      FILE"
CE 200 PRINT "{2 DOWN}"SPC(9)P$"{UP}{LEFT}{RVS}2
      {OFF}"SPC(3)"SEND DISK COMMANDS"
DH 210 PRINT "{2 DOWN}"SPC(9)P$"{UP}{LEFT}{RVS}3
      {OFF}"SPC(3)"SAVE"
HK 220 PRINT "{2 DOWN}"SPC(9)P$"{UP}{LEFT}{RVS}4
      {OFF}"SPC(3)"VIEW DIFFERENT DISK"
PB 230 PRINT "{2 DOWN}"SPC(9)P$"{UP}{LEFT}{RVS}5
      {OFF}"SPC(3)"QUIT"
JB 240 PRINT "{2 DOWN}"SPC(28)P$"{UP}{LEFT}{RVS}
      {OFF}":PRINT "{UP}"SPC(9)"ENTER YOUR CHOICE
      {SPACE}{N}{B}{LEFT}";

```

```

GQ 250 POKE 204,0:GET CG$:IF CG$<"1" OR CG$>"5" TH
    EN 250
BR 260 POKE 204,1:POKE 207,0:PRINT"{RVS}"CG$:CG=VA
    L(CG$)
EF 270 ON CG GOTO 290,1090,520,1240,1340
ER 280 REM *** VIEW FILES ***
XP 290 C=PEEK(0)-1:A=1
EJ 300 IF A=> C THEN 330
AH 310 IF A<0 AND C/6=INT(C/6) THEN A=C-5:GOTO 330
SB 320 IF A<0 THEN DN=INT(C/6):A=DN*6+1
HQ 330 PRINT"{CLR}{DOWN}{BLK}{OFF}{3 SPACES}DISK T
    ITLE: {RVS}"F$(0)" ":PRINT"{DOWN}":PQ=5
HG 340 FOR I=A TO A+PQ
CX 350 IF F$(I)=CHR$(32)+"{18 SPACES}" OR F$(I)="
    THEN F$(I)="" :PQ=PQ-1
RJ 360 NEXT
PE 370 FOR I= A TO A+PQ
DE 380 IF I>9 THEN W$=" {LEFT}{RVS}":GOTO 400
GR 390 W$="{2 SPACES}{LEFT}{RVS}"
FE 400 PRINT"{LEFT}"I;W$;LEFT$(F$(I),16)": "MT$:PRI
    NT"{2 UP}"SPC(20);
FA 410 PRINT"{RVS}{WHT}"S$(I)"{2 DOWN}{BLK}":NEXT
AJ 420 PRINT DN$"{3 DOWN} F1-MENU F3-ADD F5-FWD F7
    -BACK ↑-DELETE{HOME}":POKE 198,0:WAIT 198,1
AA 430 IF PEEK(197)=54 THEN 1280
SH 440 IF PEEK(197)=4 THEN 190
BF 450 IF PEEK(197)=5 THEN 610
MS 460 IF PEEK(197)=6 AND A=C THEN A=1:GOTO 300
RM 470 IF PEEK(197)=6 AND A+6>C THEN A=1:GOTO 300
JA 480 IF PEEK(197)=6 AND A<C THEN A=A+6:GOTO 300
FP 490 IF PEEK(197)=3 THEN A=A-6:GOTO 300
DC 500 GOTO 430
XJ 510 REM *** SAVE DESCRIPTIONS ***
RA 520 PRINT"{2 UP}"SPC(28)" {LEFT}{2 DOWN} ":PRIN
    T"{2 UP}"SPC(9)"{2 SPACES}SAVING "CHR$(34)"
    DIR FILE";
KB 530 PRINT CHR$(34)"...":FOR R=1 TO C:IF S$(R)="
    " THEN S$(R)="{RVS} "
MQ 540 NEXT R
XG 550 CLOSE1:CLOSE 15:OPEN 1,8,15,"S0:DIR FILE":C
    LOSE 1:OPEN 15,8,15,"I "
CH 560 OPEN 1,8,8,"0:DIR FILE,S,W":GOSUB 1050
JX 570 PRINT#1,C
JA 580 FOR I=1 TO C:PRINT#1,LEFT$(F$(I),16):PRINT#
    1,S$(I):NEXT:CLOSE 1
CD 590 PRINT"{UP}{31 SPACES}":GOTO 190
JD 600 REM *** ADD STATEMENTS ***
HX 610 PRINTDN$"{3 DOWN}"SPC(9)"{RVS}F3-ADD{HOME}"
KK 620 PRINT DN$"{DOWN}{3 SPACES}ENTER FILE

```

```

      {SHIFT-SPACE}#";:POKE 198,0:INPUT FI$:FI=VAL(FI$)
CX 630 IF FI$="" THEN PRINTDN$"{DOWN}{{18 SPACES}}"
      :GOTO 420
KA 640 IF FI<A OR FI>A+PQ THEN PRINT DN$ SPC(17)"
      {DOWN}{{18 SPACES}}":GOTO 610
XM 650 PRINT DN$"{DOWN}{{26 SPACES}}":IF FI<7 THEN Z
      =FI:GOTO 680
GA 660 Z=INT(FI/6)+1:DD=Z*6-FI:Z=6-DD
DD 670 IF Z=0 THEN Z=6
EX 680 PRINT"{HOME}{{DOWN}}":FORI=1 TO Z:PRINT"
      {DOWN}":NEXT
HE 690 F$(FI)=LEFT$(F$(FI),16)
SC 700 I$=""
GQ 710 PRINT"{UP}"SPC(19)"{RVS}>{WHT}"SPC(LEN(I$))
      ;:B$="<":CC=LEN(B$):L=1
PQ 720 PRINT"{RVS}"MID$(B$,L,2);"{LEFT}";:L=-L*(L<
      CC)+1
FG 730 GET A$:IF A$="" OR A$=CHR$(34) OR A$=CHR$(4
      4) OR A$=CHR$(58) THEN 730
GA 740 IF A$=CHR$(59) THEN 730
GQ 750 IF A$=CHR$(13) AND LEN(I$)=18 THEN 910
PF 760 IF LEN(I$)=18 AND A$<>CHR$(20) THEN I$=I$+A
      $:PRINT A$"{OFF}{{4 SPACES}}{RVS}<{LEFT}";:GO
      TO 720
BJ 770 IF LEN(I$)=55 THEN PP=1
JC 780 IF LEN(I$)=55 AND A$<>CHR$(20) AND A$<>CHR$
      (13) THEN 730
QF 790 IF A$=CHR$(13) AND LEN(I$)=0 THEN 330
EM 800 IF A$=CHR$(13) THEN 910
XX 810 IF A$=CHR$(20) AND LEN(I$)=19 THEN 880
HF 820 IF A$=CHR$(20) AND LEN(I$)>0 THEN 850
EQ 830 IF ASC(A$)<32 OR ASC(A$)>95 THEN 720
AK 840 I$=I$+A$:PRINT A$;:GOTO 720
EE 850 I$=LEFT$(I$,LEN(I$)-1)
FQ 860 IF LEN(I$)=54 THEN PRINT"{OFF} {RVS}
      {2 LEFT}";:GOTO 720
PA 870 PRINT"{RVS} {2 LEFT}";:GOTO 720
HE 880 I$=LEFT$(I$,LEN(I$)-1):PRINT"{RVS} ":D=19
DQ 890 IF PP=1 THEN PRINT"{3 UP}"SPC(38)";:GOTO 7
      20
CC 900 PRINT"{2 UP}"SPC(38)";:GOTO 720
FK 910 S$(FI)=I$
SX 920 IF LEN(S$(FI))<19 THEN 330
ME 930 S$(FI)=MID$(S$(FI),1,19)+"{OFF}{{4 SPACES}}
      {RVS}"+MID$(S$(FI),20,37)
SM 940 S$(FI)=S$(FI)+"{UP}":GOTO 330
JH 950 OPEN 15,8,15,"I"
EK 960 OPEN 1,8,8,"0:DIR FILE,S,R":GOSUB 1050:IF E
      N=62 THEN CLOSE 15:RETURN

```

```

HF 970 PRINT "{UP}"SPC(19)"DIR FILE{6 SPACES}?"
BM 980 INPUT#1,C
QC 990 FOR I=1 TO C:INPUT#1,T$(I):INPUT#1,S$(I):S2$(I)=S$(I):NEXT
RR 1000 CLOSE 1:CLOSE 15:NF=PEEK(0)-1
QP 1010 FOR A=1 TO NF:IF LEFT$(F$(A),16)=T$(A) THEN S$(A)=S2$(A):GOTO 1040
PR 1020 FOR B=1 TO C:IF LEFT$(F$(A),16)=T$(B) THEN S$(A)=S2$(B):B=C+2
AD 1030 NEXT B:IF B=C+1 THEN S$(A)=""
QD 1040 NEXT A:RETURN
RQ 1050 INPUT#15,EN,EM$:IF EN=0 THEN RETURN
CX 1060 IF EN=62 AND XX=1 THEN XX=0:RETURN
CH 1070 PRINT "{CLR}{4 DOWN}DISK ERROR:{RVS}";EN;EM$:PRINT "{HOME}{4 DOWN}"SPC(14)"{RVS} ":END
EM 1080 REM *** SEND DISK COMMANDS ***
QG 1090 PRINT "{CLR}{2 DOWN} DISK COMMANDS":PRINT "{2 DOWN} COMMANDS":PRINT "{SHIFT-SPACE}{8 T}"
RS 1100 PRINT "{DOWN} S0:FILENAME"SPC(11)"SCRATCH A FILE
AQ 1110 PRINT "{DOWN} R0:NEWNAME=OLDNAME{4 SPACES}R ENAME A FILE
RM 1120 PRINT "{DOWN} V0"SPC(20)"VALIDATE A DISK":PRINT "{DOWN} I0"SPC(20)"INITIALIZE"
QF 1130 PRINT "{2 DOWN} ENTER COMMAND{3 SPACES}(RETURN FOR MENU)":Z$="":PRINT " >";
AQ 1140 GET U$:IF U$="" THEN 1210
JB 1150 IF LEN(Z$)=37 AND U$<>CHR$(13) AND U$<>CHR$(20) THEN 1140
AQ 1160 U=ASC(U$):IF U=13 THEN 1220
DH 1170 IF U<>20 THEN 1200
AJ 1180 IF LEN(Z$)<>0 THEN Z$=MID$(Z$,1,LEN(Z$)-1):PRINT U$;
CR 1190 GOTO 1140
RG 1200 IF U<32 OR U>90 THEN 1140
AB 1210 PRINT "<{LEFT}";U$;:Z$=Z$+U$:GOTO 1140
CE 1220 IF Z$="" THEN 190
GA 1230 CLOSE1:OPEN 1,8,15,Z$:CLOSE 1:CLOSE 15:OPEN 15,8,15,"I":GOSUB 1050:GOTO190
JD 1240 PRINT "{CLR}{8 DOWN}{7 SPACES}INSERT NEW DISK INTO DRIVE"
AX 1250 PRINT "{2 DOWN}{7 SPACES}PRESS ANY KEY TO CONTINUE":POKE 198,0:WAIT 198,1:GET E$
GE 1260 PRINT "{UP}{13 SPACES}PLEASE WAIT{8 SPACES}":CLR:GOTO 130
EE 1270 REM *** DELETE DESCRIPTIONS ***
KX 1280 PRINTDN$"{3 DOWN}"SPC(31)"{RVS}{↑-DELETE}{HOME}"
SF 1290 PRINT DN$"{DOWN}"SPC(10)"{29 SPACES}"

```

```
GK 1300 POKE 198,0:FI$="":PRINT DN$"{DOWN}
      {2 SPACES}DELETE #";:INPUT FI$:FI=VAL(FI$)
EA 1310 IF FI$="0" THEN PRINT DN$"{DOWN}
      {14 SPACES}":GOTO 420
QX 1320 IF FI<A OR FI>A+PQ THEN 1280
JF 1330 S$(FI)="{RVS} " :GOTO 330
EP 1340 POKE 198,0:SYS 198:END
GB 1350 DATA 169,255,133,0,165,55,133,71,165,56,13
      3
QG 1360 DATA 72,162,1,32,198,255,166,0,232,134,0
FD 1370 DATA 216,56,165,71,233,20,133,71,176,5,166
AR 1380 DATA 72,202,134,72,32,207,255,164,144,208
MQ 1390 DATA 22,201,34,208,245,160,0,32,207,255,20
      1
PC 1400 DATA 34,240,249,145,71,200,192,19,208,242
ME 1410 DATA 240,207,32,204,255,96,0,0,0
```

Super Synth

Lawrence Cotton

Your 64 never sounded this much like a real synthesizer. With a playable keyboard and saveable sounds, this innovative program is fun for the novice and educational for the more serious music programmer.

No two electronic keyboards—whether they're synthesizers or computer-based keyboards—are exactly alike. But electronic keyboards have one thing in common: They're programmable. The sounds are controlled by turning knobs, sliding controls, or pressing buttons in various combinations.

This one's a bit different. In "Super Synth," the settings are controlled completely by the computer. You play notes on the keyboard, but the sound waveforms and modifiers are generated completely at random by the computer. You have absolutely *no* control over the sounds—but you'll be amazed at some of the sounds it *can* create.

The basis of these thick, animated sounds is two voices playing slightly out of tune with one another. So even the less complex sounds have richer nuances than would be possible with single-voice sounds. Add ring modulation, synchronization, vibrato, and sweep (the last two courtesy of voice 3), and you have some really nice—and some not so nice—sounds.

The computer doesn't care whether the sounds are musically pleasing or not, so some will be downright awful. Depending on your musical perspective, the percentage of musically acceptable sounds ranges from about 10 to 50 percent.

Recording the Sounds You Want

When you get a nice sound with Super Synth, you can take a "snapshot" of it as you hear it. As you build a library of favorite sounds, you can load them from disk to play on the keyboard.

Seven waveforms are used: triangle, sawtooth, square, noise, ring modulation, synchronization, and the unlikely combination of square/triangle/ring modulation. Two of

these waveforms are independently assigned to the first two voices, but occasionally voice 2 is turned off completely to allow the sweeping effect to become more obvious. Voice 3 is used only for modulation purposes.

Type in the program, save it to disk, and run it. After about a ten-second delay while frequencies are being calculated, the keyboard is displayed. Its pitches are continuous, but physically interrupted. That is, the ↑ and the Z are musically adjacent white notes on the piano. Q is the lowest note, and ; is the highest.

Most electronic keyboards depend on the user to set up the first sound, but this one defaults to (begins with) a good basic synthesizer-like sound. To generate a new sound, press f3. Random values will be sent to the sound chip. Now play a few notes to hear the sound. If you don't like it, press f3 again.

If you want to save or load a sound, press f5 or f7 respectively. (Rather than trying to coin names for each of these

Violin Lessons

When writing this article, the author stumbled on a "patch" (synthesizer jargon for a group of switch, knob, and slider settings) which very closely simulates the sounds of a violin played by an amateur. The patch is extraordinarily (and humorously) realistic.

Here's how to create and save "Violin Lessons":

1. Type in and save the following program, "Sound Patcher," to disk. When you run it, it will create a short file on your disk called VIOLIN LESSONS.

Sound Patcher

```
100 POKE53281,0:PRINTCHR$(147)CHR$(5)
110 S$="VIOLIN LESSONS":OPEN1,8,1,S$
120 FOR T=1 TO 17:READ V:PRINT#1,V:NEXT:CLOSE1:END
130 DATA 2,1,33,33,10,8,8,8,240,1,9,17,8,8,8,31,
    0
```

2. Load and run Super Synth. Confirm that sound is there by pressing a few keys.
3. Press f7, type VIOLIN LESSONS, and press RETURN.
4. Play the keyboard in the usual way, especially the sequence of letters X, V, N, B, V, B, C, and V.

sounds, consider just numbering them consecutively.) Pressing f1 at any time returns the computer to the beginning sound.

The only input from the user is from the menu on the screen with the keyboard. Besides the note keys themselves, the only keys used are the function keys (mentioned above) and the RETURN key, which displays the values of the sound currently being heard. If you're experienced at programming the SID chip, you can use these values in your own programs.

Program Structure

Lines 100–210 calculate the frequencies needed to produce pitches ranging from C3 (Q key) to D#6 (; key). Each note, for each voice, requires two values (high and low bytes of the note's decimal frequency). These values are calculated and stored in arrays F1 and F2.

To achieve the richer basic sounds, voice 1's frequencies are multiplied by a factor M (1.005) for voice 2, which is pitched just slightly higher than voice 1. These are stored in arrays F3 and F4. Arrays G1–G4 and H1–H4 contain values for frequencies one octave higher and lower, respectively.

Lines 220–240 define constants, initialize variables, and GOSUB to line 490, a routine to print the keyboard screen. Lines 250–270 POKE the default values to the SID chip.

Lines 280–460 are the heart of the program. They form a continuous loop which looks at memory location 197—the key currently being pressed. When the value there is less than 8 (RETURN or the function keys), control goes to line 470. (The INST/DEL and cursor keys also return values less than 8, but the program just loops back to 280 when they are pressed.)

The secret to quick keyboard response is to have all the frequencies prestored in arrays whose “addresses” correspond to the keyboard matrix values PEEKed at 197. When a key is pressed, the precalculated frequency values are POKEd to the SID chip and the voice is turned on.

Line 300 sends control on to other lines, depending on the random value of Z. The random value of FL further modifies the path the program takes. Two key memory locations to note are V+27 (54299) in line 400 and V+22 (54294) in line 410. Location 54294 controls the cutoff frequency of the programmable filter for sweeping sounds. Location 54299 reads voice 3's waveform and thus controls vibrato.

Lines 340 and 350 POKE one of voice 3's frequency registers with a multiple of voice 1's frequency; this controls the synchronized and ring-modulated sounds if waveforms 21, 23, or 85 are randomly selected.

Lines 310, 320, and 460 POKE the arrays' values to the frequency registers to produce sounds one octave higher, one octave lower, and at the basic pitches, respectively. Lines 370 and 430 turn the notes on and off.

Line 470 sends control to other parts of the program, depending on whether the RETURN or function keys are pressed. The subroutine at line 480 calculates envelope values (from attack, decay, sustain, and release values) to be POKEd into voices 1 and 2. Voice 3's envelope generator is not used.

Lines 490–630 print the keyboard screen.

Lines 640–680 contain 40 pairs of data units: the key matrix location (PEEKed at 197) and the corresponding note's decimal frequency.

Lines 690–1040 are the randomizing subroutine. The choices yield six values for Z; three values for FL; 255 step limits; seven waveforms for each of voices 1 and 2; 33,750 different envelope values (the same values are POKEd into voices 1 and 2; attack is limited to ten values); four filter combinations; 40 sweep step sizes; four vibrato shapes; 145 vibrato speeds; eight different square wave shapes for each of the three voices (voice 3's waveform controls the shape of the vibrato); and three different filters (low-pass, mid-pass, and high-pass). Unfortunately, not all of these millions of combinations will produce audible differences in the sounds.

Lines 1050–1190 print the values to screen and wait for you to press RETURN. These values are printed on the screen in the same order that a sound is saved to disk.

The values for Z cause the following effects: 1—octave up; 2—unison; 3—ring modulation; 4—sync; 5—repeat; and 6—octave down. The values for FL are: 0—normal; 1—vibrato; and 2—sweep. The four resonance values POKEd to 54295 are: 240—no voice is filtered; 241—voice 1 is filtered; 242—voice 2 is filtered; and 243—both voices are filtered.

Step limit (SL) and sync speed (XT) are used in line 380. ADSR values are not POKEd directly to the sound chip; they are used to calculate POKeable values (AD and SR) in the subroutine at line 480. Vibrato speed (VI) is POKEd to 54286; vibrato shape (VS) is POKEd to 54290; and the filter value (VO), which includes full volume of 15, is POKEd to 54296.

Lines 1200–1310 contain the subroutines for saving and loading sounds to and from disk. Note that there are *no* error-trapping routines here, so be careful what you type when saving to or loading from the disk.

Super Synth

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

```

RP 100 POKE53281,0:POKE53280,0:PRINT"{CLR}":POKE21
    4,10
JR 110 PRINTTAB(9)"{DOWN}{YEL}WELCOME TO SUPER-SYN
    TH"
RM 120 PRINT:PRINTTAB(3)"WHILE WAITING FOR FREQUEN
    CIES TO BE
MP 130 PRINT:PRINTTAB(5)"CALCULATED, TURN UP THE V
    OLUME.":K=256:CH=35
SF 140 M=1.005:MA=64:DIMF1(MA),F2(MA),F3(MA),F4(MA
    ),G1(MA),G2(MA),G3(MA),G4(MA)
KK 150 DIMH1(MA),H2(MA),H3(MA),H4(MA)
QG 160 PRINT:READT,N:N1=N*M:N2=N*2:N3=N*2*M:N4=INT
    (N/2):N5=INT(N/2*M)
SB 170 F1(T)=INT(N/K):F2(T)=N-(F1(T)*K):F3(T)=INT(
    N1/K):F4(T)=INT(N1-(F3(T)*K))
BD 180 G1(T)=INT(N2/K):G2(T)=N2-(G1(T)*K):G3(T)=IN
    T(N3/K):G4(T)=INT(N3-(G3(T)*K))
SB 190 H1(T)=INT(N4/K):H2(T)=N4-(H1(T)*K):H3(T)=IN
    T(N5/K):H4(T)=INT(N5-(H3(T)*K))
EH 200 IFT<>CHTHEN160
RR 210 IFCH<>50THENFORT=1TO10:PRINT:NEXT:PRINTTAB(
    16)"THANKS!":CH=50:GOTO160
SC 220 NF=8:NK=64:KB=197:V=54272:V1=V+1:V2=V:V3=V+
    8:V4=V+7:RN=RND(-TI)
XM 230 FL=0:DB=8:DC=8:DD=8:VO=31:VS=17:VI=90:XT=1:
    PO=240:Z=2:W1=33:W2=33:AT=8:DE=8
RS 240 SU=8:RE=8:GOSUB490
RX 250 GOSUB480:FORT=VTOV+23:POKET,0:NEXT:POKEV+5,
    AD:POKEV+6,SR:POKEV+12,AD
CC 260 POKEV+13,SR:POKEV+3,DB:POKEV+10,DC:POKEV+17
    ,DD:POKEV+14,VI:POKEV+18,VS
SB 270 POKEV+23,PO:POKEV+24,VO
RP 280 T=PEEK(KB):IFT=NKTHEN280
SR 290 IFT<NFTHEN470
QF 300 ONZGOTO310,330,340,350,360,320
GB 310 POKEV1,G1(T):POKEV2,G2(T):POKEV3,G3(T):POKE
    V4,G4(T):GOTO370
RK 320 POKEV1,H1(T):POKEV2,H2(T):POKEV3,H3(T):POKE
    V4,H4(T):GOTO370
BX 330 GOSUB460:GOTO370
SQ 340 GOSUB460:POKEV+15,F1(T)/.7:GOTO370

```

```

MD 350 GOSUB460:POKEV+15,F1(T)/2:GOTO370
GK 360 GOSUB460:POKEV+4,W1:POKEV+11,W2:FOR Y=1 TO 10:
      NEXT:GOTO430
MA 370 POKEV+4,W1:POKEV+11,W2
AD 380 IFZ=4 THEN FORU=1 TO SLSTEPXT:POKEV+1,U:IF PEEK(
      KB)=T THEN NEXT
GF 390 IFZ=4 THEN GOSUB440:GOTO430
KD 400 IFFL=1 THEN POKEV,PEEK(V+27):POKEV+7,PEEK(V+2
      7):GOTO420
MJ 410 IFFL=2 THEN FORU=1 TO SLSTEP10:POKEV+22,U:IF PEE
      K(KB)=T THEN NEXT:GOSUB440:GOTO430
FD 420 IF PEEK(KB)=T THEN 380
QE 430 POKEV+4,W1-1:POKEV+11,W2-1:POKEV+15,0:GOTO2
      80
AB 440 IF PEEK(KB)<>N THEN 440
FC 450 RETURN
XC 460 POKEV1,F1(T):POKEV2,F2(T):POKEV3,F3(T):POKE
      V4,F4(T):RETURN
GC 470 ONT+1 GOTO280,1050,280,1260,230,690,1200,280
CM 480 AD=AT*16+DE:SR=SU*16+RE:RETURN
EF 490 POKE53280,0:POKE53281,0:PRINT" {CLR} {YEL}"
GA 500 PRINTTAB(12)"KEYBOARD SCREEN{2 DOWN}"
BS 510 PRINTTAB(13)"F1 - NORMAL":PRINTTAB(13)"F3 -
      NEW SOUND
KM 520 PRINTTAB(13)"F5 - SAVE SOUND
JP 530 PRINTTAB(13)"F7 - LOAD SOUND{2 DOWN}{WHT}"
HJ 540 PRINTTAB(5)"[M]{RVS} {RIGHT} {RIGHT} -
      {RIGHT} {RIGHT} {RIGHT} - {RIGHT} {RIGHT} B
      {RIGHT} {RIGHT} {OFF}C{RVS}"
BE 550 PRINTTAB(5)"[M]{RVS} {OFF}2{RVS} {OFF}3
      {RVS} - {OFF}5{RVS} {OFF}6{RVS} {OFF}7{RVS}
      - {OFF}9{RVS} {OFF}0{RVS} - {OFF}-{RVS}
      {OFF}£{RVS} {OFF}H{RVS}"
JA 560 PRINTTAB(5)"[M]{RVS} _ _ _ _ _ _ _ _ _ _
      {SPACE}- - {OFF} {YEL}*I{WHT}"
KD 570 PRINTTAB(5)"[M]{RVS}Q-W-E-R-T-Y-U-I-O-P-@-*
      -↑-Z{OFF}{2 SPACES}{YEL}B"
GF 580 PRINTTAB(35)"B":PRINTTAB(6)"UCCCCCCCCCCCCC
      CCCCCCCCCCCCCCK"
AB 590 PRINTTAB(6)"B":PRINTTAB(6)"B {WHT}[N]{RVS}
      {SPACE}B {RIGHT} {RIGHT} - {RIGHT} {RIGHT}
      {SPACE}{RIGHT} - {RIGHT} {RIGHT} {OFF}[H]"
PS 600 PRINTTAB(6)"{YEL}B{WHT} [N]{RVS} B {OFF}D
      {RVS} {OFF}F{RVS} - {OFF}H{RVS} {OFF}J{RVS}
      {OFF}K{RVS} - {OFF}: {RVS} {OFF}; {RVS}
      {OFF}[H]"
FJ 610 PRINTTAB(6)"{YEL}J*{WHT}[M]{RVS} B B _ _ _
      {SPACE}- - {SHIFT-SPACE}- _ _ {OFF}[H]"
MD 620 PRINTTAB(8)"[N]{RVS}ZBX-C-V-B-N-M-,.-./-
      {OFF}[H]{DOWN}"

```

```
GP 630 PRINTTAB(8)"{YEL}RETURN FOR VALUES SCREEN":  
      RETURN  
AS 640 DATA 62,2145,9,2408,14,2703,17,2864,22,3215  
      ,25,3608,30,4050,33,4291  
JJ 650 DATA 38,4817,41,5407,46,5728,49,6430,54,721  
      7,12,8101,23,8583,20,9634  
HE 660 DATA 31,10814,28,11457,39,12860,36,14435,47  
      ,16203,44,17167,55,19269  
QH 670 DATA 59,2273,8,2551,16,3034,19,3406,24,3823  
      ,32,4547,35,5103,43,6069,48,6812  
XD 680 DATA 51,7647,18,9094,21,10207,29,12139,34,1  
      3625,37,15294,45,18188,50,20415  
BP 690 Z=INT(6*RND(1))+1:FL=INT(3*RND(1))+0  
HQ 700 SL=INT(255*RND(1))+1  
CS 710 W1=INT(7*RND(1))+1:ONW1GOTO720,730,740,750,  
      760,770,780  
SG 720 W1=17:GOTO790  
AG 730 W1=33:GOTO790  
BF 740 W1=65:GOTO790  
CH 750 W1=129:GOTO790  
FH 760 W1=21:GOTO790  
KE 770 W1=23:GOTO790  
KD 780 W1=85  
GK 790 W2=INT(8*RND(1))+1:ONW2GOTO800,810,820,830,  
      840,850,860,870  
AX 800 W2=1:GOTO880  
DJ 810 W2=17:GOTO880  
FJ 820 W2=33:GOTO880  
GX 830 W2=65:GOTO880  
XK 840 W2=129:GOTO880  
MJ 850 W2=21:GOTO880  
SS 860 W2=23:GOTO880  
PK 870 W2=85  
DR 880 AT=INT(10*RND(1))+1:DE=INT(15*RND(1))+1:SU=  
      INT(15*RND(1))+1  
RF 890 RE=INT(15*RND(1))+1:SO=INT(4*RND(1))+1:ONSO  
      GOTO895,900,910,920  
JC 895 PO=240:GOTO930  
HQ 900 PO=241:GOTO930  
RC 910 PO=242:GOTO930  
MQ 920 PO=243  
CA 930 XT=INT(40*RND(1))+1  
BR 940 VS=INT(4*RND(1))+1:ONVSGOTO950,960,970,980  
XR 950 VS=17:GOTO990  
BR 960 VS=33:GOTO990  
DA 970 VS=65:GOTO990  
EF 980 VS=129  
AE 990 VI=INT(200*RND(1))+55  
MH 1000 DB=INT(8*RND(1))+1:DC=INT(8*RND(1))+1:DD=I  
      NT(8*RND(1))+1
```

```

AX 1010 VO=INT(3*RND(1))+1:ONVOGOTO1020,1030,1040
KF 1020 VO=31:GOTO250
AJ 1030 VO=45:GOTO250
PR 1040 VO=79:GOTO250
BF 1050 POKE53280,6:POKE53281,6:POKE198,0:PRINT
    {CLR}{WHT}":PRINTTAB(13)"VALUES SCREEN
    {2 DOWN}"
MC 1060 PRINT"Z ="TAB(21)Z:PRINT"FL ="TAB(21)FL
ED 1070 PRINT"VOICE 1 ="TAB(21)W1:PRINT"VOICE 2 ="
    TAB(21)W2
JX 1080 PRINT"ATTACK ="TAB(21)AT:PRINT"DECAY ="TAB
    (21)DE
HB 1090 PRINT"SUSTAIN ="TAB(21)SU:PRINT"RELEASE ="
    TAB(21)RE
DG 1100 PRINT"RESONANCE ="TAB(21)PO
HA 1110 PRINT"SYNC SPEED ="TAB(21)XT
QB 1120 PRINT"VIBRATO SPEED ="TAB(21)VI
EB 1130 PRINT"VIBRATO SHAPE ="TAB(21)VS:PRINT"PULS
    E SHAPE VOICE 1 ="DB
JE 1140 PRINT"PULSE SHAPE VOICE 2 ="DC:PRINT"PULSE
    SHAPE VOICE 3 ="DD
ME 1150 PRINT"FILTER ="TAB(21)VO
HS 1160 PRINT"STEP LIMIT ="TAB(21)SL
BC 1170 PRINT"{DOWN}{RVS}PRESS RETURN FOR KEYBOARD
    SCREEN
QD 1180 GETA$:IFA$<>CHR$(13)THEN1180
CA 1190 GOSUB490:GOTO280
HP 1200 S$="":POKE198,0:PRINT"{CLR}":POKE214,9:PRI
    NT:POKE211,4
SJ 1210 INPUT"SOUND TO SAVE";S$:IFS$=""THEN GOSUB49
    0:GOTO280
RD 1220 OPEN1,8,1,S$:PRINT#1,Z
DK 1230 PRINT#1,FL:PRINT#1,W1:PRINT#1,W2:PRINT#1,A
    T:PRINT#1,DE:PRINT#1,SU
HF 1240 PRINT#1,RE:PRINT#1,PO:PRINT#1,XT:PRINT#1,V
    I:PRINT#1,VS:PRINT#1,DB
EB 1250 PRINT#1,DC:PRINT#1,DD:PRINT#1,VO:PRINT#1,S
    L:CLOSE1:GOSUB490:GOTO280
AG 1260 S$="":POKE198,0:PRINT"{CLR}":POKE214,9:PRI
    NT:POKE211,4
SJ 1270 INPUT"SOUND TO LOAD";S$:IFS$=""THEN GOSUB49
    0:GOTO280
HM 1280 OPEN1,8,0,S$:INPUT#1,Z
GE 1290 INPUT#1,FL:INPUT#1,W1:INPUT#1,W2:INPUT#1,A
    T:INPUT#1,DE:INPUT#1,SU
XB 1300 INPUT#1,RE:INPUT#1,PO:INPUT#1,XT:INPUT#1,V
    I:INPUT#1,VS:INPUT#1,DB
PX 1310 INPUT#1,DC:INPUT#1,DD:INPUT#1,VO:INPUT#1,S
    L:CLOSE1:GOSUB490:GOTO250

```

The Coordinator

Lee Noel, Jr.

This unusual program unlocks the creative potential of full hi-res. Convenient color control, straightforward syntax, and numerous other carefully designed features make it a particularly effective tool for the BASIC programmer.

Colorful, detailed design work and the charting of math equations are two obvious uses for the full hi-res display. Unfortunately, BASIC is poorly suited to these tasks. As a result, a large number of machine language aids have been developed to help the graphics programmer.

While they're all good, they tend to force the programmer into a restricted environment. "The Coordinator" is designed to preserve the entire range and flexibility of the BASIC programmer, and at the same time provide the ML speed and efficiency so useful in hi-res operations.

True hi-res is often neglected on the 64—its color system is hard to understand and difficult to master, but the results are well worth the effort. With The Coordinator, the effort is less and the understanding easier.

Getting Started

Start by typing in Program 1, a BASIC loader that POKes The Coordinator into memory. It uses a special code to reduce your typing time. When you've typed in and run the loader without error messages, save a copy and then type **SYS 707** as shown on the screen. You should see a **CAN'T CONTINUE** error message. Try again. This time, there should be just the familiar **READY** prompt. Now, for one final check, type **POKE 1020,0:SYS 707**. The original error message should reappear. If you pass all these checks, there's almost certainly a working version of The Coordinator in memory.

Now you're ready to create a fast-loading version of the program. Type **NEW** to clear out the loader. (Don't worry; neither **NEW** nor **RUN/STOP-RESTORE** ever has any harmful effect on The Coordinator.) Enter Program 2, "Object

Maker," and save a copy. It's set up for disk users—if you're using tape, change the 8 to a 1 in line 30.

Now type **RUN**. The program creates two short files: A COORD.OBJ and B COORD.OBJ. When the READY prompt appears again, your files are completed. Disk files are one and three blocks long, respectively. (Disk users should note that this short program does not read the error channel. Make sure you're properly prepared before running it.) Tape files have a combined save time of about a minute.

The two files make up the whole of The Coordinator. Once they're created, you no longer need the BASIC loader (Program 1). From then on, whenever you want to use The Coordinator, type:

LOAD"A COORD.OBJ",8,1 (for disk)

or

LOAD"A COORD.OBJ",1,1 (for tape)

followed by

NEW

then

LOAD"B COORD.OBJ",8,1 (for disk)

or

LOAD"B COORD.OBJ",1,1 (for tape)

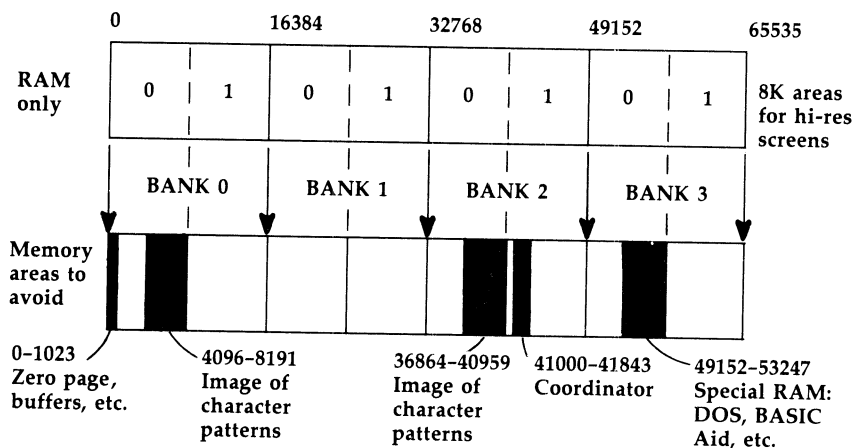
followed by

NEW

(Tape users can generally ignore LOAD ERROR messages when loading B COORD.OBJ.) You're now ready to begin your BASIC program.

Since The Coordinator is tucked into out-of-the-way places, you can forget about it now. All of BASIC memory is free, as are the cassette buffer and the overworked 4K of RAM at 49152–53247. (See Figure 1a.)

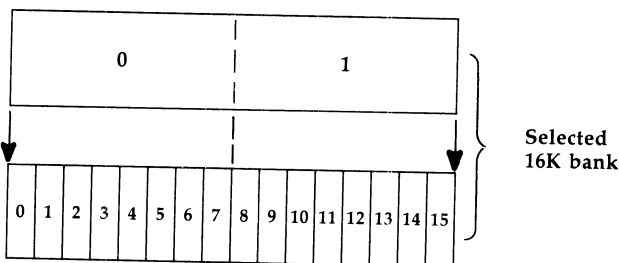
To make things clearer, type in Program 3, "Demo." It's of general interest since it can be used as a framework for all your hi-res work. Line 10 contains the device variable D. Set it to 1 if you're using tape. Although the program is filled with

Figure 1a. Commodore 64 Memory Map

The Coordinator is located at 681-767, 1020-1023, and 41000-41843.

Figure 1b.

The two possible 8K hi-res screen locations

**Figure 1c.**

The 16 possible 1K color memory locations

The position of any hi-res combination can be described by three locations—Bank, Screen, and Color memory. The Demo uses 1, 1, 7 (See line 55).

REMARKS, they're there for your convenience. They may be omitted if you wish. When you've finished typing Program 3 in, save a copy, but don't run it yet.

One Giant Programmable Character

Full hi-res turns your 64's display into one giant programmable character, 320 pixels across and 200 pixels high. The Coordinator treats this like a piece of graph paper, although there are no visible features. There's an *x*-axis on the bottom, running horizontally from 0 to 319, and a *y*-axis on the left, running vertically from 0 to 199. The origin—the 0,0 start point—is at the lower left-hand corner. Thus, the graph is set up on a

standard Cartesian system, the same kind you may have used to study coordinate geometry.

Any point can be described in terms of one x value combined with one y value. For example, $x = 160$ and $y = 100$ is a point at the center of the display. All your BASIC program has to do is generate some x and y values. The Coordinator takes care of everything else.

Demo will show you how this works in practice. Lines 10 and 15 are for convenience. If you haven't already loaded your Coordinator files, they will. Make sure you have the disk containing A COORD.OBJ and B COORD.OBJ in the drive, or the tape rewound to the right place, before running Demo. (This step is skipped if the program is run more than once.)

Since it does most of its work automatically, The Coordinator has only four simple commands.

1. **SYS 686** clears hi-res screen and color memory, and then plots the current values your BASIC program has specified for variables x and y .
2. **SYS 707** plots x and y .
3. **SYS 714** erases x and y .
4. **SYS 681** resets The Coordinator if you change your hi-res setup. This means you can have several hi-res displays in memory at the same time.

There are two additional features. After using one of the above commands, you can PEEK(767) to see if the point was set (turned on) before the SYS. Any nonzero result from PEEK(767) indicates that the point *was* on. You can use this to avoid erasing points, or you may wish to detect collisions between certain points. The final feature enables you to determine if The Coordinator has initialized itself. If PEEK(1020) returns 64, it probably has. This can be useful in certain situations that crop up when you use the program in complex ways.

Lines 25 and 30 simply assign meaningful variable names to these addresses. You can, of course, use any names you like.

How It Works

How can The Coordinator do anything significant with these minimal commands? Run the Demo program and watch.

Briefly, here's how The Coordinator works. Any time you SYS to it, it looks for and stores the current values of variables x and y . If x and y are not defined, they're treated as if they're

zero. Remember, SYS CLEAR plots after it clears. You can avoid plotting a point by setting x and y to values that don't appear on the screen. $X = -1 : Y = -1$ works fine.

Offscreen quantities cause no trouble. If you tried to plot them with most utilities, there is a chance that you'd corrupt your BASIC program or even lock up your computer. This can't happen with The Coordinator because it deals only with reasonable values. All others are rejected. Good values are automatically converted into integers for display purposes. As a result, you have complete freedom to plot experimentally. After all, you may be graphing x and y to see what their values are: It would be a shame if out-of-range quantities couldn't be accommodated safely. You're protected with The Coordinator.

Every time you use the program, it checks the color information stored at certain addresses of your 64. These addresses are assigned to variables in lines 40 and 45 of Demo. You'll notice that they represent the normal foreground (text) and background color registers on the 64. Why learn special new commands for color? Just use the familiar registers; The Coordinator gets all its color information from them. Figure 2 lists the important color registers.

Figure 2. Coordinator Color Sources

	Color	Source	Notes
Full Hi-res	Background	53281	Can be set for each 8×8 cell. Any bit on (set to 1) shows in this color. Only one color per cell.
	Plotting	646	
Multicolor	Background	53281	Affects entire screen. Set for each 8×8 cell. Given by bit pattern 0, 1.
	First Plotting	646	
	Second Plotting	53282	
	Third Plotting	53283	Set for each 8×8 cell. Given by bit pattern 1, 1.

In true hi-res, background and foreground colors are set individually for each 8×8 character-sized block of the display. This gives 1000 possible differently colored background areas, each of which can contain one contrasting plotting color. SYS CLEAR applies the colors in the registers to color memory, as well as clearing all lit pixels from the screen. As a consequence, a uniform background color is displayed. The

other commands apply color information to whatever 8×8 cell a point is in. This explains the small rectangles and vari-colored lines you see as the demo runs. With a little thought, this type of screen can be beautifully colored, and can display a subtlety and fineness of detail not possible with the more popular four-color multicolor hi-res.

The truly observant will notice MFLAG in line 30 of Demo. The Coordinator also works in multicolor hi-res. MFLAG is a flag used by Demo to turn multicolor mode on or off.

Look at line 55 and the subroutine at 185. These set up the locations of the hi-res screen and color memory. You do this from BASIC, so that you're in control. Full memory flexibility is still available. The Coordinator's CLEAR and RESET functions, or an initial PLOT or ERASE, examine the necessary registers and adjust themselves accordingly. Figure 1 explains the possible arrangements graphically.

By this stage, you'll probably have a good idea of what's going on as Demo runs. When it's over, one of the three sine waves disappears. Press any key to return to a normal display. (This is the function of the remaining lines of the program, 155–175.) You can also use the RUN/STOP-RESTORE combination at any time. Now rerun the program with MFLAG set to one. You'll see that the differences are sweeping, but Figure 2 should help you out.

Multicolor works by having only 160 double-wide (two-pixel) x values. These fat coordinates give extra color information to the computer. Since The Coordinator is mainly for use in true hi-res, you must specify both pixels yourself.

In essence, it's easy. The first pixel of a pair must have 0 or an even number as x coordinate. (Coordinator x coordinates are always 0–319.) Keeping that in mind, you can see how to obtain the different colors:

(x starts as an even coordinate; y has no effect on color.)

1. Bit pattern 01 gives color in address 646. SYS
ERASE:X=X+1 :SYS PLOT
2. Bit pattern 10 gives color in address 53282. SYS
PLOT:X=X+1 :SYS ERASE
3. Bit pattern 11 gives color in address 53283. SYS
PLOT:X=X+1 :SYS PLOT

You may want to assign priorities by skipping the erase function or checking for previously set points. The need for

What Good Is It If You Can't Save It?

A lot of thought and dogged programming may be required to produce successful hi-res screens. The effort is wasted if you can't save and reload them for future display or fine-tuning. Luckily, there is a way to save and load hi-res screens from within your BASIC programs.

Begin by loading Demo and deleting lines 65–145. Then type in "Mountain Demo," Program 4, and "Save/Load Subroutine," Program 5, to form one combined program. Again, be sure to set D to 1 in line 220 if you're using tape. Also, tape users—and disk users who dislike Save-with-Replace—should change line 295 to:

295 SYSPCHR\$(E)+SN\$,D,1:POKE193,0:POKE194,B(OK)

(Disk users who leave the line intact will have the option of updating their work while keeping filenames constant.) Now save the combined program.

When run, the routine plots one of several kinds of mountain formations, and then offers you the chance to save the current formation or reload an earlier design. If you get something you like, save it by following the prompts. Filenames may be up to fifteen characters long. Otherwise, press RUN/STOP-RESTORE and rerun the program until you're satisfied.

You'll note that the mountains are drawn quite slowly. The Coordinator is very fast, but BASIC is lazily providing it with x/y values for shading, rock formation, jaggedness, and several other factors.

When you save a mountain, make sure that you've got plenty of room on your disk or tape. A minimum of about 10K is stored by this system. The information is contained in three files (four for multicolor), denoted by your filename with a prefix of 1, 2, 3, or 4.

You can juggle the combined program around to suit almost any hi-res need. However, screens that lie under the ROM areas cannot be saved by this method.

Multicolor experimenters should note two things. Mountain Demo was written for true hi-res and may surprise you. More importantly, the multicolor mode 11 bit pair gets its color information from color memory, which starts at 55296. This area is affected any time the screen is cleared, as the Save/Load Subroutine (Program 5) does by printing E\$ in lines 230, 290, and 345. Hence, one-

third of the color information available in this mode may be scrambled. Adapting the Save/Load Subroutine to sidestep this problem can teach you a lot about hi-res graphics.

How It Works

Every BASIC programmer is familiar with saving and loading programs. This feature of the computer is built in, and is usually limited to the storage and retrieval of BASIC programs. But the Kernal routines that perform save and load can be used anywhere in memory. All it takes is an adjustment of pointers that normally keep track of where your BASIC program is.

The method used here, in Object Maker, and at the beginning of Demo, is an adaptation of that used by Raeto West in Chapter 6 of his reference guide, *Programming the Commodore 64* from COMPUTE! Books. The programs below show how the system works. Note that any address on the 64 can be represented in the following way:

$$\text{Address} = \text{Low Byte} + (256 * \text{High Byte})$$

A byte can have any whole number value in the range 0–255.

To save:

```
10 SYS 57812"FILENAME",D,1 :REM SET FILENAME
20 POKE 193,low byte of start address:POKE 194,high byte
of
start
30 POKE 174,low byte + 1 of end address:POKE 175,high byte
of end
40 SYS 62957 :REM SAVE
```

To load:

```
10 POKE 147,0 :REM SET LOAD FLAG
20 SYS 57812"FILENAME",D,1 :REM SET FILENAME
30 SYS 62631 :REM LOAD
```

Lines 295–330 of the Save/Load Subroutine show the method in action.

As you can see, there's no interference with normal program flow, and the routines are fast and flexible. Try them, and they'll soon become indispensable.

this will become obvious as you use the program. Where x values are out of your control, you'll want to use a function like that defined in line 125. This will insure that the x coordinates are even.

A few final notes may be useful. The Coordinator can plot under ROM, so no BASIC memory need be sacrificed to the display. If you set the variables in line 55 to 3, 1, and 5, respectively, The Coordinator will plot your screen under the Kernal and color under the SID registers. You can't check for previously set points, however, because the screen cannot be read while the ROM is active.

While displays located under ROM are always safe from interference from BASIC, those in BASIC RAM may need to be protected, depending on the length of your program and the amount of memory it uses for variables. Another point—if you reset your computer using a cold start, SYS 64738, you'll wipe out the low portion of the program, and you'll need to reload A COORD.OBJ.

Finally, no matter how helpful a utility is, bugs will still creep into your hi-res programming. You may need to check the normal screen for error messages, so you'll want to avoid the RUN/STOP-RESTORE combination, which clears the screen. If you use the Demo framework, just press RUN/STOP, and type GOTO 170. You'll instantly flip back to the normal display and be able to read the error message. The Coordinator should help make this procedure a rare one.

Program 1. The Coordinator Loader

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

```

SS 1 DATA L,1,H,A7,2,L,36,85,1,4C,25
PE 2 DATA 93,A0,T,T,L,37,85,1,4C,8B,63
GD 3 DATA E3,L,37,85,1,60,L,0,H,2E,29
CC 4 DATA A0,Z,5,L,1,H,2E,A0,L,36,1E
XR 5 DATA 85,1,4C,G,A0,A2,37,86,1,85,1D
MX 6 DATA 45,20,E7,B0,20,A2,BB,A6,61,E0,B9
MM 7 DATA 8A,B0,D,20,9B,BC,A5,64,A4,65,2E
SP 8 DATA A6,66,E0,80,K,3,68,68,60,A2,9C
DH 9 DATA 36,86,1,60,P,FC,3,7C9,40,Z,BE
BQ 10 DATA 72,L,20,H,FF,3,A,H,FC,3,W
AD 11 DATA A0,0,G,FE,3,G,34,A0,L,1,3B
GM 12 DATA H,FD,3,99,44,A0,99,4C,A0,C8,4C
AR 13 DATA A,D0,F6,P,0,DD,29,3,H,2A,29
RH 14 DATA A0,L,3,38,W,2A,A0,A,H,2A,5B

```

HE 15 DATA A0,A,A,A,A,A,H,30,A0,P,26
 EC 16 DATA 18,D0,48,29,Z,4A,4A,H,35,A0,4E
 BM 17 DATA 68,29,8,Z,3,X,2A,A0,P,2A,19
 PM 18 DATA A0,C9,6,Z,9,C9,5,Z,5,P,23
 CB 19 DATA 2A,A0,D0,5,A2,1A,4C,B7,2,A,A8
 EQ 20 DATA A,A,A,A,A,H,2D,A0,L,0,H,2D
 GJ 21 DATA 2C,A0,20,B8,A2,L,0,85,46,L,9B
 HF 22 DATA 59,20,D6,2,C9,0,Z,3,4C,BE,34
 RG 23 DATA 2,C0,C8,K,3,4C,BE,2,H,57,45
 BB 24 DATA A0,G,56,A0,L,58,20,D6,2,C9,13
 RH 25 DATA 2,K,3,4C,BE,2,C9,1,Z,2,BB
 XF 26 DATA D0,7,C0,40,K,3,4C,BE,2,H,1A
 KM 27 DATA 55,A0,G,54,A0,P,56,A0,H,40,3D
 JG 28 DATA A0,A0,3,4E,40,A0,88,D0,FA,38,A
 AE 29 DATA L,18,W,40,A0,H,40,A0,L,7,19
 JX 30 DATA 2D,56,A0,H,41,A0,38,L,7,W,P
 PR 31 DATA 41,A0,H,41,A0,P,55,A0,0,54,37
 GM 32 DATA A0,H,43,A0,G,42,A0,A0,3,18,A
 JX 33 DATA 4E,43,A0,6E,42,A0,88,D0,F7,L,75
 FM 34 DATA 7,2D,54,A0,H,43,A0,38,L,7,3F
 PK 35 DATA W,43,A0,H,43,A0,L,0,H,29,21
 JR 36 DATA A0,P,40,A0,A0,6,A,2E,29,A0,9D
 ES 37 DATA 88,D0,F9,H,28,A0,18,P,40,A0,9B
 JC 38 DATA 6D,29,A0,H,29,A0,L,0,H,FF,98
 KD 39 DATA 2,P,42,A0,A,A,A,K,3,X,Q
 EC 40 DATA 29,A0,18,6D,28,A0,H,28,A0,K,10
 BK 41 DATA 3,X,29,A0,18,P,41,A0,6D,28,3
 CR 42 DATA A0,H,28,A0,H,36,A0,K,3,X,D6
 QD 43 DATA 29,A0,P,29,A0,H,37,A0,18,P,52
 EQ 44 DATA 2C,A0,6D,28,A0,H,28,A0,P,2D,5
 EF 45 DATA A0,6D,29,A0,H,29,A0,P,28,A0,9F
 GA 46 DATA H,44,A2,H,79,A2,P,29,A0,H,41
 SA 47 DATA 45,A2,H,7A,A2,A0,3,4E,37,A0,8A
 KA 48 DATA 6E,36,A0,88,D0,F7,P,36,A0,H,2C
 SG 49 DATA 4C,A2,H,3E,A2,18,P,35,A0,6D,E
 XC 50 DATA 37,A0,H,4D,A2,18,L,D8,6D,37,23
 PM 51 DATA A0,H,3F,A2,20,83,A2,P,23,D0,CF
 PQ 52 DATA 29,F,Q,32,A0,Z,3,H,0,4,31
 XF 53 DATA 20,60,A3,P,0,4,85,2,P,5C,18
 AK 54 DATA A0,H,0,4,20,6A,A3,0,43,A0,84
 EF 55 DATA B9,44,A0,48,25,2,Z,3,H,FF,A7
 GC 56 DATA 2,68,Q,2E,A0,D0,A,Q,FF,2,D8
 BA 57 DATA D0,11,5,2,18,K,9,Q,FF,2,5E
 DP 58 DATA Z,7,49,FF,25,2,H,0,4,L,J
 FP 59 DATA 0,H,2E,A0,4C,BE,2,P,16,D0,BC
 JH 60 DATA 29,10,H,32,A0,P,86,2,A0,4,6C
 JB 61 DATA A,88,D0,FC,H,5C,A0,P,32,A0,82
 FR 62 DATA Z,F,P,22,D0,29,F,18,6D,5C,5C
 JE 63 DATA A0,H,5C,A0,18,K,C,P,21,D0,69
 SB 64 DATA 29,F,18,6D,5C,A0,H,5C,A0,60,11

```

HC 65 DATA 20,83,A2,P,30,A0,18,6D,35,A0,82
BF 66 DATA H,35,A0,L,0,H,3C,A0,P,A7,6D
FX 67 DATA 2,Z,8,L,0,H,A7,2,4C,5F,3A
PP 68 DATA A3,P,2D,A0,H,38,A0,L,1F,H,84
JH 69 DATA 3A,A0,L,40,H,3B,A0,P,38,A0,94
AM 70 DATA H,FB,A2,L,0,H,FA,A2,A8,Q,59
RX 71 DATA 3A,A0,P,3C,A0,99,0,4,C8,D0,16
AK 72 DATA FA,X,FB,A2,CA,D0,F4,O,FB,A2,33
HF 73 DATA G,15,A3,O,FA,A2,G,14,A3,A0,55
GS 74 DATA 0,99,0,4,C8,CC,3B,A0,D0,F7,B2
GM 75 DATA P,FB,A2,C9,DB,Z,3C,P,38,A0,75
SC 76 DATA CD,35,A0,Z,1C,P,5C,A0,H,3C,19
KB 77 DATA A0,P,35,A0,H,38,A0,L,3,H,6C
SB 78 DATA 3A,A0,L,R,H,3B,A0,20,60,A3,87
BK 79 DATA 18,K,A0,20,6A,A3,P,32,A0,Z,5A
PH 80 DATA 10,P,23,D0,29,F,H,3C,A0,L,15
HD 81 DATA D8,H,38,A0,4C,E7,A2,60,L,7F,1A
HE 82 DATA H,D,DC,L,34,85,1,60,L,36,6D
FP 83 DATA 85,1,L,81,H,D,DC,60,0,0,6
PC 84 A=681:H=707:U=10:S=1:R=83:R=R-1
SJ 85 D=0:E$=CHR$(147):M$(1)="LINE # ":M$(2)="DATA
      ":M$(3)=E$+"RECHECK ALL LINES "
BK 86 PRINTES"{WHT} PLEASE WAIT 2 MIN. 40 SEC."
BM 87 FORI=STOS+R:E=I:FORJ=1TOU:READN$
XC 88 IFJ=1THENM=PEEK(63)+PEEK(64)*256:IFM<>ITHENF
      =1:GOTO98
EE 89 GOSUB99
HG 90 ON-(F=2)GOTO98
DJ 91 POKEA,N:A=A+1-(A=764)*40335
EM 92 E=E-N-J+U-((N+J-U>E)*(2*(N+J-U-E))):D=D+N
FQ 93 NEXT:READN$:GOSUB99
MX 94 IFN<>ETHENF=2:GOTO98
GP 95 NEXT
HM 96 IFD<>94716THENF=3:GOTO98
MA 97 PRINT"OK! SYS"H:END
SJ 98 PRINTES"LINE"M"HAS "M$(F)"ERROR":STOP
GJ 99 L=LEN(N$):C=ASC(N$):ON-(L<2ANDC>=71ANDC<=90)
      GOTO104
HG 100 N(1)=0:N(2)=N(1)
AR 101 IFL>2THENF=2:GOTO106
RX 102 FORK=1TOL:H$(K)=MID$(N$,K,1):N(K)=ASC(H$(K))
      :N(K)=N(K)-55-(N(K)<65)*7
BP 103 NEXT:N=N(2)-16*(L=2)*N(1)-(L=1)*N(1):GOTO105
MF 104 N=C:N=N+69-(N>=76)*24-(N>=82)*57
RA 105 IFN<0ORN>255THENF=2
MR 106 RETURN

```


Program 2. Object Maker

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

```

KK 10 SA=53239:FORI=0TO8:READML:POKESA+I,ML:NEXT
KQ 20 DATA 32,248,2,32,237,245,76,190,2
EM 30 D=8:E=65
EJ 40 SYS57812CHR$(E)+" COORD.OBJ",D,1
GQ 50 POKE193,169:POKE194,2:POKE174,253:POKE175,2
GH 60 SYS62957:E=E+1
DM 70 SYS57812CHR$(E)+" COORD.OBJ",D,1
GF 80 POKE193,140:POKE194,160:POKE174,116:POKE175,
    163
AM 90 SYSSA

```

Program 3. Demo

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

```

GA 5 REM *** LOAD ML FROM DISK OR TAPE ***
AP 10 D=8:E=65:A=E:F=147:E$=CHR$(F):P=57812:L=6263
    1:S=62957:IFPEEK(687)=54THEN25
HD 15 POKEF,0:SYSPCHR$(E)+" COORD.OBJ",D,1:SYSL:E=
    E+1:IFE<A+2THEN15
SG 20 REM *** SET COORDINATOR VARIABLES ***
FG 25 RESET=681:CLEAR=686:PLOT=707:ERASE=714
HX 30 SETFLAG=767:INIT=1020:MFLAG=0:REM MFLAG=1 TU
    RNS ON MULTICOLOR, 0 IS OFF
GX 35 REM *** SET COLOR REGISTERS ***
PB 40 FG=646:BO=53280:BG=BO+1
GE 45 IFMFTHENM2=BO+2:M3=BO+3:POKEM2,6:POKEM3,4:RE
    M SET MULTICOLOR REGS. (M1=FG)
EH 50 REM *** SET HI-RES VARIABLES ***
GS 55 BANK=1:SCNOFFSET=1:COLOFFSET=7:GOSUB185
JB 60 REM *** CREATE DISPLAY (FROM LINE 65 TO LINE
    145) ***
HB 65 POKEBO,0:POKEBG,0:POKEFG,1:Y=100:X=160:SYSCL
    EAR:O=X:N=Y:REM SET SCREEN
MA 70 FORR=7TO87STEP8:POKEBG,-(R/8>7)*R/7:POKEFG,R
    /7-8*(R/8>7):REM RADIUS & COLOR
EG 75 FORA=0TO1/2STEP2/R:X=R*SIN(A)+O:Y=R*COS(A)+N
    :SYSPLOT:REM SWEEP 90 DEGREES
XD 80 X=-X+2*O:SYSPLOT:Y=-Y+2*N:SYSPLOT:X=-X+2*O:S
    YSPLOT:REM BUT PLOT 4 QUADRANTS
EG 85 NEXTA
HR 90 NEXTR
FH 95 IFMFTHENPOKEM3,5:REM BIT PATTERN 1,1 PLOTS G
    REEN IF MULTICOLOR
EC 100 Y=100:FORX=0TO319:POKEBG,X/8:POKEFG,X/8+1:S
    YSPLOT:NEXT:REMDRAW COLORED LINE

```

```

RG 105 POKEBG,0:POKEFG,10:FORX=0TO319:SYSERASE:NEX
T:REM ERASE LINE WITH LT. RED
HR 110 Y=95:POKEFG,1:FORX=0TO319:SYSPLT:IFPEEK(SE
TFLAG)THENSYSPLT:GOTO120
BB 115 SYSERASE:REM 95-105 MOVE A WHITE POINT BUT
{SPACE}DON'T ERASE
CJ 120 NEXT
CA 125 DEFFNMC(A)=INT(A)-(INT(A/2)<>INT(A)/2):SYSC
L:R=95:POKE646,2
JG 130 FORA=0TO319STEP2.26:X=FNMC(A):Y=R+80*SIN(A/
20):SYSER:X=X+1:SYSPL:NEXT
RS 135 FORA=0TO319STEP2.26:X=FNMC(A):Y=R+60*SIN(A/
25):SYSPL:X=X+1:SYSER:NEXT
SD 140 FORA=0TO319STEP2.26:X=FNMC(A):Y=R+40*SIN(A/
30):SYSPL:X=X+1:SYSPL:NEXT
JS 145 FORA=0TO319STEP2.26:X=FNMC(A):Y=R+60*SIN(A/
25):SYSER:X=X+1:SYSER:NEXT
JF 150 REM *** RESTORE DEFAULT DISPLAY ***
SG 155 IFPEEK(198)=0THEN155:REM WAIT FOR A KEYPRES
S TO END PROGRAM
JF 160 POKE198,0:REM TIDY UP KEYPRESS IF IT COMES
AF 165 POKEFG,PEEK(BG)+1:REM ENSURE TEXT COLOR IS
{SPACE}DIFFERENT FROM BACKGROUND
CR 170 POKE53270,200:POKE56576,151:POKE53265,27:PO
KE53272,21:IFSFTHENRETURN
KB 175 END
HF 180 REM *** SET UP HI-RES ***
MR 185 POKE56576,(PEEK(56576)AND252)OR(3-BANK):REM
SET BANK
AC 190 POKE53265,PEEK(53265)OR32:REM TURN ON BITMA
P
ED 195 IFMFTHENPOKE53270,PEEK(53270)OR16:REM SET M
ULTICOLOR IF DESIRED
PX 200 POKE53272,(COLOFFSET*16+SCNOFFSET*8):REM PO
SITION HI-RES AND COLOR MEMORIES
GD 205 RETURN

```

Program 4. Mountain Demo

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

```

RG 65 POKE198,0:POKEFG,11:POKEBG,0:POKEBO,0:X=-1:Y
=-1:SYSC
GE 66 J%=RND(0)*10:J%=-J%*(J%>3ANDJ%<7):ON-(J%=0)G
OTO66:J%=J%-4:C=2↑J%
HD 67 J=35-(C=1)*17:V=2:A=-20:J%=RND(0)*5:B=-J%*20
:Z=1:E=0
BA 68 H=INT((320-B)/(J-10)):DIM G(H+1):G(0)=B:DEFF
NP(M)=(-1)↑INT(RND(0)*3)

```

```

HS 69 DEFFNM(R)=(N+(N<80)*N*.3)/133+(N>180)*(N-180
)/79
EM 70 FORQ=1TOH:G(Q)=G(Q-1)+J+RND(0)*10:IFG(Q)>=32
0THENO=Q:Q=H
CF 71 NEXT:DIMR(O+1,2),T(O+1,2),U(O+1),B(O+1):J%=R
ND(0)*5:M=10+(J%+4-C)*5
HM 72 J%=- (C>1):R(0,0)=G(0):T(0,0)=RND(0)*3.3-A:U(
0)=1
SF 73 FORQ=1TOO:R(Q,0)=G(Q):U(Q)=U(Q-1)*(1+2*(Q/C=
INT(Q/C)))
BC 74 T(Q,0)=T(Q-1,0)+((RND(0)*3.3+2)*U(Q)):N=R(Q,
0)
JM 75 T(Q,0)=T(Q,0)-(N<=160)*N/80+(N>160)*N/120:GO
SUB83:B(Q)=- (K>I):NEXT:E=1
QH 76 V=V+.004:M=M+V↑1.0001:R(0,1)=R(0,0)+.9+RND(0
)*.5*FNP(M)
EK 77 T(0,1)=T(0,0)+RND(0)*2:FORQ=1TOO
EM 78 R(Q,1)=R(Q,0)+(1+(B(Q)=0ANDB(Q+1)=1)*J%)*(RND
(1)*(2+C/2)+.3)
KE 79 N=R(Q,1):T(Q,1)=T(Q-1,1)+T(Q,0)-T(Q-1,0)+RND
(0)*2
KF 80 T(Q,1)=T(Q,1)-(B(Q)=1)*RND(1)*M*FNM(R)/20:IF
R(Q,1)>R(Q-1,1)THEN82
AR 81 R(Q,1)=R(Q-1,1)+.01:IFQ>=3THEN T(Q,1)=T(Q-1+(
C=1)*2,1)-6
JR 82 GOSUB83:R(Q,0)=R(Q,1):T(Q,0)=T(Q,1):NEXT:R(0
,0)=R(0,1):T(0,0)=T(0,1):GOTO76
HB 83 H=R(Q-1,E):I=T(Q-1,E):J=R(Q,E):K=T(Q,E)
FK 84 W=(J-H)*(1.3+RND(0)*.9-(K<I)*1.7*(RND(0)+1))
/SQR((J-H)↑2+(K-I)↑2)
SS 85 IFK>170ANDJ>0ANDJ<320THENJ=R(Q,0):K=T(Q,0):Z
=0
CD 86 FORX=HTOJSTEPW:Y=I+(K-I)*(X-H)/(J-H):SYSPL:N
EXT:IFZTHENRETURN
JX 87 FORY=0TO199STEP8:FORX=0TO319STEP8:U=FNP(M)*R
ND(0)*24
CH 88 POKEFG,7+2*(Y>36+U)-9*(Y>76+U)+13*(Y>114+U):
SYSER:IFPEEK(SE)THENSYSPL
JE 89 NEXT:NEXT:POKE49,PEEK(47):POKE50,PEEK(48):GO
SUB220

```

Program 5. Save/Load Subroutine

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

```

HD 210 REM *** SCREEN SAVE/LOAD SUBROUTINE ***
JC 215 REM NOTE: IS DEPENDENT ON VARIABLES FROM MA
IN PROGRAM
XA 220 D=8:E=49:OK=1:B(1)=BA*64+SC*32:T(1)=B(1)+32
:B(2)=BA*64+CO*4:T(2)=B(2)+4

```

```
XH 225 B(3)=208:T(3)=B(3)+1:B(4)=216:T(4)=B(4)+4:S
      F=1:GOSUB165:SF=0
ME 230 PRINT$:PRINT
KC 235 PRINTTAB(7)"** SAVE OR LOAD SCREEN **":PRIN
      T:INPUT"{2 SPACES}SCREEN NAME";S?N$
CP 240 GV=LEN(SN$):ON-(GV<1ORGV>15)GOTO230:PRINT
EE 245 PRINT"> FOR SAVE - PRESS {RVS}S{OFF}":PRINT
      :PRINT"> FOR LOAD - PRESS {RVS}L{OFF}"
BX 250 GETL$:IFL$=""ORL$<>"S"ANDL$<>"L"THEN250
SG 255 PRINT:PRINT:PRINT" {RVS}RESULTS{OFF} - NAME
      IS "SN$:PRINTTAB(11)"AND THIS IS A ";
GJ 260 IFL$="S"THENPRINT"SAVE":GOTO270
JK 265 PRINT"LOAD"
FM 270 PRINT:PRINT:PRINTTAB(6)">> IF CORRECT - PRE
      SS {RVS}C{OFF} <<"
FH 275 PRINT:PRINTTAB(5)"ANY OTHER KEY ALLOWS CHAN
      GES"
DF 280 GETM$:ON-(M$="")-2*(M$="C")GOTO280,290
XG 285 GOTO230
GK 290 PRINT$:PRINT" SAVING:"SN$:IFL$="L"THENPRIN
      TE$:PRINT" LOADING:"SN$:GOTO320
FE 292 PRINT" SCREEN RETURNS TO DISPLAY FOR SAVE"
      :FOR TD=0 TO 2000:NEXT:GOSUB815
KP 295 SYSP"@: "+CHR$(E)+SN$,D,1:POKE193,0:POKE194,
      B(OK)
FC 300 POKE174,0:POKE175,T(OK):SYSS
JA 305 E=E+1:OK=OK+1:IFOK<4THEN295
BS 310 IFMFANDOK=4THEN295
MG 315 SF=1:GOSUB156:SF=0:PRINT$"{DOWN}{RIGHT}
      {RVS} SAVE COMPLETED{OFF}":FORTD=0TO2000:NE
      XT:GOTO340
RS 320 POKEF,0:SYSPCHR$(E)+SN$,D,1:SYSL
MQ 325 E=E+1:IFE<52THEN320
DE 330 IFMFANDE=52THEN320
GR 335 IFD<>8THEN355
XQ 340 QA=0:OPEN15,8,15:INPUT#15,QA,QB$,QC,QD:CLOS
      E15:IFQA<20THEN355
GK 345 PRINT$:PRINT" DISK ERROR!":PRINT:PRINTTAB(
      13)"{RVS} DISK STATUS {OFF}"
EF 350 PRINT:PRINTTAB(7)QA;QB$;QC;QD:END
QM 355 GOSUB185:RETURN
```

Appendices

Appendix A

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 the Commodore 64 and 128.

BASIC Programs

This book includes programs for the Commodore 128 and 64. If you have a 128, note that most of these programs work only in 64 mode. To enter 64 mode, turn on the computer and type GO 64.

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 be sure to enter all punctuation marks, such as colons and commas, just as they appear 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 group of 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 RUN/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 most likely 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 delete characters. 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 "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 the order in which it is listed. Press RETURN at the end of each line. Use the INST/DEL key 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.

Appendix B

How to Type In Programs

In order to make it as easy as possible to type in the programs in this book, we've included two program-entry aids written in BASIC: "The Automatic Proofreader" and "MLX." To assist you in understanding how to enter these programs, COMPUTE! has established the following listing conventions.

Generally, BASIC program listings contain words within braces which spell out any special characters: {DOWN} means to press the cursor-down key; {5 SPACES} means to press the space bar five times.

To indicate that a key should be *shifted* (pressed while you're holding down the SHIFT key), the key is underlined in our listings. For example, S means to type the S key while holding down the SHIFT key. This will 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 the preceding 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 indicated; for example, [<9@>] means to type Commodore-@ nine times.

Rarely, 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.

Quote Mode

You know 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, {DOWN}'s, and {HOME}'s in our programs. The only way the computer can tell the difference between direct and programmed cursor control is by means of the quote mode.

Once you press the quote key (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 affected by quote mode is the INST/DEL key; you can still use INST/DEL to back up and edit the line. Once you type another quote, you're out of quote mode.

You also go into quote mode when you insert spaces into a line with INST/DEL. In any case, the easiest way to get out of quote mode is just to 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've included a checksum program. Please read "Automatic Proofreader" (Appendix C) before typing in any of the programs in this book.

Refer to the table on the following page when entering cursor- and color-control keys.

Keyboard Conventions

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 ↑ CRSR ↓		{ 3 }	COMMODORE 3	
{DOWN}	↑ CRSR ↓		{ 4 }	COMMODORE 4	
{LEFT}	SHIFT ← CRSR →		{ 5 }	COMMODORE 5	
{RIGHT}	← CRSR →		{ 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	

Appendix C

The Automatic Proofreader

Philip I. Nelson

"The Automatic Proofreader" helps you type in program listings for the Commodore 64 and 128, and prevents nearly every kind of typing mistake.

Type in the "Proofreader" *exactly* as listed. Since the program can't check itself, type carefully to avoid mistakes. Don't omit any lines, even if they contain unfamiliar commands. After finishing, save a copy or two on disk or tape before running it. This is important because the Proofreader erases the BASIC portion of itself when you run it, leaving only the machine language portion in memory.

Next, type **RUN** and press RETURN. After announcing which computer it's running on, the Proofreader displays the message *Proofreader Active*. Now you're ready to type in a BASIC program.

Two Letters

Every time you finish typing a line and press RETURN, the Proofreader displays a two-letter checksum in the upper left corner of the screen. Compare this result with the two-letter checksum printed to the left of the line in the program listing.

If the letters match, it's almost certain the line was typed correctly. If the letters don't match, check for your mistake and correct the line.

What It Does, What It Doesn't

The Proofreader ignores spaces not enclosed in quotation marks, so you can omit or add spaces between keywords and still see a matching checksum. However, since spaces *inside* quotation marks are almost always significant, the Proofreader

pays attention to them. For example, **10 PRINT"THIS IS BASIC"** generates a different checksum from that generated by **10 PRINT"THIS ISBASIC"**.

A common typing error is transposition—typing two successive characters in the wrong order, like PIRNT instead of PRINT or 64378 instead of 64738. The Proofreader is sensitive to the *position* of each character within the line and thus catches transposition errors.

The Proofreader does *not* accept keyword abbreviations (for example, ? instead of PRINT). If you prefer to use abbreviations, you can still check the line by LISTing it after typing it in, moving the cursor back to the line, and pressing RETURN. LISTing the line substitutes the full keyword for the abbreviation and allows the Proofreader to work properly. The same technique works for rechecking programs you've already typed in.

If you're using the Proofreader on the Commodore 128, *do not perform any GRAPHIC commands while the Proofreader is active*. When you perform a command like GRAPHIC 1, the computer moves everything at the start of BASIC program space—including the Proofreader—to another memory area, causing the Proofreader to crash. The same thing happens if you *run* any program with a GRAPHIC command while the Proofreader is in memory.

Though the Proofreader doesn't interfere with other BASIC operations, it's a good idea to disable it before running another program. However, the Proofreader is purposely difficult to dislodge: It's not affected by tape or disk operations, or by pressing RUN/STOP-RESTORE. The simplest way to disable it is to turn the computer off, then on. A gentler method is to SYS to the computer's built-in reset routine (**SYS 65341** for the 128; **SYS 64738** for the 64). These reset routines erase any program in memory, so be sure to save the program you're typing in *before* entering the SYS command.

If you own a Commodore 64, you may already have wondered whether the Proofreader works with other programming utilities like "MetaBASIC." The answer is, generally, yes, *if you're using a 64 and activate the Proofreader after installing the other utility*. For example, first load and activate MetaBASIC; then load and run the Proofreader.

When using the Proofreader with another utility, you should disable *both* programs before running a BASIC program. While the Proofreader seems unaffected by most utilities, there's no way to promise that it will work with any and

every combination of utilities you might want to use. The more utilities activated, the more fragile the system becomes.

The Automatic Proofreader

```
10 VEC=PEEK(772)+256*PEEK(773):LO=43:HI=44
20 PRINT "AUTOMATIC PROOFREADER FOR ";:IF VEC=4236
  4 THEN PRINT "C-64"
30 IF VEC=50556 THEN PRINT "VIC-20"
40 IF VEC=35158 THEN GRAPHIC CLR:PRINT "PLUS/4 & 1
  6"
50 IF VEC=17165 THEN LO=45:HI=46:GRAPHIC CLR:PRINT
  "128"
60 SA=(PEEK(LO)+256*PEEK(HI))+6:ADR=SA
70 FOR J=0 TO 166:READ BYT:POKE ADR,BYT:ADR=ADR+1:
  CHK=CHK+BYT:NEXT
80 IF CHK<>20570 THEN PRINT "*ERROR* CHECK TYPING
  {SPACE}IN DATA STATEMENTS":END
90 FOR J=1 TO 5:READ RF,LF,HF:RS=SA+RF:HB=INT(RS/2
  56):LB=RS-(256*HB)
100 CHK=CHK+RF+LF+HF:POKE SA+LF,LB:POKE SA+HF,HB:N
  EXT
110 IF CHK<>22054 THEN PRINT "*ERROR* RELOAD PROGR
  AM AND CHECK FINAL LINE":END
120 POKE SA+149,PEEK(772):POKE SA+150,PEEK(773)
130 IF VEC=17165 THEN POKE SA+14,22:POKE SA+18,23:
  POKESA+29,224:POKESA+139,224
140 PRINT CHR$(147);CHR$(17);"PROOFREADER ACTIVE":
  SYS SA
150 POKE HI,PEEK(HI)+1:POKE (PEEK(LO)+256*PEEK(HI)
  )-1,0:NEW
160 DATA 120,169,73,141,4,3,169,3,141,5,3
170 DATA 88,96,165,20,133,167,165,21,133,168,169
180 DATA 0,141,0,255,162,31,181,199,157,227,3
190 DATA 202,16,248,169,19,32,210,255,169,18,32
200 DATA 210,255,160,0,132,180,132,176,136,230,180
210 DATA 200,185,0,2,240,46,201,34,208,8,72
220 DATA 165,176,73,255,133,176,104,72,201,32,208
230 DATA 7,165,176,208,3,104,208,226,104,166,180
240 DATA 24,165,167,121,0,2,133,167,165,168,105
250 DATA 0,133,168,202,208,239,240,202,165,167,69
260 DATA 168,72,41,15,168,185,211,3,32,210,255
270 DATA 104,74,74,74,74,168,185,211,3,32,210
280 DATA 255,162,31,189,227,3,149,199,202,16,248
290 DATA 169,146,32,210,255,76,86,137,65,66,67
300 DATA 68,69,70,71,72,74,75,77,80,81,82,83,88
310 DATA 13,2,7,167,31,32,151,116,117,151,128,129,
  167,136,137
```

Appendix D

MLX

Machine Language Entry Program

Ottis R. Cowper

“MLX” is a labor-saving utility that allows almost fail-safe entry of machine language programs.

Type in and save at least two copies of “MLX” (you’ll want to use it to enter future Commodore 64 machine language programs from *COMPUTE!* magazine, *COMPUTE!’s Gazette*, and *COMPUTE!* books). If you’re using a Commodore 128, you can still use MLX—you must enter it and any ML programs in 64 mode, however.

When you’re ready to enter an ML program, load and run MLX. It asks you for a starting address and an ending address. These addresses appear in the articles accompanying each machine language program.

If you’re unfamiliar with machine language, the addresses (and all other values you enter in MLX) may appear strange. Instead of the usual decimal numbers you’re accustomed to, these numbers are in *hexadecimal*—a base 16 numbering system commonly used by ML programmers. Hexadecimal—*hex* for short—includes the numerals 0–9 and the letters A–F. But don’t worry—even if you know nothing about ML or hex, you should have no trouble using MLX.

After you enter the starting and ending addresses, you’ll be offered the option of clearing the workspace. Choose this option if you’re starting to enter a new listing. If you’re continuing a listing that’s partially typed from a previous session, don’t choose this option.

A functions menu appears. The first option in the menu is ENTER DATA. If you’re just starting to type in a program, pick this. Press the E key, and type the first number in the

first line of the program listing. If you've already typed in part of a program, type the line number where you left off typing at the end of the previous session (be sure to load the partially completed program before you resume entry). In any case, make sure the address you enter corresponds to the address of a line in the listing you are entering. Otherwise, you'll be unable to enter the data correctly. If you press E by mistake, you can return to the command menu by pressing RETURN alone when asked for the address. (You can get back to the menu from most options by pressing RETURN with no other input.)

Entering a Listing

Once you're in Enter mode, MLX prints the address for each program line for you. You then type in all nine numbers on that line, beginning with the first two-digit number after the colon (:). Each line represents eight data bytes and a checksum. Although an MLX-format listing appears similar to the "hex dump" listings from a machine language monitor program, the extra checksum number on the end allows MLX to check your typing.

When you enter a line, MLX recalculates the checksum from the eight bytes and the address, and compares this value with the number from the ninth column. If the values match, you'll hear a bell tone, the data will be added to the workspace area, and the prompt for the next line of data will appear. But if MLX detects a typing error, you'll hear a low buzz and see an error message. The line will then be redisplayed for editing.

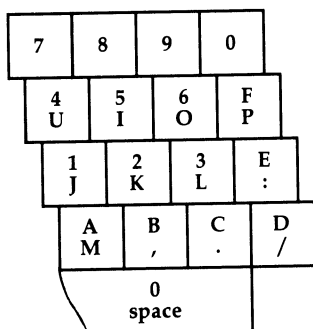
Invalid Characters Banned

Only a few keys are active while you're entering data, so you may have to unlearn some habits. You *do not* type spaces between the columns; MLX automatically inserts these for you. You *do not* press RETURN after typing the last number in a line; MLX automatically enters and checks the line after you've typed the last digit.

Only the numerals 0–9 and the letters A–F can be typed in. If you press any other key (with some exceptions noted below), you'll hear a warning buzz. To simplify typing, a keypad is available. The keypad—part of the keyboard—is active only while you're entering data. Addresses must be entered with

the normal letter and number keys. The figure below shows the keypad configuration.

64 MLX Keypad



MLX checks for transposed characters. If you're supposed to type in A0 and enter 0A instead, MLX will catch your mistake. There is one error that can slip past MLX: Because of the checksum formula used, MLX won't notice if you accidentally type FF in place of 00, and vice versa. And there's a very slim chance that you could garble a line and still end up with a combination of characters that adds up to the proper checksum. However, these mistakes should not occur if you take reasonable care while entering data.

Editing Features

To correct typing mistakes before finishing a line, use the INST/DEL key to delete the character to the left of the cursor. (The cursor-left key also deletes.) If you really mess up a line, press CLR/HOME to restart the line. The RETURN key is also active, but only before any data is typed on a line. Pressing RETURN at this point returns you to the command menu.

After you've typed a character of data, MLX disables RETURN until the cursor returns to the start of a line. Remember, you can press CLR/HOME to get to a line number prompt quickly.

More editing features are available when you're correcting lines in which MLX has detected an error. To make corrections in a line that MLX has redisplayed for editing, compare the

line on the screen with the one printed in the listing; then move the cursor to the mistake and type the correct key. The cursor-left and -right keys provide the normal cursor controls. (The INST/DEL key now works as an alternative cursor-left key.) You cannot move left beyond the first character in the line. If you try to move beyond the rightmost character, you'll reenter the line. During editing, RETURN is active; pressing it tells MLX to recheck the line. You can press the CLR/HOME key to clear the entire line if you want to start from scratch or if you want to get to a line-number prompt to use RETURN to get back to the menu.

Display Data

The second menu choice, DISPLAY DATA, examines memory and shows the contents in the same format as the program listing (including the checksum). When you press D, MLX asks you for a starting address. Be sure that the starting address you give corresponds to a line number in the listing. Otherwise, the checksum display will be meaningless. MLX displays program lines until it reaches the end of the program, at which point the menu is redisplayed. You can pause the display by pressing the space bar. (MLX finishes printing the current line before halting.) Press the space bar again to restart the display. To break out of the display and get back to the menu before the ending address is reached, press RETURN.

Other Menu Options

Two more menu selections let you save programs and load them back into the computer. These are SAVE FILE and LOAD FILE; their operation is quite straightforward. When you press S or L, MLX asks you for the filename. You'll then be asked to press either D or T to select disk or tape.

You'll notice the disk drive starting and stopping several times during a load or save. Don't panic; this is normal behavior. MLX opens and reads from or writes to the file instead of using the usual LOAD and SAVE commands. Disk users should also note that the drive prefix 0: is automatically added to the filename (line 750), so it should *not* be included when you enter the name. This also precludes the use of @ for Save-with-Replace, so remember to give each version you save a different name.

Remember that MLX saves the entire workspace area from the starting address to the ending address, so the save or load may take longer than you might expect if you've entered only a small amount of data from a long listing. When saving a partially completed listing, make sure to note the address where you stopped typing so that you'll know where to resume entry when you reload.

MLX reports the standard disk or tape error messages if any problems are detected during the save or load. (Tape users should bear in mind that Commodore computers are never able to detect errors during a save to tape.) MLX also has three special load-error messages: INCORRECT STARTING ADDRESS, which means the file you're trying to load does not have the starting address you specified when you ran MLX; LOAD ENDED AT *address*, which means the file you're trying to load ends before the ending address you specified when you started MLX; and TRUNCATED AT ENDING ADDRESS, which means the file you're trying to load extends beyond the ending address you specified when you started MLX. If you see one of these messages and feel certain that you've loaded the right file, exit and rerun MLX, being careful to enter the correct starting and ending addresses.

The QUIT menu option has the obvious effect—it stops MLX and enters BASIC. The RUN/STOP key is disabled, so the Q option lets you exit the program without turning off the computer. (Of course, RUN/STOP-RESTORE also gets you out.) You'll be asked for verification; press Y to exit to BASIC or any other key to return to the menu. After quitting, you can type RUN again and reenter MLX without losing your data, as long as you don't use the clear workspace option.

The Finished Product

When you've finished typing all the data for an ML program and saved your work, you're ready to see the results. The instructions for loading and using the finished product vary from program to program. Some ML programs are designed to be loaded and run like BASIC programs, so all you need to type is LOAD "*filename*",8 for disk or LOAD "*filename*" for tape, and then RUN. Such programs will usually have a starting address of 0801. Other programs must be reloaded to specific addresses with a command such as LOAD "*filename*",8,1 for disk or LOAD "*filename*",1,1 for tape, then started with a

SYS to a particular memory address. On the Commodore 64, the most common starting address for such programs is 49152, which corresponds to MLX address C000.

An Ounce of Prevention

By the time you finish typing in the data for a long ML program, you may have several hours invested in the project. Don't take chances—read Appendix B, "How to Type In Programs," use our "Automatic Proofreader" to type in MLX, and then test your copy *thoroughly* before first using it to enter any significant amount of data. Make sure all the menu options work as they should. Enter fragments of the program starting at several different addresses; then use the Display option to verify that the data has been entered correctly. And be sure to test the Save and Load options several times to insure that you can recall your work from disk or tape. Don't let a simple typing error in the new MLX cost you several nights of hard work.

MLX

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

```
SS 10 REM VERSION 1.1: LINES 830,950 MODIFIED, LINES 485-487 ADDED
EK 100 POKE 56,50:CLR:DIM I$,I,J,A,B,A$,B$,A(7),N$
DM 110 C4=48:C6=16:C7=7:Z2=2:Z4=254:Z5=255:Z6=256:Z7=127
CJ 120 FA=PEEK(45)+Z6*PEEK(46):BS=PEEK(55)+Z6*PEEK(56):H$="0123456789ABCDEF"
SB 130 R$=CHR$(13):L$="{LEFT}":S$=" ":D$=CHR$(20):Z$=CHR$(0):T$="{13 RIGHT}"
CQ 140 SD=54272:FOR I=SD TO SD+23:POKE I,0:NEXT:POKE SD+24,15:POKE 788,52
FC 150 PRINT "{CLR}"CHR$(142)CHR$(8):POKE 53280,15:POKE 53281,15
EJ 160 PRINT T$" {RED}{RVS}{2 SPACES}[8 @]{2 SPACES}"SPC(28)"{2 SPACES}{OFF}{BLU} MLX II {RED}{RVS}{2 SPACES}"SPC(28)"{12 SPACES}{BLU}"
FR 170 PRINT "{3 DOWN}{3 SPACES}COMPUTE!'S MACHINE {SPACE}LANGUAGE EDITOR{3 DOWN}"
JB 180 PRINT "{BLK}STARTING ADDRESS[4]";:GOSUB300:S A=AD:GOSUB1040:IF F THEN180
GF 190 PRINT "{BLK}{2 SPACES}ENDING ADDRESS[4]";:GOSUB300:EA=AD:GOSUB1030:IF F THEN190
```

```

KR 200 INPUT "{3 DOWN}{BLK}CLEAR WORKSPACE [Y/N][4]"
      ";A$:IF LEFT$(A$,1)<>"Y"THEN220
PG 210 PRINT "{2 DOWN}{BLU}WORKING...";:FORI=BS TO
      {SPACE}BS+EA-SA+7:POKE I,0:NEXT:PRINT"DONE"
DR 220 PRINTTAB(10)" {2 DOWN}{BLK}{RVS} MLX COMMAND
      MENU {DOWN}[4]":PRINT T$"{RVS}E{OFF}NTER D
      ATA"
BD 230 PRINT T$"{RVS}D{OFF}ISPLAY DATA":PRINT T$"
      {RVS}L{OFF}OAD FILE"
JS 240 PRINT T$"{RVS}S{OFF}AVE FILE":PRINT T$"
      {RVS}Q{OFF}UIT{2 DOWN}{BLK}"
JH 250 GET A$:IF A$=N$ THEN250
HK 260 A=0:FOR I=1 TO 5:IF A$=MID$("EDLSQ",I,1)THE
      N A=I:I=5
FD 270 NEXT:ON A GOTO420,610,690,700,280:GOSUB1060
      :GOTO250
EJ 280 PRINT "{RVS} QUIT ":INPUT "{DOWN}[4]ARE YOU S
      URE [Y/N]";A$:IF LEFT$(A$,1)<>"Y"THEN220
EM 290 POKE SD+24,0:END
JX 300 IN$=N$:AD=0:INPUTIN$:IFLEN(IN$)<>4THENRETUR
      N
KF 310 B$=IN$:GOSUB320:AD=A:B$=MID$(IN$,3):GOSUB32
      0:AD=AD*256+A:RETURN
PP 320 A=0:FOR J=1 TO 2:A$=MID$(B$,J,1):B=ASC(A$)-
      C4+(A$>"@")*C7:A=A*C6+B
JA 330 IF B<0 OR B>15 THEN AD=0:A=-1:J=2
GX 340 NEXT:RETURN
CH 350 B=INT(A/C6):PRINT MID$(H$,B+1,1);:B=A-B*C6:
      PRINT MID$(H$,B+1,1);:RETURN
RR 360 A=INT(AD/Z6):GOSUB350:A=AD-A*Z6:GOSUB350:PR
      INT":";
BE 370 CK=INT(AD/Z6):CK=AD-Z4*CK+Z5*(CK>Z7):GOTO39
      0
PX 380 CK=CK*Z2+Z5*(CK>Z7)+A
JC 390 CK=CK+Z5*(CK>Z5):RETURN
QS 400 PRINT "{DOWN}STARTING AT[4]";:GOSUB300:IF IN
      $<>N$ THEN GOSUB1030:IF F THEN400
EX 410 RETURN
HD 420 PRINT "{RVS} ENTER DATA ":GOSUB400:IF IN$=N$
      THEN220
JK 430 OPEN3,3:PRINT
SK 440 POKE198,0:GOSUB360:IF F THEN PRINT IN$:PRIN
      T"{UP}{5 RIGHT}";
GC 450 FOR I=0 TO 24 STEP 3:B$=S$:FOR J=1 TO 2:IF
      {SPACE}F THEN B$=MID$(IN$,I+J,1)
HA 460 PRINT "{RVS}"B$L$;:IF I<24THEN PRINT "{OFF}";
HD 470 GET A$:IF A$=N$ THEN470
FK 480 IF (A$>"/"ANDAS<"")OR(A$>"@"ANDAS<"G")THEN5
      40
GS 485 A=- (A$="M")-2*(A$="," )-3*(A$="." )-4*(A$="/"
      )-5*(A$="J")-6*(A$="K")

```

```
FX 486 A=A-7*(A$="L")-8*(A$=":")-9*(A$="U")-10*(A$
      ="I")-11*(A$="O")-12*(A$="P")
CM 487 A=A-13*(A$=S$):IF A THEN A$=MID$("ABCD123E4
      56F0",A,1):GOTO 540
MP 490 IF A$=R$ AND((I=0)AND(J=1)OR F)THEN PRINT B
      $;J=2:NEXT:I=24:GOTO550
KC 500 IF A$="{HOME}" THEN PRINT B$:J=2:NEXT:I=24:
      NEXT:F=0:GOTO440
MX 510 IF(A$="{RIGHT}")ANDF THENPRINT B$;L$;:GOTO54
      0
GK 520 IF A$<>L$ AND A$<>D$ OR((I=0)AND(J=1))THEN
      {SPACE}GOSUB1060:GOTO470
HG 530 A$=L$+S$+L$:PRINT B$;L$;:J=2-J:IF J THEN PRI
      NT L$;:I=I-3
QS 540 PRINT A$;:NEXT J:PRINT S$;
PM 550 NEXT I:PRINT:PRINT"{UP}{5 RIGHT}";:INPUT#3,
      IN$:IF IN$=N$ THEN CLOSE3:GOTO220
QC 560 FOR I=1 TO 25 STEP3:B$=MID$(IN$,I):GOSUB320
      :IF I<25? THEN GOSUB380:A(I/3)=A
PK 570 NEXT:I:IF A<>CK THEN GOSUB1060:PRINT"{BLK}
      {RVS} ERROR: REENTER LINE [4]":F=1:GOTO440
HJ 580 GOSUB1080:B=BS+AD-SA:FOR I=0 TO 7:POKE B+I,
      A(I):NEXT
QQ 590 AD=AD+8:IF AD>EA THEN CLOSE3:PRINT"{DOWN}
      {BLU}** END OF ENTRY **{BLK}{2 DOWN}":GOTO7
      00
GQ 600 F=0:GOTO440
QA 610 PRINT"{CLR}{DOWN}{RVS} DISPLAY DATA ":GOSUB
      400:IF IN$=N$ THEN220
RJ 620 PRINT"{DOWN}{BLU}PRESS: {RVS}SPACE{OFF} TO
      {SPACE}PAUSE, {RVS}RETURN{OFF} TO BREAK[4]
      {DOWN}"
KS 630 GOSUB360:B=BS+AD-SA:FORI=BTO B+7:A=PEEK(I):
      GOSUB350:GOSUB380:PRINT S$;
CC 640 NEXT:PRINT"{RVS}";:A=CK:GOSUB350:PRINT
KH 650 F=1:AD=AD+8:IF AD>EA THENPRINT"{DOWN}{BLU}*
      * END OF DATA **":GOTO220
KC 660 GET A$:IF A$=R$ THEN GOSUB1080:GOTO220
EQ 670 IF A$=S$ THEN F=F+1:GOSUB1080
AD 680 ONFGOTO630,660,630
CM 690 PRINT"{DOWN}{RVS} LOAD DATA ":OP=1:GOTO710
PC 700 PRINT"{DOWN}{RVS} SAVE FILE ":OP=0
RX 710 IN$=N$:INPUT"{DOWN}FILENAME[4]";IN$:IF IN$=
      N$ THEN220
PR 720 F=0:PRINT"{DOWN}{BLK}{RVS}T{OFF}APE OR
      {RVS}D{OFF}ISK: [4]";
FP 730 GET A$:IF A$="T"THEN PRINT"T{DOWN}":GOTO880
HQ 740 IF A$<>"D"THEN730
```

```

HH 750 PRINT"D{DOWN}":OPEN15,8,15,"I0:":B=EA-SA:IN
    $="0: "+IN$:IF OP THEN810
SQ 760 OPEN 1,8,8,IN$+" ,P,W":GOSUB860:IF A THEN220
FJ 770 AH=INT(SA/256):AL=SA-(AH*256):PRINT#1,CHR$(
    AL);CHR$(AH);
PE 780 FOR I=0 TO B:PRINT#1,CHR$(PEEK(BS+I));:IF S
    T THEN800
FC 790 NEXT:CLOSE1:CLOSE15:GOTO940
GS 800 GOSUB1060:PRINT"{DOWN}{BLK}ERROR DURING SAV
    E:[4]":GOSUB860:GOTO220
MA 810 OPEN 1,8,8,IN$+" ,P,R":GOSUB860:IF A THEN220
GE 820 GET#1,A$,B$:AD=ASC(A$+Z$)+256*ASC(B$+Z$):IF
    AD<>SA THEN F=1:GOTO850
RX 830 FOR I=0 TO B:GET#1,A$:POKE BS+I,ASC(A$+Z$):
    IF(I<>B)AND ST THEN F=2:AD=I:I=B
FA 840 NEXT:IF ST<>64 THEN F=3
FQ 850 CLOSE1:CLOSE15:ON ABS(F>0)+1 GOTO960,970
SA 860 INPUT#15,A,A$:IF A THEN CLOSE1:CLOSE15:GOSU
    B1060:PRINT"{RVS}ERROR: "A$
GQ 870 RETURN
EJ 880 POKE183,PEEK(FA+2):POKE187,PEEK(FA+3):POKE1
    88,PEEK(FA+4):IFOP=0 THEN920
HJ 890 SYS 63466:IF(PEEK(783)AND1)THEN GOSUB1060:P
    RINT"{DOWN}{RVS} FILE NOT FOUND ":GOTO690
CS 900 AD=PEEK(829)+256*PEEK(830):IF AD<>SA THEN F
    =1:GOTO970
SC 910 A=PEEK(831)+256*PEEK(832)-1:F=F-2*(A<EA)-3*
    (A>EA):AD=A-AD:GOTO930
KM 920 A=SA:B=EA+1:GOSUB1010:POKE780,3:SYS 63338
JF 930 A=BS:B=BS+(EA-SA)+1:GOSUB1010:ON OP GOTO950
    :SYS 63591
AE 940 GOSUB1080:PRINT"{BLU}** SAVE COMPLETED **":
    GOTO220
XP 950 POKE147,0:SYS 63562:IF ST>0 THEN970
FR 960 GOSUB1080:PRINT"{BLU}** LOAD COMPLETED **":
    GOTO220
DP 970 GOSUB1060:PRINT"{BLK}{RVS}ERROR DURING LOAD
    :{DOWN}[4]":ON F GOSUB980,990,1000:GOTO220
PP 980 PRINT"INCORRECT STARTING ADDRESS ( ";:GOSUB3
    60:PRINT")":RETURN
GR 990 PRINT"LOAD ENDED AT ";:AD=SA+AD:GOSUB360:PR
    INT D$:RETURN
FD 1000 PRINT"TRUNCATED AT ENDING ADDRESS":RETURN
RX 1010 AH=INT(A/256):AL=A-(AH*256):POKE193,AL:POK
    E194,AH
FF 1020 AH=INT(B/256):AL=B-(AH*256):POKE174,AL:POK
    E175,AH:RETURN
FX 1030 IF AD<SA OR AD>EA THEN1050
HA 1040 IF(AD>511 AND AD<40960)OR(AD>49151 AND AD<
    53248)THEN GOSUB1080:F=0:RETURN

```

```
HC 1050 GOSUB1060:PRINT"{RVS} INVALID ADDRESS  
      {DOWN}{BLK}":F=1:RETURN  
AR 1060 POKE SD+5,31:POKE SD+6,208:POKE SD,240:POK  
      E SD+1,4:POKE SD+4,33  
DX 1070 FOR S=1 TO 100:NEXT:GOTO1090  
PF 1080 POKE SD+5,8:POKE SD+6,240:POKE SD,0:POKE S  
      D+1,90:POKE SD+4,17  
AC 1090 FOR S=1 TO 100:NEXT:POKE SD+4,0:POKE SD,0:  
      POKE SD+1,0:RETURN
```


Index

- "Address Cataloger"
 - editing functions 139-41
 - program listing 143-51
- !AGAIN 245
- "Animation Demo" 278-79
- "Animation Player" program listing 280
- arithmetic operators 81
- array 201
- ASC 204
- assembler commands 227-30
- assembling source code 227
- asterisk 226
- AUTO 225
- "Automatic Proofreader, The" program listing 346
- BAM 290
- "Banners for Commodore 1525, 801, 803" program listing 159-60
- "Banners for Commodore 1526, 802" program listing 160-62
- BASIC interpreter 223, 224
 - modifications and enhancements 224-25
 - new BASIC commands 225-26
- block allocation map. *See* BAM
- !BOX 246
- braces 339, 341
- brackets 341
- "Brick the Wall" program listing 15-17
- "Budget Planner" program listing 130-36
- BYTE 229
- CAN'T CONTINUE error message 322
- cassette buffer space 284
- !CENTER 246
- CGET 263
- chaining programs 233
- characters, customized 153
- CLEAR 262
- !CLOSE 244
- CLR 229
- !CLR 245
- colons 153
- Comets (module in "Solarpix") 66
- commas 153
- Commodore key 341
- Commodore 64 Graphics 159
- Commodore 64 memory map figures 324
- Commodore 64 Programmer's Reference Guide 158
- compiler 199, 200
- COMPUTE!'s First Book of Commodore 64 159
- COMPUTE!'s First Book of Commodore 64 Sound and Graphics 159
- concatenated 202
- constants 201
 - integer 201
 - string 201
- "Construction Set, The"
 - "Creature Character Set" program listing 115-20
 - "House Character Set" program listing 110-15
 - main program listing 106-10
 - "Shapes Character Set" program listing 120-25
- contractions 167
- "Coordinator, The"
 - color sources figure 326
 - program listing 330-32
- copying disks 298
- CPUT 263
- Creating Arcade Games on the 64 159
- "Custom Labels" program listing 155-56
- DATA 202
- DATA statements 340
- DEF 229
- !DEFINE 242
- "Demo" program listing 333-34
- destination disk 299
- device number 228
- DGET 263
- dictionary disk, creating 166
- "Dictionary Manager" program listing 175-78
- DIM 202
- !DIM 242
- "Directory Extension" program listing 309-13
- !DISPLAY 246
- division sign 82
- dollar sign 201
- Doodle 74
- double-width printing 154
- DPUT 263
- "Eagles and Gators" program listing 39-53

Easy Script 166
 eccentricity 66
 eclipse 66
 elliptical orbits 66
 ending address 347
 enhanced printing 154
 !ERASE 246
 Facts (module in "Solarpix") 67
 "Fast Assembler, The" program listing
 234-40
 FILL 261-62
 "Fill 64"
 commands 261-63
 program listing 265-77
 FLIP 262
 FOR 203
 formatting a disk 289-91
 garbage characters, printing 142
 GET 203
 global 229, 233
 hexadecimal 347
 hidden RAM 284
 hi-res 322
 load 328-29
 save 328-29
 !HUE 246
 hyphen 167
 "Idol of Orion, The" 56-61
 IF 203
 ILLEGAL ADDRESSING MODE ERROR
 227
 ILLEGAL DIRECT ERROR 227
 ILLEGAL QUANTITY ERROR 247
 INCLUDE 229, 233
 INPUT 204, 232
 integer arrays 201
 interactive 165
 interpreter 199
 interrupt 38
 keyboard conventions 343
 !KINETIC 245
KoalaPainter 74
 labels 226-27, 152
 linefeed 142
 !LOAD 244
 LOWRES 262
 machine language 347
 machine language assembler 223
 mass 67
 "Math Attack" program listing 96-99
 MEMORY 263
 memory 284
 memory location 36
 !MENU 245
 MID\$ 204
 minus sign 82

MISSING LINE NUMBER ERROR 230
 MLX keypad, Commodore 64 349
 "MLX" program listing 352-56
 "Mountain Demo" program listing
 334-35
 multiplication sign 82
 NEW 229
 "Number Construction Kit" program
 listing 83-85
 numeric variables 201
 object code 223, 229
 "Object Maker" program listing 333
 OLD 226
 !OPEN 243
 ORG 228, 230
 origin 228
Paperback Writer 191
PaperClip 165, 166, 174
 parentheses 202
 patch 315
 percent sign 201
 "Pirate Cove" program listing 87-93
 PLAY 262
 plus sign 82
 !PRINT 246
 printer modifications 154
 program counter 228
Programming the Commodore 64 329
 quotation marks 201
 quote mode 341-42
 "Raceway" program listing 4-13
 "RAM Expansion Demo" program
 listing 286
 raster 36
 raster displays 36
 read errors 300
 RECORD 262
 "ReDisk" 168
 register 36
 reserved words 201
 !RESTORE 244
 RETURN 229
 REWIND 262
 "Ringside Karate" 19-33
 RND 204
 !SAVE 244
 "Save/Load Subroutine" 335-36
 !SCROLL 245
 search string 141
 semicolon 226
 SEND 229, 232
 "Sequential File Converter for *Speed-Script*"
 program listing 194-96
 "Sequential File Reader" program
 listing 282-83
 sequential files 193, 281

SETCOLOR 262
 shadow registers 37–38
 shifted key 341
 “64 Doodler Demo” 221–22
 “64 RAM Expander” program listing 286
 “Solarpix” program listing 68–73
 “Sound Patcher” program listing 315
 source code 223
 source disk 299
 “SpeedCheck”
 active keys 170–73
 customizing 173–74
 program listing 178–85
SpeedScript 165, 166, 167, 174, 186, 187, 190, 191, 193
 “SpeedView” program listing 188–89
 spelling checkers 165
 split screens 36–37
 spreadsheet 126
 “Sprint”
 keywords 203
 error messages 205
 program listing 206–21
 sprites 34, 36–39
 starting address 347
STRING TOO LONG ERROR 247
 string variables 201
 Sun and Planets (module in “Solarpix”) 65
 Sun, Earth, and Moon (module in “Solarpix”) 66
 “Super Synth” program listing 318–21
 SWITCH 262
SYNTAX ERROR 246
 synthesizer 314
 !TONE 246
 “Turbo Copy” program listing 301–07
 “Turbo Format” program listing 292–97
 typing in programs 339–40, 341–43
 “Ultrafont +” 105
UNDEF'D LOCATION COUNTER ERROR 228
 underlining 142
 “Unified File Maker” program listing 189
 UNSEND 230, 232
 USR files 194
 “Video Jigsaw” program listing 75–80
 “Violin Lessons” 315
 window 241
 “Window Demo” program listing 256–59
 “Window Wizard”
 commands 242–47
 error trapping 247
 program listing 248–56
 WORD 229
 “Word Counter” program listing 192
Word Writer 128 191
WordPro 165, 166, 174
 write error 300

To order your copy of *COMPUTE!'s Commodore 64/128 Collection, Volume 2 Disk*, call our toll-free US order line: 1-800-346-6767 (in NY 212-887-8525) or send your prepaid order to:

*COMPUTE!'s Commodore 64/128 Collection,
Volume 2 Disk*

COMPUTE! Publications

P.O. Box 5038

F.D.R. Station

New York, NY 10150

All orders must be prepaid (check, charge, or money order). NC residents add 5% sales tax. NY residents add 8.25% sales tax.

Send _____ copies of *COMPUTE!'s Commodore 64/128 Collection, Volume 2 Disk* at \$12.95 per copy. (890BDSK)

Subtotal \$_____

Shipping and Handling: \$2.00/disk \$_____

Sales tax (if applicable) \$_____

Total payment enclosed \$_____

☐ Payment enclosed

☐ Charge ☐ Visa ☐ MasterCard ☐ American Express

Acct. No. _____ Exp. Date _____
(Required)

Name _____

Address _____

City _____ State _____ Zip _____

Please allow 4-5 weeks for delivery.

Unlock the Hidden Power of your Commodore Computer with **COMPUTE!'S GAZETTE**

**More fun...More challenge...
More all new programs
each and every month.**

Subscribe to *COMPUTE!'s Gazette* through this special money-saving introductory offer—and start unleashing the full power of your Commodore computer.

Month after month, look to *COMPUTE!'s Gazette* to deliver the latest inside word on everything from short programming tips to the best new software. Our expert analysis and insights mean you have more fun...get more enjoyment...more of what you bought your computer for.

As a subscriber, you also receive up to 20 all-new, action-packed programs each month. Every big issue of *COMPUTE!'s Gazette* comes complete with a steady supply of the most useful, the most entertaining, the

highest quality programs like Number Quest, Address File, Treasure Hunt, Castle Dungeon, Vocab Builder, SpeedScript, and hundreds of other educational, home finance, and game programs.

So subscribe today—and unleash the hidden power of your Commodore computer. Return the card below—or call 1-800-247-5470 (in Iowa, 1-800-532-1272). Do it now.



FOR COMMODORE PERSONAL COMPUTER USERS

Inside this issue: One of the most incredible arcade-style games we've ever published.

Saloon Shootout

An action-packed, challenging game for the Commodore 64, Plus/4, and 16



Hi-Res Dump
Quick printouts of your screen artwork with this program for the 128, 64, and Plus/4.

Power BASIC:
64 RAM Disk
Now...



Budget Planner
Manage your money easy way with this mini-spreadsheet design to handle weekly and monthly income for an entire year. For the Commodore 128, 64, and Plus/4.

Also in This Issue:
A Home...



SAVE
50%

Subscription Savings Card

☐ **YES!**

I know a great deal when I see one. Sign me up for 12 big issues of *COMPUTE!'s Gazette* for just \$18. I save 50% off the newsstand price.

☐ Payment enclosed ☐ Bill me ☐ Charge my VISA/MasterCard

Credit Card # _____ Exp. Date _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

Outside U.S.A. please add \$6 (U.S.) per year for postage.

Say YES now to **COMPUTE!'s GAZETTE**

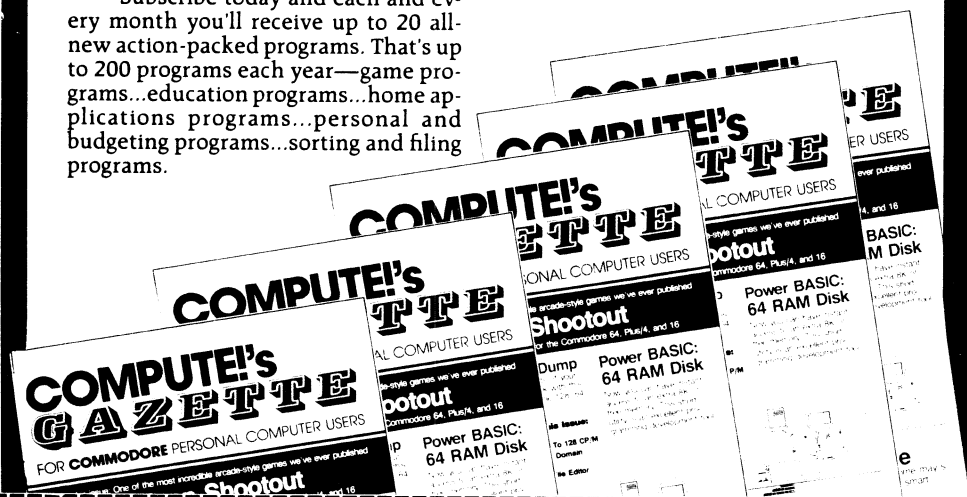
You can search far and wide and you simply *won't* find a better magazine...a better source of insightful, stimulating, usable information for your Commodore computer than *COMPUTE!'s Gazette*.

COMPUTE!'s Gazette works harder...digs deeper...researches further—all to help guarantee that you get the absolute most from your Commodore.

Subscribe today and each and every month you'll receive up to 20 all-new action-packed programs. That's up to 200 programs each year—game programs...education programs...home applications programs...personal and budgeting programs...sorting and filing programs.

Add it up for yourself. Where else can you get exciting programs each month...expert advice...insightful analysis...up-to-the-minute software reviews and so much more—all for just \$18.

So why wait. Subscribe now to *COMPUTE!'s Gazette*—and get the most from your Commodore computer. Return card below—or call 1-800-247-5470 (in Iowa, 1-800-532-1272).



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 7551 DES MOINES, IA

POSTAGE WILL BE PAID BY ADDRESSEE

COMPUTE!'s Gazette

P.O. Box 10775

Des Moines, IA 50347-0775



Never Fade Away

Continuing in COMPUTE! Publications' tradition of support for Commodore computers, *COMPUTE!'s Commodore 64/128 Collection, Volume 2*, is an anthology of our best and latest games, applications, educational software, programming utilities—some published, some never before seen.

With more than 40 programs, all ready to type in and run on the 64, 64C, or 128 (in 64 mode), this collection presents a variety of high-quality, low-cost software for your computer. Here's just a sample of what you'll find inside:

- Arcade games like "Raceway" and "Ringside Karate" that test your joystick skills to the limit
- Educational games and programs where you display the solar system, explore islands for treasure, and play with numbers
- Home-productivity software such as "Budget Planner" and "Address Cataloger"
- An entire chapter of enhancements and utilities for *SpeedScript*, COMPUTE! Publications' ever-popular word processor
- Programming applications like "Sprint," a BASIC compiler, and "The Fast Assembler," an easy-to-use machine language assembler
- General utilities, from "Turbo Format" to "Directory Extension"

Take a look at what your computer can do with *COMPUTE!'s Commodore 64/128 Collection, Volume 2* and its programs. You'll be impressed—perhaps even amazed.

All the programs in this book are also available on a companion disk. See coupon in the back for details.

ISBN 0-87455-089-0