COMPUTE!'s

Commodore Collection

Twenty-eight original programs for the VIC and 64

VOLUME ONE

Never-before-published games, educational programs, applications, and utilities for the VIC-20 and Commodore 64.

A **COMPUTE! Books** Publication
$12.95

COMPUTE!'s

Commodore Collection

VOLUME ONE

COMPUTE! Publications, Inc.
One of the ABC Publishing Companies
Greensboro, North Carolina

Printed in the United States of America

ISBN 0-942386-55-8

10 9 8 7 6 5 4 3 2 1

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

Contents

Foreword

Welcome to *COMPUTE!'s Commodore Collection, Volume 1.* It's packed with articles (all published here for the first time) that are designed to make your Commodore computer more useful and exciting than ever before. Fascinating games, versatile educational programs, and practical applications — this book has them all.

Have you ever explored an abandoned castle? "Save the King," a fascinating three-dimensional adventure game, gives you the opportunity.

Is your youngster learning the multiplication tables? "Spider Math" can help.

Do you have a record collection or program library that you would like to catalog? With "File Cabinet" there's nothing to it.

Programmers, too, will find this book extremely useful. Do you need to design a custom character set using multicolor characters? This book shows you how. You'll learn how to turn your Commodore into a numeric keypad, how to automatically format columns of numbers, even how to add a PRINT AT command to your BASIC repertoire.

Whether you're a novice or an expert, this book is sure to be of great value — and you don't have to be a computer technician to use it. Each program has been thoroughly tested, and all are ready to run. Just type them in. But if you do want to modify a listing, that's easy too. In fact, many of the articles include specific instructions to help you customize the routines to suit your particular needs.

You'll enjoy *COMPUTE!'s Commodore Collection, Volume 1.* But just as important, you'll use it again and again.

Chapter 1

Games

Hang Glider

Alan Keyser

Tired of wearing out your fingers playing keyboard games? Break out your joystick and get ready for "Hang Glider," a challenging arcade-style game for the unexpanded VIC or the Commodore 64.

For Homer Propless, hang glider ace, it starts like any other Friday. A few laps around the track, a big steak for lunch, then off to the desert to ride the afternoon thermals.

His friends Rupert and Roscoe help him launch his glider off Stack Mesa, and it isn't long till Homer is soaring far above the desert terrain. From an altitude of 500 feet he has quite a view, too. On his left he can see the mountains, almost a hundred miles away. On his right is the blue expanse of a huge lake. And straight ahead . . . what?

Giant green pterodactyls are making off with his friends!

That's right: Flying pterodactyls have kidnapped not only Rupert and Roscoe but the entire population of Palm Springs. Now it's all up to Homer. Using his trusty blue hang glider, he must rescue the captives from the backs of the giant birds, which are at this very moment escaping across the lake. Can he do it? Only if he's careful — and only with your help.

Use your joystick to guide Homer's hang glider right, left, and up. If you get too low, glide into the thermal updraft at the right of the screen. Be careful, though. If you collide with a bird, run into the side of a mesa, or crash into the lake, it's all over.

You'll start each round with six hang gliders. Every time you rescue one of the captives, you'll get 100 points; every time you crash, you'll lose one of your gliders. The score is displayed on the screen, and the game is over when all six gliders are gone.

Palm Springs is counting on you.

Program 1. Hang Glider, VIC Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
1 PRINT"{CLR}"                                    :rem 149
1Ø POKE56,28:POKE52,28:CLR                         :rem 18
2Ø FORI=7168TO7679:POKEI,PEEK(I+256ØØ):NEXTI
                                                  :rem 172
5Ø FORI=7448TO7551:READJ:POKEI,J:NEXTI            :rem 168
6Ø POKE36869,255                                  :rem 1Ø9
65 DIMB(2,2):A=4                                   :rem 84
```

```
70 POKE36879,29:SP=1:X=16:Y=1:X1=-2:Y1=1:PL=32:CO=
   30720:NI=1:FS=240                          :rem 179
80 GOSUB800                                   :rem 128
90 GOTO500                                    :rem 55
100 POKE37154,127:D=(PEEK(37137)AND28)OR(PEEK(3715
    2)AND128)                                 :rem 38
110 IFD=156ANDX<17THENY1=1                    :rem 52
115 IFD=156ANDPEEK(L+22)=44THENX1=0:Y1=0      :rem 98
120 IFD=152THENY1=-2:SP=SP-1:FS=FS+1          :rem 176
130 IFD=28THENX1=2:Y1=1:SP=SP+1:FS=246        :rem 44
140 IFD=140THENX1=-2:Y1=1:SP=SP+1:FS=246      :rem 133
150 IFD=148THENY1=2:SP=SP+1:FS=FS+1:RETURN:rem 163
151 REM ***SP>3ALLOWED ABOVE***                :rem 0
160 IFSP>3THENSP=3                            :rem 124
165 IFSP<0THENY1=2:SP=0                       :rem 172
170 RETURN                                    :rem 120
200 IFNI=1THENNI=2:GOTO205                    :rem 101
202 IFNI=2THENNI=1                            :rem 93
205 I=NI                                      :rem 180
210 IFB(I,0)>0THEN250                         :rem 142
220 B(I,1)=INT(RND(1)*10)+5                   :rem 151
225 IFPEEK(B(I,1)*22+7684)<>32THEN220         :rem 11
230 B(I,0)=4                                  :rem 62
240 B(I,2)=1                                  :rem 62
250 BL=7680+B(I,0)+B(I,1)*22                  :rem 140
255 POKEBL,32:POKEBL-1,32:POKEBL-2,32:POKEBL-23,32
                                              :rem 209
260 B(I,0)=B(I,0)+1:IFRND(1)>.9THENB(I,1)=B(I,1)+1
                                              :rem 175
265 BL=7680+B(I,0)+B(I,1)*22                  :rem 146
270 IFPEEK(BL)<>32ANDPEEK(BL)<>35THENB(I,0)=0:GOTO
    295                                       :rem 171
275 IFPEEK(BL)=35ORPEEK(BL-1)=35ORPEEK(BL-2)=35THE
    N400                                      :rem 234
280 IFPEEK(BL-23)=35ANDB(I,2)=1THENSCO=SCO+100:B(I
    ,2)=0:GOSUB300                            :rem 255
285 POKEBL,39:POKEBL+CO,5:POKEBL-1,37:POKEBL-1+CO,
    5:POKEBL-2,38:POKEBL-2+CO,5               :rem 200
290 IFB(I,2)=1THENPOKEBL-23,36:POKEBL-23+CO,2
                                              :rem 137
295 IFRND(1)>.9THENGOSUB370                   :rem 131
299 RETURN                                    :rem 132
300 POKE36876,232:FORQ=1TO100:NEXTQ:POKE36876,240:
    FORQ=1TO200:NEXTQ:POKE36876,0              :rem 92
310 PL=32:RETURN                              :rem 236
370 FORQ=220TO250:POKE36876,Q:NEXT:FORQ=250TO230ST
    EP-1:POKE36876,Q:NEXTQ:POKE36876,0        :rem 243
375 RETURN                                    :rem 127
400 REM DIE                                    :rem 74
```

```
410 POKE36877,220:FORL=15TO0STEP-1:POKE36878,L:FOR
    M=1TO100:NEXTM,L                          :rem 79
420 A=A-1:IFA<-1THEN460                       :rem 32
450 GOTO70                                    :rem 57
460 PRINT"{CLR}{8 DOWN}{6 RIGHT}PLAY AGAIN?":rem 9
470 GETA$:IFA$=""THEN470                      :rem 89
480 IFA$<>"Y"THENEND                         :rem 173
490 CLR:GOTO65                                :rem 92
500 GOSUB200                                 :rem 167
501 GOSUB100                                 :rem 167
502 IFFS>240ANDD<>152ANDD<>148THENFS=FS-1    :rem 230
503 POKE36877,FS                             :rem 155
510 POKEL,PL                                 :rem 217
520 X=X+X1:Y=Y+Y1                             :rem 22
530 IFX>21THENX=X-22                         :rem 205
535 IFX<0THENX=22+X                          :rem 155
540 IFY>21THEN400                            :rem 229
545 IFY<1THENY=1                             :rem 233
550 L1=7680+X+Y*22                           :rem 190
555 PL=PEEK(L1)                              :rem 107
560 IFPL>39ANDPL<43ANDD=156THEN700            :rem 56
570 IFPL=44THEN400                            :rem 47
580 IFPL>36ANDPL<40THEN400                    :rem 65
590 IFPL<>36THEN670                          :rem 120
600 SCO=SCO+100:GOSUB300                     :rem 166
610 FORI=1TO2:IFL1=7657+(B(I,0)+B(I,1)*22)THENB(I,
    2)=0                                     :rem 226
620 NEXTI                                     :rem 32
670 POKEL1,35:POKEL1+30720,6                  :rem 76
675 L=L1                                     :rem 168
679 PRINT"{HOME}{BLK}{5 RIGHT}";SCO          :rem 203
680 GOTO500                                  :rem 108
700 L1=L1-22:X1=0:Y1=0                       :rem 192
710 IFY>0THENY=Y-1                           :rem 106
720 IFL1<7702THENL1=L1+22                    :rem 166
730 PL=PEEK(L1):IFPL=35THENPL=PEEK(L1+22)    :rem 250
740 POKEL1,35:L=L1                            :rem 27
750 GOTO680                                  :rem 115
800 L=7720:PRINT"{CLR}"                      :rem 143
810 FORI=7790TO8164STEP22:POKEI,44:POKEI+1,44:POKE
    I+30720,0:POKEI+30721,0:NEXTI              :rem 8
820 FORI=8166TO8185:POKEI,43:POKEI+30720,6:NEXTI
                                             :rem 151
830 FORI=8084TO8172STEP22:FORJ=0TO2:POKEI+J,44:POK
    EI+J+30720,0:NEXTJ,I                      :rem 59
840 FORI=8159TO7719STEP-22:POKEI,40:POKEI+1,41:POK
    EI+2,42:FORJ=0TO2:POKEJ+I+30720,6          :rem 7
845 NEXTJ,I                                  :rem 159
850 FORI=7691TO7691+A:POKEI,47:POKEI+30720,0:NEXT:
    IFA=-1THENPOKE7691,32                     :rem 41
```

```
860 FORI=38415TO38421:POKEI,6:NEXT          :rem 66
870 POKE7695,45:POKE7701,46:FORI=7696TO7700:POKEI,
    44:NEXT                                 :rem 191
880 PRINT"{HOME}{RVS}{BLK}SCORE:{OFF}"      :rem 110
885 POKE36878,15:POKE36877,240              :rem 176
890 RETURN                                  :rem 129
1000 DATA0,0,102,153,0,24,24,0              :rem 199
1010 DATA56,60,24,16,24,20,18,16             :rem 70
1020 DATA60,126,255,255,222,56,112,192      :rem 121
1030 DATA31,31,31,15,3,0,0,0                :rem 103
1040 DATA12,11,156,240,112,32,0,0            :rem 99
1050 DATA192,96,24,199,96,24,7,0            :rem 100
1060 DATA0,0,0,195,60,0,195,60              :rem 223
1070 DATA3,6,24,227,6,24,224,0              :rem 228
1080 DATA0,34,119,255,255,255,255,255        :rem 85
1090 DATA255,255,255,255,255,255,255,255    :rem 248
1100 DATA31,127,63,127,255,127,63,31         :rem 20
1110 DATA240,252,224,192,240,248,224,192    :rem 220
1120 DATA24,60,126,255,126,36,0,0           :rem 117
```

Program 2. Hang Glider, 64 Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
1 POKE53281,1:POKE53280,14:PRINT"{CLR}{BLU}
  {9 DOWN}";TAB(16);"GLIDER"                :rem 18
2 PRINT"{2 DOWN}";TAB(9)"{RVS}REDEFINING CHARACTER
  S{OFF}"                                   :rem 100
14 POKE52,48:POKE56,48:CLR                   :rem 26
16 POKE56334,PEEK(56334)AND254              :rem 176
18 POKE1,PEEK(1)AND251                        :rem 7
20 FORI=0TO1095:POKEI+12288,PEEK(I+53248):NEXT
                                            :rem 232
22 POKE1,PEEK(1)OR4                         :rem 108
24 POKE56334,PEEK(56334)OR1                  :rem 19
50 FORI=12568TO12671:READJ:POKEI,J:NEXTI      :rem 6
60 POKE53272,(PEEK(53272)AND240)OR29          :rem 3
63 PRINT"{CLR}":POKE53281,0:POKE53280,0:FORT=1TO20
   00:NEXT                                  :rem 29
65 DIMB(2,2):A=4                            :rem 84
70 POKE53280,3:POKE53281,1:SP=1:X=16:Y=1:X1=+2:Y1=
   1:PL=32:CO=54272:NI=1:FS=240             :rem 59
80 CV=54272:FORGL=CV TO CV+24:POKE GL,0:NEXT
                                            :rem 106
85 POKECV+24,15:POKECV+19,72:POKECV+20,129:POKECV+
   14,50                                    :rem 254
90 GOSUB800:GOTO500                         :rem 137
100 POKE56579,127:D=15-(PEEK(56320)AND15)   :rem 228
110 IFD=0ANDX<17THENY1=1                    :rem 200
115 IFD=0ANDPEEK(L+40)=44THENX1=0:Y1=0      :rem 246
```

```
120 IFD=1THENY1=-2:SP=SP-1:FS=FS+1          :rem 73
130 IFD=8THENX1=2:Y1=1:SP=SP+1:FS=246      :rem 250
140 IFD=4THENX1=-2:Y1=1:SP=SP+1:FS=246      :rem 36
150 IFD=2THENY1=2:SP=SP+1:FS=FS+1:RETURN    :rem 56
151 REM ***SP>3ALLOWED ABOVE***              :rem 0
160 IFSP>3THENSP=3                         :rem 124
165 IFSP<0THENY1=2:SP=0                    :rem 172
170 RETURN                                 :rem 120
200 IFNI=1THENNI=2:GOTO205                 :rem 101
202 IFNI=2THENNI=1                          :rem 93
205 I=NI                                   :rem 180
210 IFB(I,0)>0THEN250                      :rem 142
220 B(I,1)=INT(RND(1)*10)+5                :rem 151
225 IFPEEK(B(I,1)*40+1028)<>32THEN220      :rem 253
230 B(I,0)=6                                :rem 64
240 B(I,2)=1                                :rem 62
250 BL=1024+B(I,0)+B(I,1)*40               :rem 126
255 POKEBL,32:POKEBL-1,32:POKEBL-2,32:POKEBL-41,32
                                           :rem 209
260 B(I,0)=B(I,0)+1:IFRND(1)>.9THENB(I,1)=B(I,1)+1
                                           :rem 175
265 BL=1024+B(I,0)+B(I,1)*40               :rem 132
270 IFPEEK(BL)<>32ANDPEEK(BL)<>35THENB(I,0)=0:GOTO
    295                                    :rem 171
275 IFPEEK(BL)=35ORPEEK(BL-1)=35ORPEEK(BL-2)=35THE
    N400                                   :rem 234
280 IFPEEK(BL-40)=35ANDB(I,2)=1THENSCO=SCO+100:B(I
    ,2)=0:GOSUB300                         :rem 254
285 POKEBL,39:POKEBL+CO,5:POKEBL-1,37:POKEBL-1+CO,
    5:POKEBL-2,38:POKEBL-2+CO,5            :rem 200
290 IFB(I,2)=1THENPOKEBL-41,36:POKEBL-41+CO,2
                                           :rem 137
295 IFRND(1)>.9THENGOSUB370                :rem 131
299 RETURN                                 :rem 132
300 POKECV+18,33                           :rem 129
305 POKECV+15,45:POKECV+18,32              :rem 173
310 FORT=1TO150:NEXT                       :rem 239
315 POKECV+15,69:POKECV+18,17              :rem 183
320 FORT=1TO250:NEXT                       :rem 241
325 POKECV+18,16                           :rem 137
330 PL=32:RETURN                           :rem 238
370 POKECV+18,17                           :rem 138
375 FORT=44TO83:POKECV+15,T:NEXT            :rem 25
380 FORT=84TO45STEP-1:POKECV+15,T:NEXT     :rem 177
385 POKECV+18,16                           :rem 143
390 POKECV+18,8:RETURN                     :rem 118
400 REM DIE                                 :rem 74
410 GOSUB370                               :rem 175
420 A=A-1:IFA<-1THEN460                     :rem 32
```

```
450 GOTO70                                          :rem 57
460 PRINT"{CLR}{8 DOWN}{6 RIGHT}PLAY AGAIN?":rem 9
470 GETA$:IFA$=""THEN470                            :rem 89
480 IFA$<>"Y"THENPOKE53272,21:POKECV+15,0:END
                                                    :rem 151
490 CLR:GOTO65                                      :rem 92
500 GOSUB200                                        :rem 167
501 GOSUB100                                        :rem 167
502 IFFS>240ANDD<>152ANDD<>148THENFS=FS-1           :rem 230
510 POKEL,PL                                        :rem 217
520 X=X+X1:Y=Y+Y1                                   :rem 22
530 IFX>39THENX=X-40                                :rem 214
535 IFX<0THENX=40+X                                 :rem 155
540 IFY>24THEN400                                   :rem 232
545 IFY<1THENY=1                                    :rem 233
550 L1=1024+X+Y*40                                  :rem 176
555 PL=PEEK(L1)                                     :rem 107
560 IFPL>39ANDPL<43ANDD=0THEN700                    :rem 204
570 IFPL=44THEN400                                  :rem 47
580 IFPL>36ANDPL<40THEN400                          :rem 65
590 IFPL<>36THEN670                                 :rem 120
600 SCO=SCO+100:GOSUB300                            :rem 166
610 FORI=1TO2:IFL1=1001+(B(I,0)+B(I,1)*40)THENB(I,
    2)=0                                            :rem 203
620 NEXTI                                           :rem 32
670 POKEL1,35:POKEL1+54272,6                        :rem 84
675 L=L1                                            :rem 168
679 PRINT"{HOME}{BLK}{5 RIGHT}";SCO                 :rem 203
680 GOTO500                                         :rem 108
700 L1=L1-40:X1=0:Y1=0                              :rem 192
710 IFY>0THENY=Y-1                                  :rem 106
720 IFL1<1064THENL1=L1+40                           :rem 161
730 PL=PEEK(L1):IFPL=35THENPL=PEEK(L1+40)           :rem 250
740 POKEL1,35:L=L1                                  :rem 27
750 GOTO680                                         :rem 115
800 L=1100:PRINT"{CLR}"                             :rem 129
805 FORI=1224TO1984STEP40:FORJ=0TO2                 :rem 153
810 POKEI+J,44:POKEI+1+J,44:POKEI+J+54272,0:POKEI+
    J+54273,0:NEXTJ,I                               :rem 206
820 FORI=1988TO2023:POKEI,43:POKEI+54272,6:NEXTI
                                                    :rem 149
830 FORI=1832TO1992STEP40:FORJ=0TO2:POKEI+J,44:POK
    EI+J+54272,0:NEXTJ,I                            :rem 64
840 FORI=1979TO1099STEP-40:POKEI,40:POKEI+1,41:POK
    EI+2,42                                         :rem 168
845 FORJ=0TO2:POKEJ+I+54272,6:NEXTJ,I               :rem 4
850 FORI=1035TO1035+A:POKEI,47:POKEI+54272,0:NEXT:
    IFA=-1THENPOKE1035,32                           :rem 7
860 FORI=55296TO55335:POKEI,6:NEXT                  :rem 75
```

```
870 POKE1039,45:POKE1063,46:FORI=1040TO1061:POKEI,
    44:NEXT                                    :rem 143
880 PRINT"{HOME}{BLK}SCORE:"                   :rem 202
885 POKE54296,15                               :rem 112
890 RETURN                                     :rem 129
1000 DATA0,0,102,153,0,24,24,0                 :rem 199
1010 DATA56,60,24,16,24,20,18,16                :rem 70
1020 DATA60,126,255,255,222,56,112,192         :rem 121
1030 DATA31,31,31,15,3,0,0,0                   :rem 103
1040 DATA12,11,156,240,112,32,0,0               :rem 99
1050 DATA192,96,24,199,96,24,7,0               :rem 100
1060 DATA0,0,0,195,60,0,195,60                 :rem 223
1070 DATA3,6,24,227,6,24,224,0                 :rem 228
1080 DATA0,34,119,255,255,255,255,255           :rem 85
1090 DATA255,255,255,255,255,255,255,255       :rem 248
1100 DATA31,127,63,127,255,127,63,31            :rem 20
1110 DATA240,252,224,192,240,248,224,192       :rem 220
1120 DATA24,60,126,255,126,36,0,0              :rem 117
```

Reaction

Jeff Sprague
64 Version Translated by Gregg Peele

"Reaction" is an exciting arcade-style game that will sharpen your reflexes and give you hours of challenging fun. Written for the unexpanded VIC or the 64, it is a good example of what a short program can do.

Are you tired of games where your sole duty is to manipulate a passive dot eater around the screen? If so, "Reaction" is the game for you. It lets you move a dot eater, all right — but this dot eater has a mind of its own.

Game play is straightforward. Use your joystick to control a large white brick as it races around the screen. The object is to run over each of the hollow squares while avoiding those that are solid.

It sounds simple, but there are a couple of catches. Wherever the white brick goes, it leaves a solid wall behind it. You cannot run into one of those walls or you will be destroyed. In addition, once the white brick has started moving, it cannot be stopped. You can only control its direction, using your joystick.

When you run the program, you will be asked to select the desired difficulty level (1-20). Level 1 puts one solid and one hollow square on the screen, level 2 yields two solid and two hollow squares, and so on. For free practice rounds, add 100 to the desired level. For instance, if you wanted a practice round with five hollow bricks, you would enter 105 when the prompt asks for difficulty level.

This program, which uses a machine language routine to read the joystick, is extremely fast. As a result, most players will find it very hard to clear level 5. You can slow it down by adding the following line:

160 FOR I = 1 TO H:NEXT

Let H be any number from 2 to 100, depending on how much you want to slow things down. High numbers produce the greatest decrease in speed.

Program 1. Reaction, VIC Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 PRINT"{CLR}{WHT}":POKE36879,14:POKE36878,15:POK
   E36869,255:POKE56,28:GOSUB1000              :rem 6
99 REM LINE 100-200 ARE THE MAIN LOOP***   :rem 100
```

```
100 POKEX,A:SYSJOY:OM=M:IFD(PEEK(E)-B)=.THENM=OM:G
    OTO120                                    :rem 94
110 M=D(PEEK(E)-B)                           :rem 219
120 X=X+M:IFX<CORX>DTHENX=X-M:M=.            :rem 155
130 P=PEEK(X):IF(P=AANDM<>.)ORP=ETHEN500     :rem 214
135 POKEG,.                                  :rem 105
140 IFP=.THENDE=DE+E:POKEG,250:IFDE=STHEN400
                                             :rem 229
150 IFP=FTHEN300                             :rem 186
200 GOTO 100                                  :rem 92
300 REM HIT PRIZE                            :rem 230
400 REM WIN ROUTINE***                        :rem 10
410 GOSUB610                                 :rem 172
420 FORI=160TO220STEP5:FORJ=ITOI+30:POKEG,J:NEXT:N
    EXT:POKEG,.                              :rem 207
430 PRINT"{2 DOWN}{WHT}YOU WON!":PRINT"{DOWN}{RED}
    PREPARE FOR LEVEL ";S+1:S=S+1            :rem 170
440 FORI=1TO3000:NEXT:GOSUB1110:GOTO100      :rem 150
500 REM LOSE ROUTINE**                        :rem 38
510 GOSUB610                                 :rem 173
520 FORI=220TO127STEP-.5:POKEG,I:NEXT         :rem 63
530 PRINT"{DOWN}{WHT}YOU DIDN'T COMPLETE{3 SPACES}
    LEVEL ";S;"."                             :rem 34
535 PRINT"{DOWN}";DE/S*100;"%"                :rem 36
540 PRINT"{DOWN}PREPARE TO RETURN..."        :rem 151
550 FORI=1TO4000:NEXT:GOSUB1110:GOTO100      :rem 153
610 PRINT"{CLR}{WHT}":POKE36869,240:PRINT"
    {6 SPACES}REACTION"                      :rem 155
620 REM                                      :rem 124
630 RETURN                                   :rem 121
1000 REM GAME SET-UP**                       :rem 209
1005 DATA169,127,141,34,145,173,32,145,41,128,133,
     0,173,31,145,41,28,133,1,169,255        :rem 238
1010 DATA 141,34,145,165,0,74,74,74,74,133,0,165,1
     ,74,74,101,0,133,1,96                   :rem 146
1020 FORI=840TO880:READP:POKEI,P:NEXT:REM READ JOY
     STICK ROUTINE***                        :rem 170
1030 DATA0,0,1,0,0,0,-1,0,22,-22,0           :rem 106
1040 FORI=0TO10:READD(I):NEXT                 :rem 19
1050 X=7932:A=3:B=5:C=7724:D=8185:JOY=840:E=1:F=2:
     G=36876                                 :rem 213
1052 FORI=7192TO7199:POKEI,255:NEXT          :rem 120
1054 DATA0,60,36,36,36,60,0,0,0,60,60,60,60,60,0,0
                                             :rem 157
1056 FORI=7168TO7183:READP:POKEI,P:NEXT      :rem 210
1058 FORI=7424TO7431:POKEI,.:NEXT              :rem 3
1060 PRINT"{HOME}"                           :rem 171
1070 PRINT"{2 DOWN}{GRN}{RVS}USE THE JOYSTICK TO
     {4 SPACES}MOVE AROUND THE PLAY- FIELD."
                                             :rem 79
```

```
1080 PRINT"{CYN}{DOWN}{RVS}HIT THE {OFF}@{RVS}'S W
     HILE TRY-ING TO AVOID THE {OFF}A{RVS}'S."
                                          :rem 135
1090 PRINT"{DOWN}{YEL}{RVS}DIFFICULTY (1-20)";:INP
     UTS                                  :rem 184
1100 IFS<1ORS>20THENE=32:S=S-100          :rem 150
1110 PRINT"{CLR}":POKE36869,255:DE=.:OM=.:M=.
                                          :rem 201
1120 FORI=1TOS:POKEINT(460*RND(1)+C),E:NEXT:FORI=1
     TOS:POKEINT(460*RND(1)+C),.:NEXT:E=1  :rem 74
1130 PRINT"{HOME}{RVS}{GRN}REACTION"       :rem 46
1140 IFPEEK(A)=.THENA=A+1:GOTO1140        :rem 250
1200 RETURN                               :rem 163
```

Program 2. Reaction, 64 Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
5 POKE53281,0:POKE53280,12                :rem 189
10 PRINT"{CLR}{WHT}":POKE56,48:PRINT"{8 DOWN}
   {10 RIGHT}ENTERING CHARACTERS":GOSUB1000
                                          :rem 162
99 REM LINE 100-200 ARE THE MAIN LOOP***  :rem 100
100 POKE53272,29:POKEX+54272,1:POKEX,A:OM=M:IF41-P
    EEK(2)=.THENM=OM:GOTO120              :rem 254
110 M=41-PEEK(2)                           :rem 86
120 X=X+M:IFX<CORX>DTHENX=X-M:M=.         :rem 155
130 P=PEEK(X):IF(P=AANDM<>.)ORP=ETHEN500  :rem 214
135 POKEG,.                               :rem 105
140 IFP=.THENDE=DE+E:POKEG,250:IFDE=STHEN400
                                          :rem 229
150 IFP=FTHEN300                          :rem 186
200 GOTO 100                               :rem 92
300 REM HIT PRIZE                         :rem 230
400 REM WIN ROUTINE***                     :rem 10
410 GOSUB610                              :rem 172
420 FORI=160TO220STEP5:FORJ=ITOI+30:POKEG,J:NEXT:N
    EXT:POKEG,.:POKE53272,21              :rem 202
430 PRINT"{2 DOWN}{WHT}{5 DOWN}{15 RIGHT}YOU WON!"
                                          :rem 169
435 PRINT"{DOWN}{WHT}{3 DOWN}{10 RIGHT}PREPARE FOR
     LEVEL ";S+1:S=S+1                    :rem 169
440 FORI=1TO3000:NEXT:POKE53272,29:GOSUB1110:GOTO1
    00                                    :rem 153
500 REM LOSE ROUTINE**                     :rem 38
510 GOSUB610                              :rem 173
520 FORI=220TO127STEP-.5:POKEG,I:NEXT      :rem 63
530 PRINT"{DOWN}{WHT}{6 DOWN}{5 RIGHT}YOU DIDN'T C
    OMPLETE LEVEL ";S;"{LEFT}."           :rem 182
```

```
535 PRINT"{DOWN}{12 RIGHT}PERCENTAGE";INT(DE/S*100
    );"%"                                    :rem 154
540 PRINT"{DOWN}{10 RIGHT}PREPARE TO RETURN..."
                                             :rem 185
550 FORI=1TO4000:NEXT:POKE53272,29:GOSUB1110:GOTO1
    00                                       :rem 156
610 PRINT"{CLR}{WHT}":POKE53272,21:PRINT"{5 DOWN}
    {15 SPACES}REACTION"                     :rem 176
620 REM                                      :rem 124
630 RETURN                                   :rem 121
1000 REM GAME SET-UP**                       :rem 209
1010 POKE56334,PEEK(56334)AND254              :rem 11
1015 POKE1,PEEK(1)AND251                     :rem 101
1020 FOR I= 1024 TO 1536:POKEI+12288,PEEK(I+53248)
     :NEXT                                   :rem 224
1025 POKE1,PEEK(1)OR4:POKE56334,PEEK(56334)OR1
                                             :rem 183
1050 X=1482:A=3:B=5:C=1064:D=2023:E=1:F=2    :rem 230
1052 FORI=12288+24TO12288+31:POKEI,255:NEXT
                                             :rem 245
1054 DATA0,60,36,36,36,60,0,0,0,60,60,60,60,60,0,0
                                             :rem 157
1056 FORI=12288TO12288+15:READP:POKEI,P:NEXT
                                             :rem 196
1058 FORI=12288+256TO12288+256+7:POKEI,.:NEXT:GOSU
     B 1300:SYS49152                          :rem 27
1060 PRINT"{CLR}":POKE53272,29                :rem 46
1070 PRINT"{5 DOWN}{PUR}{RVS}{11 RIGHT}USE THE JOY
     STICK TO"                               :rem 105
1075 PRINT"{2 DOWN}{PUR}{6 RIGHT}{RVS}MOVE AROUND
     {SPACE}THE PLAY- FIELD."                :rem 242
1080 PRINT"{GRN}{2 DOWN}{RVS}{11 RIGHT}HIT THE
     {OFF}@{RVS}'S WHILE"                    :rem 200
1085 PRINT"{2 DOWN}{9 RIGHT}{RVS}TRYING TO AVOID T
     HE {OFF}A{RVS}'S."                       :rem 57
1090 PRINT"{10 RIGHT}{DOWN}{YEL}{RVS}DIFFICULTY (1
     -20)";:INPUTS                           :rem 218
1100 IFS<1ORS>20THENE=32:S=S-100             :rem 150
1110 PRINT"{CLR}":DE=.:OM=.:M=.              :rem 136
1120 FORI=1TOS:Q=INT(959*RND(1)+C):POKEQ+54272,1:P
     OKEQ,E:NEXT                             :rem 235
1125 FORI=1TOS:Q=INT(959*RND(1)+C):POKEQ+54272,1:P
     OKEQ,.:NEXT:E=1                         :rem 198
1130 PRINT"{HOME}{RVS}{GRN}REACTION"          :rem 46
1140 IFPEEK(A)=.THENA=A+1:GOTO1140           :rem 250
1200 RETURN                                  :rem 163
1300 I=49152                                  :rem 79
1310 READ B:IF B=256 THEN RETURN              :rem 23
1320 POKE I,B:I=I+1:GOTO 1310                 :rem 73
```

```
1330 DATA 120,169,13,141,20,3,169          :rem 75
1340 DATA 192,141,21,3,88,96,173           :rem 44
1350 DATA 0,220,41,15,73,15,168           :rem 233
1360 DATA 185,29,192,133,2,76,49           :rem 52
1370 DATA 234,41,81,1,41,42,41            :rem 181
1380 DATA 41,41,40,256                     :rem 53
```

Text Adventure Basics

B.A. Miller

Text adventures let you combine computer and imagination to create captivating role-playing games. This article presents a simple text adventure called "Old West" and describes techniques that you can use to write adventure games of your own. It will run on the VIC (with at least 8K expansion) or on the 64.

You've ridden hard all day, returning from your mine, and all you have to show for it is an empty wallet and a golden tan. You're tired, broke, and discouraged.

And then you come to the town.

It's not much to look at, even as ghost towns go, but a poster nailed to the post office door catches your eye. You step closer to read it:

"WANTED ALIVE — BLACK BART! $1000 REWARD"

A thousand dollars, eh? That's better than you did with the gold. But where could this Black Bart be? There's no one else around.

And then you hear a horse. You have a sudden feeling: An adventure is about to begin.

Adventure games can be among the most exciting computer games you'll find. A typical adventure will create a world, give you the rules by which it works, and then set you on your own to solve a mystery or find a treasure using common sense and your own ingenuity.

"Old West" places you in a seemingly deserted western town and offers a $1000 reward if you can find (and capture alive) the elusive Black Bart. You suspect that Bart is hiding somewhere in the town, but you'll have to use your wits to locate and capture him.

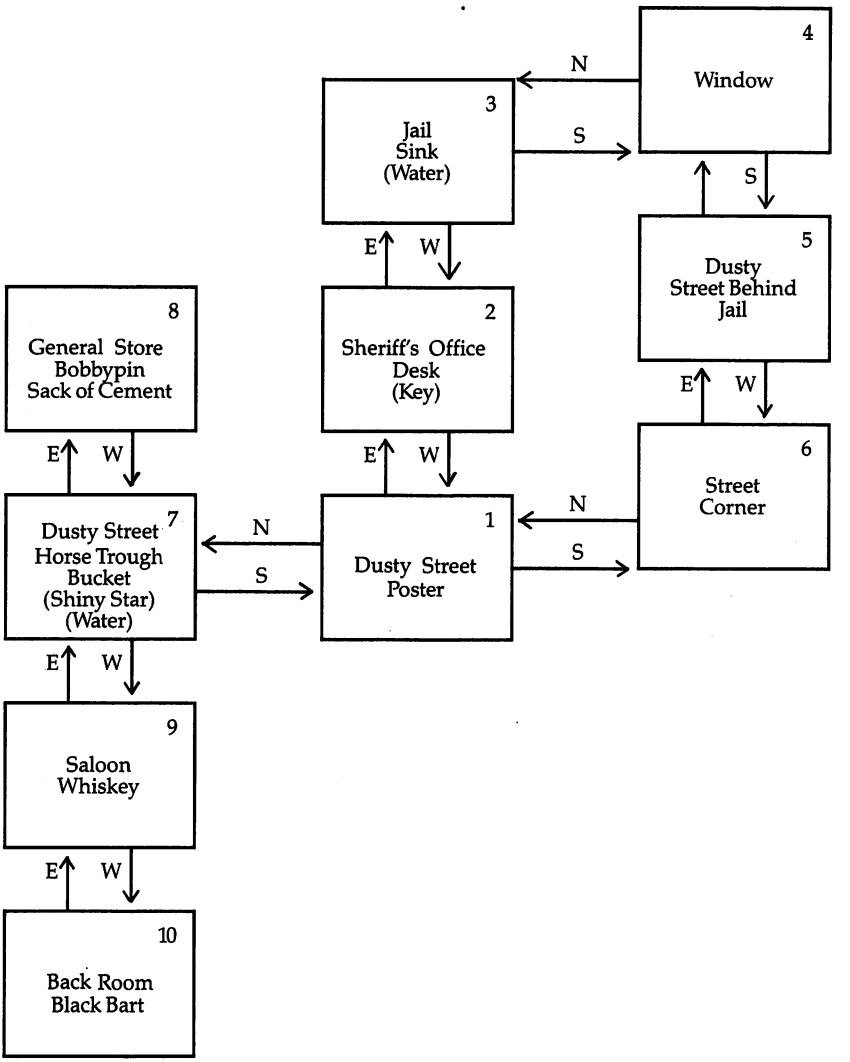
But Old West is more than a western version of hide and seek. It's an example of how an adventure game is written, and by studying the program you'll learn what you need to write adventure games of your own.

Creating Your Own Adventure Game

The following paragraphs outline basic steps taken to develop a game like Old West. They will give you a quick overview of adventure programming techniques, and by reading this discussion and studying the programs, you'll be well on your way to creating adventure games of your own.

Make a map. The first step in designing an adventure game is to map out the complete adventure. Then assign numbers to each room, list the objects that can be found in each room, and note from what directions you may enter and leave each room. Since Old West is a simple adventure, it has only ten rooms. In addition, there are only four possible directions of movement (north, south, east, and west). A complete map is shown in Figure 1.

Figure 1. Room Map



Construct a move matrix. The move matrix shows how each room connects with the others. The matrix for Old West is shown in Figure 2. Starting points are listed along the left, and directions are listed across the top. For example, if you start at room 1 (row 1) and want to move north (column 1), you can look at the row 1-column 1 intersection to see where you will end up. In this case, you would be in room 7.

Figure 2. Move Matrix

		Direction of Movement			
		N (1)	S (2)	E (3)	W (4)
Room	1	7	6	2	0
	2	0	0	(3)	1
	3	0	(4)	0	(2)
	4	3	5	0	0
	5	(4)	0	0	6
	6	1	0	5	0
	7	0	1	8	9
	8	0	0	0	7
	9	0	0	7	(10)
	10	0	0	9	0

Entries can be thought of as having the form M(a,b)=c, where a is the room where you start, b is the direction of movement, and c is the room where you end up. The entry M(1,2)=6 means that if you start at room 1 (dusty street) and GO SOUTH (direction 2), you will end up in room 6 (street corner). Similarly, the entry M(6,3)=5 means that if you GO EAST from room 6, you will land in room 5.

Notice that the move matrix contains zeros, nonzero numbers, and nonzero numbers in parentheses. Zeros indicate moves that cannot be made, perhaps because the rooms do not connect. Nonzero numbers represent moves that can be made at any time. Nonzero numbers in parentheses are set to zero at the start of the game but represent conditional moves that can eventually be made. For example, M(2,3) (moving from the sheriff's office to the jail) is initially set to zero, since you have not yet found the key to the jail. However, once the key has been found and the jail has been unlocked, M(2,3) is set to 3 (so you can move from the sheriff's office to the jail), and M(3,4) is set to 2 (so you can go from the jail to the office).

The (4) at M(3,2) and M(5,1) means that you can get to room 4, the window, but not by going N, S, E, or W. The command GO WINDOW gets you in and out of the window. Similarly, GO ROOM allows you to pass from room 9 (the saloon) to room 10 (the room in back). Program lines 650-680 read in the room descriptions, and lines 620-640 read in the move matrix.

Describe objects in rooms. The next step is to describe each object and to note where it is located. Be careful to match everything up! I use the common technique of assigning a two-letter abbreviation to each object. Lines 500-512 describe each object, line 570 stores the abbreviation for each in the long string N$, and lines 600-610 record the room where each object originally appears. For example, the wanted poster (PO) is in room 1 and the desk (DE) is in room 2. Hidden objects like the key are given room number 50, which doesn't exist, so they will not appear until the room number is changed. When the desk drawer is opened, for instance, the value for the key L(17) will be set to 2, making it appear in the office.

To make an object disappear (for example, to DRINK something), L(object) is set to 50. To carry an object, L(object) becomes zero. To drop an object, L(N) = L. N is the noun representing the object, and L is always the current room location. By clustering movable objects together (8 N 19 if movable), one IF statement can decide if the player can take that object.

Select action verbs. Action verbs allow the player to take action within the game. In Old West, allowable action verbs are listed in V$ (line 560) by their first two letters. GO, TAKE, GET, PICK, OPEN, and PUT are some allowable verbs. Each is matched with a GOTO in line 300. An input string (OPEN JAIL, for instance) is decoded by subroutine 1000-1045 into A$ = OP and B$ = JA. Lines 150-170 then convert A$ into a value V = 5 and B$ into N = 6, the relative positions of OP and JA in V$ and N$. Line 300 uses V = 5 to GOTO 480, the OPEN routine.

Establish switches. To tell if certain conditions have occurred, switches C1 (mixed cement), K1 (found key), W1 (have water), and S1 (sleeping Black Bart) are used. The remainder of the program is made up of IF statements, which compare and act on the conditions previously established.

Old West for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 PRINTCHR$(147):C1=0:K1=0:L1=0:B1=0:W1=0:J1=0:S
    1=0                                          :rem 4
110 DIM N$(19),L(26),M(10,4),R$(10)             :rem 169
120 GOSUB 500:L=1:PRINT"I HEAR A RASPING NOISE":FO
    RTD=1 TO 1000:NEXT TD                       :rem 127
125 PRINT"I HEAR A HORSE!":FOR TD=1TO300:NEXTTD
                                                :rem 35
126 PRINT:FORCY=1TO10:PRINT"{2 SPACES}CLOPPITY":FO
    RTD=1TO200:NEXTTD:NEXTCY                    :rem 86
127 FOR TD=1TO1000:NEXT TD:PRINT                :rem 195
130 GOSUB 700                                   :rem 171
135 IFM(4,2)<>0ORL(13)<>3THEN140                 :rem 21
136 PRINT"{CLR}":PRINT"YOU CAUGHT BLACK BART!"
                                                :rem 105
137 PRINT"THE REWARD IS YOURS!"                 :rem 113
138 END                                         :rem 115
140 PRINT:PRINT"TELL ME WHAT TO DO":GOSUB 1000
                                                :rem 213
145 IF S=1THENS=0:GOTO140                       :rem 225
150 FOR I = 1 TO LEN(V$)STEP2                   :rem 239
160 IF MID$(V$,I,2)=A$THENV=(I+1)/2:GOTO200
                                                :rem 130
170 NEXTI                                        :rem 32
180 PRINT"I DON'T KNOW HOW TO DO THAT":GOTO140
                                                :rem 171
200 IF B$=""THEN 300                            :rem 202
210 FOR I=1TOLEN(N$)STEP2                       :rem 228
220 IF MID$(N$,I,2)<>B$THEN 230                  :rem 91
225 N=(I+1)/2                                   :rem 123
227 IFN=21THENN=9                                :rem 11
229 GOTO300                                     :rem 105
230 NEXTI                                        :rem 29
300 ON V GOTO 350,400,400,450,480,800,800,830,130,
    800,900,980,950                             :rem 147
309 REM UNLOCK DRAWER{2 SPACES}DESK OR CELL
                                                :rem 249
310 IF N=6ANDL(17)=0ANDL=2THENJ1=1:M(2,3)=3:GOSUB1
    150:M(3,4)=2:GOTO130                         :rem 10
320 PRINT"CAN'T DO THAT YET"                    :rem 105
330 GOTO130                                      :rem 99
350 IFN<>7 ANDN< 22 THEN380                     :rem 177
355 IF N=7 ANDM(4,2)<>0AND(L=5ORL=3)THENL=4:GOTO13
    0                                           :rem 205
360 IFM(L,N-21)=0THEN395                         :rem 89
370 L=M(L,N-21):GOTO130                          :rem 30
380 IF N<>4 OR L<>7THEN 385                     :rem 147
```

```
381 PRINT"SPLASH--I'M ALL WET":L(18)=7     :rem 204
382 IFL(11)=5ØTHENPRINT"I FOUND SOMETHING":L(11)=7
    :GOTO13Ø                               :rem 244
385 IFN=5 ANDL=9THENL=1Ø:GOTO13Ø           :rem 164
395 PRINT"CAN'T GO THAT WAY":GOTO13Ø       :rem 126
4ØØ IFC>=4THENPRINT"GO TOO MUCH--TAKE INVENTORY":G
    OTO14Ø                                 :rem 31
4Ø5 IFN=14THEN425                          :rem 226
41Ø IFL(N)<>LTHENPRINT"I DON'T SEE IT HERE":GOTO14
    Ø                                      :rem 32
42Ø IFN<8ORN>19THENPRINT"CAN'T TAKE THAT":GOTO14Ø
                                           :rem 41
425 IFN=14ANDL<>2ANDL(16)<>Ø THENPRINT"CAN'T DO TH
    AT YET":GOTO14Ø                        :rem 113
43Ø IFN=14ANDL=2THENN$(3)="SINK FULL OF WATER":W1=
    1:GOTO13Ø                               :rem 1
435 IFN=14ANDL(16)=ØTHENN$(16)="BUCKET OF WATER":W
    1=1:GOTO13Ø                            :rem 7Ø
44Ø IF N=13AND S1=1 THEN 444               :rem 166
441 IFN<>13THEN444                          :rem 31
442 PRINT"BLACK BART GETS MAD!":PRINT"HE DRAWS HIS
     REVOLVER!"                            :rem 17Ø
443 PRINT"THAT'S THE END":END              :rem 224
444 IFN=13ANDL(11)<>ØTHENPRINT"CAN'T DO THAT YET":
    GOTO14Ø                                :rem 161
445 PRINT"OKAY":C=C+1:L(N)=Ø:GOTO14Ø       :rem 149
45Ø IFL1=1THENPRINT"IT'S NOT LOCKED":GOTO14ØRUN
                                           :rem 2Ø2
46Ø IFL=2ANDL(9)=ØANDL1=ØAND(N=2ORN=2ØORN=26)THEN4
    65                                     :rem 136
462 GOTO47Ø                                :rem 112
465 L1=1:K1=1:PRINT"OKAY":N$(2)="UNLOCKED DESK"
                                            :rem 25
47Ø GOTO14Ø                                :rem 1Ø5
48Ø IFN<>2ANDN<>2ØANDN<>26THENPRINT"CAN'T":GOTO14Ø
                                           :rem 122
49Ø IFK1=1ANDL=2ANDL(17)=5ØTHENL(17)=2:GOTO495
                                           :rem 214
492 PRINT"CAN'T":GOTO14Ø                   :rem 197
495 PRINT"OKAY--THERE'S SOMETHING IN THERE":GOTO14
    Ø                                      :rem 184
5ØØ FORI=1TO19:READN$(I):NEXTI             :rem 1ØØ
51Ø DATAPOSTER,DESK WITH LOCKED DRAWER,SINK,HORSE
    {SPACE}TROUGH,ROOM IN BACK             :rem 139
511 DATA JAIL CELL-LOCKED,EMPTY WINDOW,BARS, BOBBY
    PIN,SACK OF CEMENT                     :rem 215
512 DATA SHINY STAR,BOTTLE OF ELIXIR,BLACK BART, W
    ATER,WATER,BUCKET,KEY                  :rem 222
514 DATA WATER,EMPTY WINDOW                :rem 2Ø2
```

```
550 V1$="NSEW"                                    :rem 3
560 V$="GOTAGEPIOPPUDRMILOGISWFIREUN"            :rem 244
570 N$="PODESITRROJAWIBABOCESTWHBLWAWABUKEWAWIDRPI
    NOSOEAWELO"                                  :rem 239
600 FORI=1TO19:READL(I):NEXTI                     :rem 63
610 DATA 1,2,3,7,9,2,3,5,8,8,50,9,10,50,11,7,50,11
    ,5                                           :rem 210
620 FORI=1TO10:FORJ=1TO4:READM(I,J):NEXTJ:NEXTI
                                                 :rem 34
630 DATA 7,6,2,,,,,1,,,,0,3,5,,,0,,,6            :rem 197
640 DATA 1,,5,,,1,8,9,,,,7,,,7,,,,9,             :rem 167
650 FORI=1TO10:READR$(I):NEXTI                   :rem 101
660 DATA DUSTY STREET,OFFICE,JAIL,WINDOW,DUSTY STR
    EET BEHIND JAIL                              :rem 180
670 DATA STREET CORNER,DUSTY STREET,GENERAL STORE,
    SALOON,ROOM IN BACK                          :rem 179
675 R$(2)="SHERIFF'S "+R$(2)                     :rem 193
680 RETURN                                       :rem 126
700 PRINT"I AM IN A "R$(L)                        :rem 42
710 PRINT"I SEE ";                               :rem 202
720 FOR I = 1 TO 19                               :rem 68
730 IF L(I)=L THEN PRINTN$(I);" ";:S=1            :rem 21
740 NEXT I                                        :rem 35
750 IF S=0 THENPRINT"NOTHING SPECIAL"              :rem 3
760 IF S=1 THEN PRINT:S=0                        :rem 163
765 IF L=0THENRETURN                             :rem 249
770 PRINT"OBVIOUS EXITS ARE ";                    :rem 54
775 FORI=1TO4:IFM(L,I)=0THEN785                  :rem 128
780 PRINTMID$(V1$,I,1)+" ";                      :rem 162
785 NEXTI:PRINT                                  :rem 243
790 RETURN                                       :rem 128
800 IFN=10ORN=8THEN830                            :rem 65
805 IFN=12ANDS1=0THEN870                         :rem 172
810 IFL(N)=0THENL(N)=L:C=C-1:PRINT"OKAY":GOTO130
                                                 :rem 196
820 PRINT"I'M NOT CARRYING IT":GOTO140            :rem 29
830 PRINT"WHERE":INPUT C$                         :rem 24
840 IFLEFT$(C$,2)<>"SI"ANDLEFT$(C$,2)<>"WA"ANDLEFT
    $(C$,2)<>"BU"THEN 845                        :rem 213
842 IFW1=1ANDN=10ANDL(10)=0THEN850                :rem 236
843 GOTO847                                      :rem 123
845 IFLEFT$(C$,2)="WI"ANDL(8)=0ANDC1=1ANDL=4ANDN=8
    THEN860                                      :rem 115
847 PRINT"CAN'T DO THAT YET":GOTO140             :rem 127
850 PRINT"OKAY.{2 SPACES}IT'S MIXED":C1=1:L(10)=50
    :C=C-1                                       :rem 175
855 IFLEFT$(C$,2)="BU"THENN$(16)="BUCKET OF CEMENT
     MIXTURE":GOTO130                            :rem 204
856 N$(15)="CEMENT MIXTURE":L(15)=L:GOTO130 :rem 4
```

```
860 PRINT"OKAY":N$(7)="BARRED WINDOW":L=3:M(4,2)=0
    :L(8)=50:C=C-1: GOTO130                    :rem 192
870 IFL(12)<>0THEN847                          :rem 170
875 IFA$<>"GI"THEN900                          :rem 174
876 IFL<>10THEN847                              :rem 45
880 L(12)=50:PRINT"BLACK BART GULPS{6 SPACES}ELIXI
    R AND IMMEDIATELYFALLS ASLEEP"             :rem 94
890 S1=1:C=C-1:N$(13)="SLEEPING "+N$(13):GOTO140
                                               :rem 129
900 IFN=12ANDL(12)=0THENL(12)=50:PRINT"GLUG GLUG G
    LUG":C=C-1:GOTO140                         :rem 34
910 IFN=12 THENPRINT"CAN'T DO THAT YET":GOTO140
                                               :rem 34
920 PRINT"CAN'T DRINK THAT":GOTO140            :rem 106
950 IFN<>1ORL<>1THENPRINT"CAN'T":GOTO140        :rem 19
960 PRINT"WANTED ALIVE-BLACK BART! $1000 REWARD"
                                               :rem 34
970 GOTO140                                    :rem 110
980 IFN<>3ORL<>3THENPRINT"CAN'T":GOTO140        :rem 26
990 PRINT"HOW";:INPUTC$                        :rem 205
995 IFLEFT$(C$,2)="WA"THENL(14)=3:W1=1         :rem 133
996 GOTO130                                    :rem 117
1000 INPUT A$:B$=""                            :rem 215
1005 IF LEFT$(A$,2)="QU"THENEND                :rem 229
1010 IFLEN(A$)=1THEN1060                        :rem 74
1020 FORI=1TOLEN(A$)                           :rem 153
1030 IFMID$(A$,I,1)=" "AND LEN(A$)>I+1THENB$=MID$(
     A$,I+1,2):GOTO1045                        :rem 151
1040 NEXTI                                      :rem 77
1045 A$=LEFT$(A$,2):RETURN                     :rem 233
1060 IFA$<>"I"THEN1100                         :rem 179
1070 PRINT"I'M CARRYING ";:T=L:L=0:GOSUB720:rem 77
1080 L=T:S=1:RETURN                            :rem 187
1100 FORI=1TO4                                  :rem 55
1110 IFA$<>MID$(V1$,I,1)THEN1130               :rem 241
1120 A$="GO":B$=MID$(N$,2*I+41,2):RETURN       :rem 183
1130 NEXTI                                      :rem 77
1140 PRINT"CAN'T DO THAT":S=1:RETURN           :rem 189
1150 N$(6)="OPEN JAIL CELL":RETURN             :rem 205
```

Nim

George Trepal

"Nim" is a game that you cannot win — unless you're the computer! Use it to amaze and impress your friends; the program runs on either the VIC or the 64.

Suppose someone said, "Let's play a game. I have 21 objects here, and on each turn we can take away one, two, or three of them. The person left with the last object loses. Since I'm a good sport, I'll let you go first."

Reasonable enough, you'd decide, so you'd give it a try.

You would lose.

You would try again, and lose again, over and over until you finally got tired.

That's the game of "Nim."

If you know the secret, there is no way you can lose at Nim. The key is to let the other player go first (how thoughtful!) and then to take a number of objects that, added to the number taken by your opponent, equals four. Each complete turn will thus remove four objects. Note that 21 divided by 4 leaves a remainder of 1; by letting your opponent go first, you guarantee that he will be stuck with the remainder. You can use any number that leaves such a remainder when divided by four, and the results will always be the same.

The program is straightforward. Lines 120-190 contain the computer's responses, which are chosen randomly to add life to the game. Line 200 actually starts the game, by setting S equal to 21. To make things even more entertaining, you could modify the program to use a different S each time the game is played; simply generate a random number, multiply by four, and add one to get a suitable value.

Line 320 initiates play. Line 340 clears the keyboard buffer, and line 370 GETs N$, the number string that is your guess. It is not necessary to press return after typing in your guess. Line 380 checks for keyboard input. If nothing has been typed, it loops back and looks again until something does come in.

Line 400 checks to make sure that the number entered is 1, 2, or 3; line 420 makes sure that no decimal numbers have been entered. Once a number has been accepted, lines 440-460 display the guess and the number of objects remaining.

Lines 480-500 are included to make game play even more life-like. They introduce a variable delay which can last from less than

a second to several seconds, and they give the appearance that the computer is agonizing over its next choice. Most players interpret a short delay as "Gee, I made a dumb move, and the computer didn't even have to think." A long delay usually has the opposite effect, even though the delays are completely random.

Lines 510-600 generate the computer's next guess, display it on the screen, tell you how many objects the computer took, and remind you how many objects remain. If more than one object remains, line 630 returns you to line 320. If only one remains, line 610 initiates the YOU LOSE routine. The rest of the program resets things for another game.

Though this is a bare-bones sort of game, it definitely has the capacity to drive people right up the walls. Unleash it the next time someone tries to tell you that a computer is just a boxful of wires and solder.

Nim for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
120 REM COMPUTER RESPONSES                    :rem 168
130 W$(1)="I THINK I'LL TAKE"                  :rem 6
140 W$(2)="THIS TIME I WANT"                  :rem 254
150 W$(3)="I HAVE TAKEN"                      :rem 246
160 W$(4)="I'LL HAVE"                          :rem 68
170 W$(5)="I TOOK"                            :rem 160
180 W$(6)="LET ME HAVE"                       :rem 183
190 W$(7)="THIS TIME GIVE ME"                  :rem 66
200 S=21                                      :rem 133
210 PRINT"{CLR}LET'S PLAY THE GAME OF"         :rem 28
220 PRINT"NIM.{2 SPACES}WE START WITH 21"      :rem 64
230 PRINT"THINGS.{2 SPACES}EACH TURN WE"       :rem 87
240 PRINT"CAN TAKE AWAY ONE,TWO,"             :rem 196
250 PRINT"OR THREE THINGS."                   :rem 124
260 PRINT                                      :rem 37
270 PRINT"THE ONE WHO HAS TO"                 :rem 154
280 PRINT"TAKE THE LAST THING"                 :rem 31
290 PRINT"LOSES THE GAME."                     :rem 27
300 PRINT                                      :rem 32
310 PRINT"YOU GET TO GO FIRST!"                :rem 36
320 PRINT"HOW MANY DO YOU WANT?"              :rem 146
330 REM CLEAR KEYBOARD BUFFER                 :rem 236
340 POKE 198,0                                :rem 196
350 REM GET NUMBER & MAKE SURE                :rem 168
360 REM IT IS 1,2, OR 3                        :rem 69
370 GET N$                                    :rem 236
380 IF N$="" THEN 370                         :rem 230
390 N = ASC(N$) - 48                           :rem 90
```

```
400 IF N<1 OR N>3 THEN PRINT"NO! YOU CAN TAKE ONLY
     1,2, OR 3!":GOTO 370                    :rem 239
410 REM CHECK FOR DECIMAL NUMBERS            :rem 201
420 IF N <> INT(N) THEN PRINT "NO! USE ONLY 1, 2,
    {SPACE}OR 3!"                             :rem 20
430 PRINT                                     :rem 36
440 PRINT"YOU TOOK";N                         :rem 44
450 PRINT S;"-";N"=";S-N                      :rem 56
460 S=S-N                                    :rem 248
470 REM THINKING LOOP                         :rem 21
480 RN=INT(RND(1)*3500) + 300                :rem 203
490 FOR J= 1 TO RN                           :rem 127
500 NEXT J                                    :rem 30
510 IF N = 1 THEN R = 3                      :rem 210
520 IF N = 2 THEN R = 2                      :rem 211
530 IF N = 3 THEN R = 1                      :rem 212
540 PRINT                                     :rem 38
550 REM GET RESPONSE                         :rem 205
560 RN=INT(RND(1)*7) + 1                     :rem 215
570 PRINT W$(RN);R                            :rem 34
580 PRINT                                     :rem 42
590 PRINT S;"-";R;"=";S-R                    :rem 128
600 S=S-R                                    :rem 248
610 IFS=1 THEN 640                           :rem 176
620 PRINT                                     :rem 37
630 GOTO320                                  :rem 103
640 PRINT                                     :rem 39
650 PRINT"YOU HAVE TO TAKE THE"               :rem 54
660 PRINT"LAST ONE SO YOU LOSE!"             :rem 118
670 PRINT                                     :rem 42
680 PRINT"TO PLAY AGAIN PRESS"                :rem 53
690 PRINT"THE 'A' KEY."                      :rem 247
700 GET A$                                   :rem 220
710 IF A$ = "" THEN 700                      :rem 211
720 IF A$ <> "A" THEN 700                     :rem 82
730 GOTO 200                                 :rem 101
```

Save the King

Andy Hayes
64 Formatter by Charles Brannon

"Save the King" is a realtime adventure game that shows what you can do when you combine the challenge of a text adventure with the excitement of computer graphics. Versions are included for the VIC (with at least 3K expansion) and for the 64. A screen formatter is also included, to better utilize the 64's 40-column display.

It's Celebration Day, and the crowd is exuberant. Thousands fill the castle square, awaiting the State of the Kingdom address.

But where is the king? He should have arrived an hour ago. His speech is scheduled for six o'clock — just ten minutes away — no one knows what the holdup could be.

No one but you.

You're chief of castle security, and you've just gotten word that the king has been kidnapped. According to the ransom note, he's hidden somewhere in that abandoned castle just outside town. Members of a rival political faction are holding him prisoner — and according to the laws of the land, he must abdicate the throne if he doesn't appear promptly at the appointed hour.

Can you save the king in time? By yourself, you wouldn't have a chance. But as luck would have it, you just happen to have a computerized map of the very castle where the king is being held. If you can pull off the rescue, it will mean good times for the kingdom and rich rewards for yourself. And if you fail? Some things are better not talked about at all.

You've got ten minutes. Good luck!

"Save the King" is an exciting realtime adventure game that combines vivid text with three-dimensional graphics. Your computer not only tells you what you're seeing, but shows you each room too. You view each room from the south, and all doors are clearly visible. If there's a door behind you (on the south wall), the word SOUTH will appear on the right side of the screen.

Type this program carefully. Pay particular attention to the characters in brackets; if you need help, refer to Appendix B, "How to Type In Programs." Be careful when typing the DATA statements, too, so that you do not inadvertently leave out a word or a comma.

This program recognizes five commands: GO, GET, READ, USE, and I. Use GO in combination with a direction (GO EAST, GO WEST, and so on) to move through the castle's rooms. READ lets you read the written word, as in READ BOOK. GET allows you to pick up objects (GET KNIFE) that you might find along the way, and USE (USE KEY) helps you put those objects to work. Note that USE distinguishes between singular and plural nouns; for instance, you cannot USE KEYS if you have only found one key.

The final command, I, gives you an inventory of what you're carrying. You'll refer to it often. It is particularly useful as the game progresses, when you're more likely to forget what you've gathered.

To use the program on the 64, first type in and save the main listing. Then type in and save the following lines:

```
100 PRINT "{CLR}{4 SPACES}{RVS}22 COLUMN PRINT FOR
    MATTER FOR 64":PRINT                    :rem 146
110 PRINT "READING DATA"                    :rem 119
120 FORI=828TO881:READA:CK=CK+A:POKEI,A:NEXT:POKE1
    79,883 AND 255                           :rem 92
130 IF CK<> 6032 THEN PRINT "ERROR IN DATA: CHECK
    {SPACE}TYPING.":END                     :rem 227
140 PRINT"{DOWN}BEFORE....":SYS 828:PRINT"AFTER...
    "                                       :rem 196
150 PRINT "{DOWN}PRESS RUN/STOP-RESTORE";:PRINT"TO
     REGAIN 40 COLUMNS"                     :rem 228
160 PRINT "{DOWN}ENTER {RVS}SYS 828{OFF} TO":PRINT
    "REACTIVATE, IF":PRINT"NECESSARY."      :rem 115
170 PRINT "{DOWN}DO NOT EDIT ANY":PRINT"LINES WHIL
    E IN 22 COL-UMN MODE."                   :rem 84
1000 DATA169,71,141,38,3,169,3,141          :rem 180
1010 DATA39,3,96,72,152,72,138,72           :rem 141
1020 DATA56,32,240,255,192,9,176,3          :rem 185
1030 DATA76,100,3,192,31,144,15,169         :rem 226
1040 DATA13,32,202,241,56,32,240,255          :rem 9
1050 DATA160,9,24,32,240,255,104,170         :rem 14
1060 DATA104,168,104,76,202,241              :rem 30
```

Type RUN and hit RETURN. Finally, LOAD and RUN "Save the King."

As you explore the castle in search of the king, you might find it helpful to sketch a rough map on a piece of old parchment. Lacking parchment, note paper will do just fine. Just don't get so caught up in your artwork that you let time get away. Remember, you've only got ten minutes.

Save the King for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 PRINT"{CLR}":POKE53281,1:POKE53280,6  :rem 139
110 DIM GE$(5,3),D$(5,3),R$(5,3),W%(5,3),X%(5,3),Y
    %(5,3),Z%(5,3),E%(5,3)                :rem 182
120 GOSUB1190                             :rem 222
130 FORX=0TO5:FORY=0TO3:E%(X,Y)=1:NEXTY,X :rem 107
140 X=0:Y=0                               :rem 90
150 TI$="000000"                          :rem 248
160 IFA>2THEN660                          :rem 162
170 GOSUB930                              :rem 180
180 IFW%(X,Y)=1THENGOSUB1050              :rem 181
190 IFY%(X,Y)=1THENGOSUB1110              :rem 181
200 IFZ%(X,Y)=1THENGOSUB1150              :rem 178
210 IFX%(X,Y)=1THENPRINT"{HOME}{DOWN}{14 RIGHT}SOU
    TH"                                   :rem 136
220 IFG=1THENG=0:RETURN                   :rem 213
230 GOSUB880:PRINT"{PUR}"R$(X,Y)           :rem 0
240 PRINT"{BLK}"D$(X,Y)                   :rem 141
250 IFE%(X,Y)=1THENPRINT"{GRN}"GE$(X,Y)    :rem 40
260 INPUT"{BLU}YOUR COMMAND";C$           :rem 123
270 FORI=1TOLEN(C$)                       :rem 113
280 D$=MID$(C$,I,1)                       :rem 199
290 IFD$=" "THENA=I:GOTO310                :rem 16
300 NEXT                                  :rem 210
310 A$=LEFT$(C$,I)                        :rem 178
320 B$=MID$(C$,I):A=0                      :rem 75
330 IFA$="GO "THENA=1                     :rem 127
340 IFA$="GET "THENA=2                    :rem 203
350 IFA$="USE "THENA=3                    :rem 218
360 IFA$="READ "THENA=4                    :rem 11
370 IFA$="I"THENGOSUB800                  :rem 159
380 IFA=0THEN480                          :rem 163
390 IFA>1THEN460                          :rem 164
400 IFA=1THENIFB$=" NORTH"ANDW%(X,Y)=1THENX=X+1:GO
    TO170                                 :rem 112
410 IFA=1THENIFB$=" SOUTH"ANDX%(X,Y)=1THENX=X-1:GO
    TO170                                 :rem 124
420 IFA=1THENIFB$=" WEST"ANDY%(X,Y)=1THENY=Y+1:GOT
    O170                                   :rem 46
430 IFA=1THENIFB$=" EAST"ANDZ%(X,Y)=1THENY=Y-1:GOT
    O170                                   :rem 28
440 IFB$<>" WEST"ANDB$<>" EAST"ANDB$<>" SOUTH"ANDB
    $<>" NORTH"THEN840                    :rem 137
450 G=1:GOSUB170:GOSUB880:PRINT"HOW ARE GOING TO G
    ET{2 SPACES}THROUGH THE WALL?"         :rem 51
455 GOTO 500                              :rem 108
460 IFA>2THEN660                          :rem 165
470 IFA=2THENGOTO520                      :rem 217
```

```
480 G=1:GOSUB170:GOSUB880:PRINT"I DONT KNOW HOW TO
    "                                          :rem 86
490 PRINTA$"SOMETHING."                        :rem 175
500 FORT=1TO2500:NEXT                          :rem 33
510 GOTO170                                    :rem 103
520 IFB$=" KEY"ANDX=2ANDY=3ANDE%(X,Y)=1THENU=U+1:E
    $(U)="GOLD KEY":F=1:NK=NK+1                :rem 67
530 IFB$=" CHEESE"ANDX=2ANDY=1ANDE%(X,Y)=1THENU=U+
    1:E$(U)="PIECE OF CHEESE":F=1              :rem 154
540 IFB$=" CHALK"ANDX=4ANDY=1ANDE%(X,Y)=1THENU=U+1
    :E$(U)="PIECE OF CHALK":F=1                :rem 9
550 IFB$=" BOOK"ANDX=4ANDY=3ANDE%(X,Y)=1THENU=U+1:
    E$(U)="BOOK":F=1                           :rem 161
560 IFB$=" KNIFE"ANDX=1ANDY=1ANDE%(X,Y)=1THENU=U+1
    :E$(U)="KNIFE":F=1                         :rem 33
570 IFB$=" BEETLE"ANDX=5ANDY=3ANDE%(X,Y)=1THENU=U+
    1:E$(U)="BEETLE":F=1                       :rem 176
580 IFB$=" BAT"ANDX=3ANDY=1ANDE%(X,Y)=1THENU=U+1:E
    $(U)="BAT":F=1                             :rem 249
590 IFB$=" KEY"ANDX=3ANDY=0ANDE%(X,Y)=1THENU=U+1:E
    $(U)="SILVER KEY":F=1:NK=NK+1              :rem 247
600 IFF=1THENE$=E$(U):F=0:GOTO640              :rem 183
610 IFB$=" BAT"ORB$=" BEETLE"ORB$=" KNIFE"THEN870
                                               :rem 224
615 IFB$=" BOOK"ORB$=" CHALK"ORB$=" CHEESE"THEN870
                                               :rem 43
620 GOTO850                                    :rem 110
630 G=1:GOSUB170:GOSUB880:PRINT"YOU CAN'T DO THAT!
    ":GOTO500                                  :rem 60
640 G=1:GOSUB170:GOSUB880:PRINT"YOU HAVE A":E%(X,Y
    )=0                                        :rem 167
650 PRINTE$:GOTO500                            :rem 153
660 IFA>3THEN710                               :rem 164
670 IFA$="USE "ANDB$=" KEYS"ANDX=5ANDY=1THEN890
                                               :rem 252
680 IFA$="USE "ANDB$=" KEYS"ANDX<>5ANDY<>1THEN700
                                               :rem 109
690 G=1:GOSUB170:GOSUB880:PRINT"I DONT KNOW HOW TO
    {4 SPACES}"A$"A"B$:GOTO500                 :rem 177
700 G=1:GOSUB170:GOSUB880:PRINT"WHERE ARE THE LOCK
    S?":GOTO500                                :rem 250
710 FORI=1TO10                                 :rem 58
720 GOTO780                                    :rem 113
730 IFE$(I)="BOOK"THEN760                      :rem 164
740 NEXT                                       :rem 218
750 G=1:GOSUB170:GOSUB880:PRINT"YOU HAVE NO BOOK!"
    :GOTO500                                   :rem 26
760 PRINT"{CLR}FIND THE LOCK AND MAKESURE YOU HAVE
     BOTH OF THE KEYS!":FORT=1TO5000           :rem 200
```

```
770 NEXT:GOTO170                                    :rem 232
780 IFB$<>" BOOK"THEN690                            :rem 75
790 GOTO730                                         :rem 115
800 PRINT"{CLR}{RVS}{PUR}{6 SPACES}INVENTORY
    {7 SPACES}"                                     :rem 120
810 PRINT"{BLK}YOU NOW HAVE..{RED}"                 :rem 135
820 FORI=1TO10:PRINTE$(I):NEXT                      :rem 127
830 PRINT"{DOWN}{BLK}KEY ANY KEY":POKE198,0:WAIT19
    8,1:GOTO170                                     :rem 167
840 G=1:GOSUB170:GOSUB880:PRINT"I DONT KNOW WHAT A
    ":PRINTB$" IS":GOTO500                          :rem 79
850 IFB$=" KEY"THEN870                              :rem 202
860 GOTO630                                         :rem 112
870 G=1:GOSUB170:GOSUB880:PRINT"{BLK}I SEE NO"B$:G
    OTO500                                          :rem 204
880 PRINT"{HOME}{14 DOWN}{BLK}";:RETURN             :rem 87
890 IFNK<2THENG=1:GOSUB170:GOSUB880:PRINT"YOU DON'
    T HAVE ENOUGH KEYS":GOTO500                     :rem 89
900 PRINT"{CLR}THERE IS THE KING TIEDTO A CHAIR. Y
    OU UNTIE HIM AND YOUR MISSION."                 :rem 140
910 PRINT"IS COMPLETE":END                          :rem 113
920 PRINT"{CLR}{BLK}SORRY, TIME IS UP. YOULOSE!":E
    ND                                              :rem 90
930 PRINT"{CLR}{RVS}{BLU}{13 SPACES}{OFF}":rem 195
940 IFTI$>"001000"THEN920                           :rem 90
950 FORI=1TO11:PRINT"{RVS} {OFF}{11 SPACES}{RVS}
    {OFF}":NEXT                                     :rem 13
960 PRINT"{RVS}{13 SPACES}{OFF}"                    :rem 20
970 PRINT"{HOME}{DOWN}{RIGHT}M{9 SPACES}N"          :rem 77
980 PRINT"{2 RIGHT}M{7 SPACES}N"                    :rem 71
990 PRINT"{3 RIGHT}M{5 @}N"                         :rem 153
1000 FORI=1TO5:PRINT"{3 RIGHT}{M}{RVS}{5 SPACES}
     {OFF}{G}":NEXT                                  :rem 2
1010 PRINT"{3 RIGHT}N{5 T}M"                        :rem 180
1020 PRINT"{2 RIGHT}N{7 SPACES}M"                   :rem 105
1030 PRINT"{RIGHT}N{9 SPACES}M"                     :rem 77
1040 RETURN                                         :rem 165
1050 PRINT"{HOME}{5 DOWN}{5 RIGHT}{2 U}"            :rem 0
1060 PRINT"{5 RIGHT}{2 SPACES}"                     :rem 41
1070 PRINT"{5 RIGHT}{2 SPACES}"                     :rem 42
1080 PRINT"{5 RIGHT}{2 SPACES}"                     :rem 43
1090 PRINT"{5 RIGHT}{2 SPACES}"                     :rem 44
1100 RETURN                                         :rem 162
1110 PRINT"{HOME}{4 DOWN}{RIGHT}{M}M"               :rem 124
1120 FORI=1TO5:PRINT"{RIGHT}{M}{RVS} {OFF}":NEXT
                                                    :rem 38
1130 PRINT"{RIGHT}{M}"                              :rem 90
1140 RETURN                                         :rem 166
1150 PRINT"{HOME}{4 DOWN}{10 RIGHT}N{G}"            :rem 132
```

```
1160 FORI=1TO5:PRINT"{10 RIGHT}{RVS} {OFF}{G}":NEX
     T                                          :rem 45
1170 PRINT"{11 RIGHT}{G}"                       :rem 126
1180 RETURN                                     :rem 170
1190 FORI=0TO5                                   :rem 64
1200 FORJ=0TO3                                   :rem 55
1210 READD$(I,J)                                 :rem 88
1220 READR$(I,J)                                :rem 103
1230 READW%(I,J)                                :rem 110
1240 READX%(I,J)                                :rem 112
1250 READY%(I,J)                                :rem 114
1260 READZ%(I,J)                                :rem 116
1270 READGE$(I,J)                               :rem 166
1280 NEXTJ,I                                    :rem 201
1290 RETURN                                     :rem 172
1300 DATAIT IS VERY DARK BUT{3 SPACES}THERE ARE TO
     RCHES ON{2 SPACES}THE WALLS                :rem 211
1310 DATAYOUR ARE IN THE ENTRY                   :rem 16
1320 DATA1,0,1,0                                 :rem 38
1330 DATA" "                                     :rem 37
1340 DATATHE ROOM IS COVERED{3 SPACES}WITH ABOUT 1
     '' OF DUST                                 :rem 175
1350 DATAYOU ARE IN THE MEETINGROOM.            :rem 164
1360 DATA1,0,0,1                                 :rem 42
1370 DATA" "                                     :rem 41
1380 DATATHERE IS A SKELETON ONTHE FLOOR.       :rem 206
1390 DATAYOU ARE IN A 2 WAY{4 SPACES}HALLWAY.
                                                :rem 247
1400 DATA1,0,1,0                                 :rem 37
1410 DATA" "                                     :rem 36
1420 DATA" "                                     :rem 37
1430 DATAYOU ARE IN AN EMPTY{3 SPACES}PANTRY.
                                                :rem 120
1440 DATA1,0,0,1                                 :rem 41
1450 DATAYOU SEE A CAN OF SOUP                  :rem 173
1460 DATAYOU JUST STEPPED ON A MOUSE            :rem 164
1470 DATAYOU ARE IN A HALLWAY                   :rem 165
1480 DATA1,1,0,0                                 :rem 45
1490 DATA" "                                     :rem 44
1500 DATATHE REMAINS OF A LARGETABLE ARE IN THE
     {6 SPACES}CORNER. THEY'RE BURNED.          :rem 166
1510 DATAYOU ARE IN THE DINING ROOM.             :rem 82
1520 DATA1,1,0,0                                 :rem 40
1530 DATAYOU SEE A KNIFE                        :rem 107
1540 DATATHE ROOM HAS NO SOURCEOF LIGHT VISIBLE; Y
     ET YOU CAN SEE FINE.                       :rem 136
1550 DATA YOU ARE IN THE LIVING ROOM             :rem 56
1560 DATA1,1,1,0                                 :rem 45
1570 DATA" "                                     :rem 43
```

```
1580 DATAYOU SEE LOTS OF STUFF.NONE OF IT HAS ANY
     {4 SPACES}VALUE TO YOU.                :rem 192
1590 DATAYOU ARE IN THE STORAGEROOM.        :rem 182
1600 DATA0,1,0,1                             :rem 39
1610 DATA" "                                 :rem 38
1620 DATATHE{2 SPACES}SINK{2 SPACES}IS FULL OF POT
     ATO CHIPS. AT LEASTTHEY LOOK LIKE CHIPS.
                                             :rem 32
1630 DATAYOU ARE IN A BATHROOM.              :rem 27
1640 DATA0,1,1,0                             :rem 43
1650 DATA" "                                 :rem 42
1660 DATATHE FLOOR IS COATED{3 SPACES}WITH A WAX T
     YPE SUB-{2 SPACES}STANCE.               :rem 72
1670 DATAYOU ARE IN THE KITCHEN              :rem 59
1680 DATA0,1,1,1                             :rem 48
1690 DATAYOU SEE A PIECE OF{4 SPACES}CHEESE.
                                            :rem 219
1700 DATA" "                                 :rem 38
1710 DATAYOU ARE IN A 4 WAY{4 SPACES}HALLWAY.
                                            :rem 245
1720 DATA1,1,1,1                             :rem 44
1730 DATA" "                                 :rem 41
1740 DATAIT IS VERY COLD.                   :rem 181
1750 DATAYOU ARE IN A BEDROOM.              :rem 202
1760 DATA0,0,0,1                             :rem 45
1770 DATAON THE BED IS A GOLD{2 SPACES}KEY :rem 30
1780 DATATHE DESK IS PAINTED{3 SPACES}SILVER AND G
     OLD.                                   :rem 143
1790 DATAYOU ARE IN THE STUDY.              :rem 255
1800 DATA0,0,1,0                             :rem 40
1810 DATAON THE FLOOR IS A{5 SPACES}SILVER KEY.
                                            :rem 173
1820 DATATHE ROOM IS ABOUT 25' HIGH.        :rem 246
1830 DATAYOU ARE IN THE BELFRY.              :rem 37
1840 DATA1,0,0,1                             :rem 45
1850 DATAYOU SEE A BAT FLYING{2 SPACES}ABOVE YOU
                                             :rem 13
1860 DATATHE ROOM HAS A NASTY{2 SPACES}ODOR TO IT.
      YOU COME{2 SPACES}VERY CLOSE TO THROWINGUP!
                                             :rem 13
1870 DATAYOU ARE IN A HALLWAY.              :rem 215
1880 DATA0,1,1,0                             :rem 49
1890 DATA" "                                 :rem 48
1900 DATATHERE ARE MANY NICE{3 SPACES}PAINTINGS BU
     T NOTHING OF VALUE.                    :rem 119
1910 DATAYOU ARE IN THE HOUSE{2 SPACES}MUSEUM.
                                            :rem 192
1920 DATA1,0,0,1                             :rem 44
1930 DATA" "                                 :rem 43
```

```
1940 DATATHE WALLS ARE PAINTED PURPLE AND YELLOW.
                                                  :rem 222
1950 DATAYOU ARE IN A BEDROOM.                    :rem 204
1960 DATA0,0,1,0                                   :rem 47
1970 DATA" "                                       :rem 47
1980 DATAYOU SEE MANY ANIMAL{3 SPACES}HEADS. THE O
     WNER IS{3 SPACES}A HUNTER.                   :rem 141
1990 DATAYOU ARE IN THE DISPLAYROOM.              :rem 187
2000 DATA1,1,0,1                                   :rem 35
2010 DATAYOU SEE A PIECE OF{4 SPACES}CHALK.
                                                  :rem 132
2020 DATA" "                                       :rem 34
2030 DATAYOU ARE IN A BEDROOM.                    :rem 194
2040 DATA1,0,1,0                                   :rem 38
2050 DATA" "                                       :rem 37
2060 DATA THE ROOM SMELLS NICE.                    :rem 29
2070 DATAYOU'RE IN THE LIBRARY.                    :rem 89
2080 DATA1,1,0,1                                   :rem 43
2090 DATAYOU SEE A BOOK.                           :rem 89
2100 DATA" "                                       :rem 33
2110 DATA" "                                       :rem 34
2120 DATA0,0,1,0                                   :rem 36
2130 DATA" "                                       :rem 36
2140 DATATHE EAST WALL DOOR IS LOCKED. THERE ARE T
     WO KEYHOLES.                                 :rem 171
2150 DATA" "                                       :rem 38
2160 DATA0,1,1,0                                   :rem 41
2170 DATA" "                                       :rem 40
2180 DATATHE AIR IS VERY WARM.                    :rem 233
2190 DATAYOU ARE IN A HALLWAY.                    :rem 211
2200 DATA0,1,0,1                                   :rem 36
2210 DATA" "                                       :rem 35
2220 DATA" "                                       :rem 36
2230 DATAYOU ARE IN A BATHROOM.                    :rem 24
2240 DATA0,1,0,0                                   :rem 39
2250 DATA"YOU SEE A BEETLE IN{3 SPACES}THE BATHTUB
                                                  :rem 129
```

Chapter 2

Education

Spider Math

Lee Levitt

Your Commodore computer makes an ideal tutor. This program lets your unexpanded VIC teach basic math operations, using exciting graphics and sound to make the learning fun.

"Spider Math" is an educational game that features sound, custom characters, and a variety of colors. It was designed to make math drills fun, and it can be easily customized to match the skills of any child.

The game was developed so that my six-year-old son could improve his math skills without becoming frustrated over excessively difficult problems. It allows you to choose addition, subtraction, multiplication, or division; you also have the option of setting either of the two numbers in the problems to a maximum value, to a specific number, or to a multiple of a specific number.

For instance, suppose your child is trying to learn the "times three" multiplication tables. After selecting multiplication, simply specify the first number as a 3. Then, if you set the second number to 12, all of the problems will be in the range 3 x 1 to 3 x 12. If you enter 0 (not just RETURN) in response to the prompts, the numbers chosen will automatically range up to 99.

When the game begins, seven random problems appear on the screen, and a custom character (a spider) begins to climb the column beneath the first problem. The spider can be set to any of nine speeds. If the child answers the problem correctly before the spider reaches the top of the column, a happy face appears. But if the answer is incorrect, or if no answer is given in time, a trail of spider clones appears. The correct answer is then displayed under the problem, the incorrect answer appears at the bottom of the spider column, and points are deducted from the score. Each spider clone costs two points from a perfect score of 100.

After completing a screen, hit the space bar to display the score. To play again with different problems but the same settings, press the S key. Press D to change the settings for a completely new game.

Game Notes

When playing the game, enter all answers as two-digit numbers. You do not need to hit RETURN. After the first digit is entered,

you have approximately one second to enter the second digit. If you wait longer than that, you will have to reenter the complete number.

To select addition or subtraction, press the + or − key. To select multiplication, press the letter X, which is more familiar than * to most children. For division, press %.

Spider Math

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
1 GOTO500                                          :rem 255
2 FORI=7706TO7716:POKEI,60:FORX=0TO200:NEXT:POKEI,
  160:NEXT:PRINT"{CLR}+ - X %":DI=2                 :rem 90
3 DEF FNA(X)=INT(RND(1)*X)                         :rem 225
4 GETA$:IFA$="-"THENXY=-1:GOTO10                   :rem 164
6 IFA$="+"THENXY=1:GOTO10                          :rem 248
7 IFA$="%"THENXY=2:GOTO10                          :rem 244
8 IFNOTA$="X"THEN4                                 :rem 89
10 GOSUB250:GOSUB250:XX=0:PRINT"{CLR}{DOWN}SPIDER
   {SPACE}SPEED? (1-9)"                            :rem 46
12 GETA$:IFA$=""THEN12                             :rem 233
13 NN=VAL(A$):IFNN<1ORNN>9THEN12                   :rem 238
16 SP=NN*100:POKE36879,13:POKE36881,128:XX=XX+1:IF
   XX=1THENCT=0                                    :rem 47
17 PRINT"{CLR}":P=-1:Q=FNA(5)+1:OS=30720:DD=DI
                                                   :rem 17
18 BX=99:FORX=7681TO7699STEPDD+1                   :rem 224
19 P=P+1                                           :rem 163
20 BE=FNA(12):IFABS(BE-BX)<4THEN20                 :rem 238
24 BX=BE:Y=X+22*BE:Z=Y+22:LN=Z+22:AS=LN+22:SN=Z-1:
   A(P)=BE+4:XX=99:YY=99                           :rem 86
26 IFNOTQ1=0THENXX=Q1+1                            :rem 209
28 IFNOTQ2=0THENYY=Q2+1                            :rem 215
29 IFAC=0ORML=1THENA=FNA(XX)                       :rem 22
30 IFML=0ORAC=0THEN32                              :rem 30
31 IF(XY=1ORXY=-1)ANDNOTA/AC=INT(A/AC)THEN29
                                                   :rem 15
32 IFAD=0ORMM=1THENB=FNA(YY)                       :rem 21
34 IFMM=0ORAD=0THEN36                              :rem 40
35 IF(XY=1ORXY=-1)ANDNOTB/AD=INT(B/AD)THEN32
                                                   :rem 17
36 IFXY=2ANDB=0THEN32                              :rem 46
37 IFXY=0THENC=A*B:GOTO42                          :rem 76
38 IFNOTXY=2THEN41                                 :rem 159
39 IFXY=2THENC=A/B:IFNOTC=INT(A/B)THEN29           :rem 62
40 GOTO42                                          :rem 3
41 C=A+(B*XY)                                      :rem 191
42 G(P)=C:IFC>99THEN29                             :rem 36
43 IFC<1THEN29                                     :rem 64
```

```
44 X(P)=Ø                                          :rem 206
52 CL=FNA(7):IFCL=ØTHEN52                          :rem 235
53 POKESN+OS,CL:POKELN+OS,CL                        :rem 75
58 GX=A:GY=B:GOSUB2ØØ:A=B:Y=Z:GOSUB2ØØ:POKELN,61:A
   =GX:B=GY:POKELN+1,61:POKELN+1+OS,CL              :rem 19Ø
62 IFXY=1THENPOKESN,43                             :rem 168
63 IFXY=-1THENPOKESN,45                            :rem 216
64 IFXY=ØTHENPOKESN,214                            :rem 217
65 IFXY=2THENPOKESN,59                             :rem 179
66 NEXTX                                             :rem 3
67 FORT=128TO24STEP-1:POKE36881,T:FORP=1TO15:NEXTP
   :NEXTT                                          :rem 148
68 X=Ø                                               :rem 51
69 Y=21                                            :rem 104
71 Y=Y-1:K=7681+((DD+1)*X)+(22*Y):POKEK,6Ø:GOTO8ØØ
                                                     :rem 52
77 FORTX=ØTOSP:NEXT:POKEK,16Ø:IFY=A(X)THEN81
                                                   :rem 133
8Ø GOTO71                                            :rem 9
81 POKEK,16Ø:A=G(X):Y=K-22:GOSUB2ØØ:N=Ø:GOTO91Ø
                                                     :rem 72
86 X=X+1:IFX<7THEN69                               :rem 233
87 GETA$:IFA$=""THEN87                               :rem 1
88 POKE36869,24Ø:POKE36879,27:PRINT"{CLR}SCORE IS
   {SPACE}" 1ØØ-(CT*2)                               :rem 56
92 PRINT"{DOWN}S,D,N?"                             :rem 201
93 GETA$:IFA$=""THEN93                             :rem 251
94 IFA$="D"THENCLR:GOTO1                           :rem 218
95 IFA$="S"THENXX=Ø:POKE36869,255:POKE36879,13:IFX
   Y=ØORXY=2THENA=AC                               :rem 166
97 IFA$="S"THEN16                                  :rem 206
1Ø3 END                                            :rem 1Ø7
2ØØ AA=INT(A/1Ø)+48:AB=A-((AA-48)*1Ø)+48:IFAA=48TH
    ENAA=32                                        :rem 131
2Ø6 POKEY,AA:POKEY+1,AB:POKEY+OS,CL:POKEY+1+OS,CL:
    RETURN                                         :rem 165
25Ø PRINT"{CLR}CONSTANT NUMBER"TM+1                  :rem 43
251 INPUTA$:IFA$=""THEN251                           :rem 3
252 IFTM=1THEN256                                    :rem 3
253 AC=VAL(A$):A=AC:IFA<ØTHEN251                   :rem 1Ø9
254 TM=TM+1:IFNOTAC=ØTHENGOSUB28Ø                  :rem 1Ø6
255 KX=KX+1:RETURN                                 :rem 149
256 AD=VAL(A$):B=AD:IFB<ØTHEN251                   :rem 116
257 IFNOTAD=ØTHENGOSUB28Ø                            :rem 89
26Ø IFAC=ØOR(ML=1AND(XY=1ORXY=-1))THENGOSUB27Ø
                                                   :rem 137
263 IFAD=ØOR(MM=1AND(XY=-1ORXY=1))THENGOSUB275
                                                   :rem 147
264 IFNOTAC=ØANDNOTAD=ØTHENRETURN                  :rem 21Ø
265 RETURN                                         :rem 125
```

```
270 PRINT"{CLR}MAX. 1ST #?"                          :rem 75
271 INPUTA$:IFA$=""THEN271                           :rem 7
272 Q1=VAL(A$):RETURN                                :rem 13
275 PRINT"{CLR}MAX. 2ND #?"                          :rem 60
276 INPUTA$:IFA$=""THEN277                           :rem 18
277 Q2=VAL(A$):RETURN                                :rem 19
280 IFXY=0ORXY=2THENRETURN                           :rem 23
281 PRINT"{2 DOWN}MULTIPLE?"                         :rem 57
282 GETA$:IFA$=""THEN282                             :rem 91
284 IFA$="N"THENRETURN                               :rem 112
286 IFKX=0THENML=1                                   :rem 117
287 IFKX=1THENMM=1                                   :rem 120
288 RETURN                                           :rem 130
500 PRINT"{CLR}":POKE52,29:POKE56,29:CLR             :rem 230
505 PRINTSPC(5)"SPIDER MATH"                         :rem 200
510 FORI=7168TO7679:POKEI,PEEK(I+25600):NEXT
                                                     :rem 151
530 POKE36869,255:FORI=7648TO7648+7:READA:POKEI,A:
    NEXT                                             :rem 44
540 DATA129,153,102,60,255,60,66,66                  :rem 239
550 FORI=7664TO7664+7:READA:POKEI,A:NEXT             :rem 233
560 DATA60,66,165,129,165,153,66,60                  :rem 250
562 FORI=7656TO7656+7:READA:POKEI,A:NEXT             :rem 238
563 DATA0,0,0,255,0,0,0,0                            :rem 216
564 FORI=7640TO7640+7:READA:POKEI,A:NEXT:GOTO2
                                                     :rem 135
565 DATA0,24,0,255,0,24,0,0                          :rem 70
800 GETB$:IFB$=""THEN77                              :rem 43
810 N=VAL(B$):TH=TI/60                               :rem 3
811 GETC$:IFNOTC$=""THEN819                          :rem 84
812 TJ=TI/60:IFTJ-TH<1THEN811                        :rem 14
813 GOTO77                                           :rem 67
819 N1=VAL(C$):N=N*10+N1                             :rem 65
820 IFN<1ORN>99THEN77                                :rem 32
825 IFN=G(X)THEN931                                  :rem 117
910 Y=7681+(3*X)+(22*(A(X)-1)):A=G(X):GOSUB200
                                                     :rem 206
911 FORM=KTO8121+(3*X)STEP22:CT=CT+1                 :rem 152
912 POKE36878,15:POKE36877,220:FORO=15TO0STEP-1:NE
    XTO:POKE36877,60:POKE36878,60                    :rem 2
921 POKEM,60:NEXT:A=N:Y=M+22                         :rem 213
922 GOSUB200:GOTO935                                 :rem 195
931 POKEK,62:POKE36878,15:FORL=1TO3:FORM=250TO240S
    TEP-1:POKE36876,M                                :rem 108
932 NEXT:NEXT:POKE36878,60:POKE36876,60              :rem 106
933 A=G(X):Y=7681+((DD+1)*X)+(22*(A(X)-1)):GOSUB20
    0                                                :rem 213
935 Y=A(X):GOTO86                                    :rem 2
```

Merry-Go-Match

Griff and Sheila Johnson
64 Translation by David Florance

Any educational program should attract and hold a child's interest. "Merry-Go-Match" does just that, through the use of colorful graphics, exciting sounds, and dynamic visual displays. Versions are included for the unexpanded VIC and for the 64.

Educational programs are of little value if they do not hold the child's interest. "Merry-Go-Match" is similar to the familiar TV game *Concentration.* Designed for young children, it gives practice in number, letter, and word recognition. Few programs hold the interest of children and encourage them to play again and again, but this one has been enjoyed by our preschooler time after time. Two-, three-, and four-letter words may be used in this game, or you can substitute single letters, numbers, or graphic symbols.

Eight pairs of matching words are randomly arranged and hidden behind 16 colored and numbered squares, and the object is to locate each pair. The player is asked to make a first choice by pressing the number of a square, followed by RETURN, which reveals the word behind that particular square. If the player's second choice then produces a match, the computer sounds a high tone and colors both squares cyan to match the border. If the two do not match, the computer sounds a low tone and both words are replaced by the numbered squares.

Play continues until all matches are completed. When all pairs of matching words have been located, the phrase YOU DID IT! appears on the screen, and the computer plays a short series of random musical notes. It also displays the number of guesses required to complete the game and offers the option of playing again.

When you run this program, the computer will start by counting down from 16 to 0. That tells you that it is randomly arranging the pairs of words. Sometimes the countdown will go quickly; at other times it may take several seconds. In any case, when the countdown reaches 0 the game will begin.

Program Description

Lines

5-10	Read words
15-30	Arrange words in random order
40-65	Draw border
70-80	Draw numbered squares
85-97	Ask for first guess
100-113	Ask for second guess
395	Data for color of squares
401-499	Data for words, letters, numbers, or graphic symbols
500-530	Redraw colored squares after incorrect guess
600-650	Draw cyan squares to replace matched words
1000-1030	Subroutine to reveal words behind squares

Program 1. Merry-Go-Match, VIC Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
Ø DIMC(17),K$(16),D$(16),A$(16),B(16):FORI=1TO16:R
  EADK$(I):NEXT                                  :rem 206
1 FORI=1TO17:READC(I):NEXT                       :rem 134
2 FORI=1TO16:READB(I):NEXT                       :rem 133
3 SC=3Ø72Ø                                         :rem 2
5 FORI=1TO8:REM READS WORDS                      :rem 202
6 GS=Ø                                            :rem 61
1Ø READA$(I):A$(I+8)=A$(I):NEXT                  :rem 205
12 PRINT"PLEASE STAND BY WHILE I COUNT DOWN TO ZER
   O "                                           :rem 105
15 FORI=1TO16:REM ARRANGES WORDS IN RANDOM ORDER
                                                 :rem 226
17 PRINTTAB(8);17-I                              :rem 110
2Ø J=INT(RND(X)*16+1):IFA$(J)="Ø"THEN2Ø          :rem 170
25 D$(I)=A$(J)                                   :rem 166
3Ø A$(J)="Ø":NEXT:PRINTTAB(8);Ø                   :rem 31
32 FOR I=1TO8ØØ:NEXT                            :rem 183
35 PRINT"{CLR}"                                  :rem 204
4Ø FORI=1TO22:REM DRAWS BORDER                   :rem 102
45 POKE77Ø2-1+I,1Ø2:POKE77Ø2+SC-1 +I,Ø            :rem 33
5Ø POKE77Ø2+I*22,1Ø2:POKE77Ø2+SC+I*22,Ø          :rem 125
55 POKE8164+I,1Ø2:POKE8164+SC+I,Ø                :rem 108
6Ø POKE7723+I*22,1Ø2:POKE7723+SC+I*22,Ø          :rem 132
65 NEXT                                          :rem 170
7Ø FORM=1TO16:FORJ=ØTO4:FORI=ØTO4                :rem 112
75 POKE7725+B(M)+I+J*22,16Ø:POKE7725+SC+B(M)+I+J*2
   2,C(M)                                         :rem 67
8Ø NEXTI:NEXTJ:PRINT"{HOME}"SPC((B(M)+1)/2)SPC(B(M
   )/2)SPC(9Ø)K$(M):NEXTM                        :rem 225
```

```
85 FORI=ØTO21:POKE768Ø+I,16Ø:POKE768Ø+SC+I,1:NEXT:
   INPUT"{HOME} FIRST GUESS{3 SPACES}";G$ :rem 164
9Ø IFVAL(G$)<1ORVAL(G$)>16THEN 85          :rem 133
91 G1=VAL(G$)                              :rem 19Ø
93 IFD$(G1)="Ø"THEN85                      :rem 121
95 M=G1                                    :rem 112
97 GOSUB1ØØØ                               :rem 177
1ØØ FORI=ØTO21:POKE768Ø+I,16Ø:POKE768Ø+SC+I,1:NEXT
    :INPUT"{HOME}SECOND GUESS";G$          :rem 252
11Ø IFVAL(G$)<1ORVAL(G$)>16THEN1ØØ         :rem 21Ø
111 G2=VAL(G$)                             :rem 232
113 IFD$(G2)="Ø"ORG1=G2THEN1ØØ             :rem 15Ø
12Ø GS=GS+1:M=G2:GOSUB1ØØØ                  :rem 24
13Ø IFD$(G1)=D$(G2)THEN2ØØ                 :rem 132
14Ø PRINT"{HOME}NO MATCH{9 SPACES}":FORI=1TO1ØØØ:N
    EXT:POKE36878,15:POKE36874,135         :rem 125
141 FORK=1TO1ØØØ:NEXT:POKE36878,Ø:POKE36874,Ø
                                           :rem 185
15Ø GOSUB5ØØ                               :rem 171
195 GOTO85                                  :rem 69
2ØØ PRINT"{HOME}MATCH{13 SPACES}":FORI=1TO1ØØØ:NEX
    T:POKE36878,15:POKE36876,24Ø           :rem 22Ø
2Ø1 FORK=1TO1ØØØ:NEXT:POKE36878,Ø:POKE36876,Ø:GOSU
    B6ØØ                                    :rem 8
21Ø D$(G1)="Ø":D$(G2)="Ø"                  :rem 146
22Ø FORI=1TO16                              :rem 6Ø
23Ø IFD$(I)<>"Ø"THEN85                     :rem 176
24Ø NEXT                                   :rem 213
245 PRINT"{1Ø DOWN}{4 RIGHT}!!YOU DID IT!!":PRINT"
    {DOWN}{5 RIGHT}"GS" GUESSES"           :rem 35
246 POKE36878,15:FORI=1TO1ØØ:Y=INT(RND(1)*5Ø+2ØØ)
                                           :rem 55
247 POKE36876,Y:FORJ=1TO5Ø:NEXTJ:NEXTI:POKE36878,Ø
    :POKE36876,Ø                           :rem 1Ø9
25Ø GOTO3ØØ                                 :rem 99
3ØØ FORI=ØTO21:POKE768Ø+I,16Ø:POKE768Ø+SC+I,1:NEXT
    :PRINT"{HOME}PLAY AGAIN"               :rem 168
3Ø5 GETX$:IFX$="Y"THEN5                    :rem 119
31Ø IFX$<>"N"THEN3Ø5{19 SPACES}            :rem 114
32Ø PRINT"{CLR}BYE":END                    :rem 234
39Ø DATA1,2,3,4,5,6,7,8,9,1Ø,11,12,13,14,15,16
                                           :rem 227
395 DATA6 ,4,5,6,4,5,6,4,5,6,4,5,6,4,5,6,3:REM DAT
    A FOR COLOR OF SQUARES                  :rem 86
396 DATAØ ,5,1Ø,15 ,11Ø,115,12Ø,125,22Ø,225,23Ø,23
    5,33Ø,335,34Ø,345                      :rem 144
4ØØ REM ADD WORDS IN LINES 4Ø1TO499        :rem 192
42Ø DATASAT,HIT,FAT,FAN,HAT,SUN,PIG,POT,PAN:rem 18
43Ø DATATOP,BAT,CAR,MAN,EAR,LIP,SIT,TOO     :rem Ø
```

```
440 DATAHOT,RUN,YOU,MAN,TOP,FOR,ONE,JAR     :rem 56
450 DATABED,RAT,CAT,DOG,ALL,SIP,POP,MOM     :rem 232
455 DATA STOP,TOPS,LOOK,BOOK,RICE,NICE,TREE,FREE
                                             :rem 108
460 DATA"Z","X","A","S","Q","W","+","V"      :rem 167
500 M=G1:REM IF NO MATCH REPLACES COLORED SQUARES
                                             :rem 201
510 FORJ=0TO4:FORI=0TO4                      :rem 185
520 POKE7725+B(M)+I+J*22,160:POKE7725+SC+B(M)+I+J*
    22,C(M)                                  :rem 110
530 NEXTI:NEXTJ:PRINT"{HOME}"SPC((B(M)+1)/2)SPC(B(
    M)/2)SPC(90)M                            :rem 139
540 IFM=G2THEN560                            :rem 245
550 M=G2:GOTO510                             :rem 166
560 RETURN                                   :rem 123
600 M=G1:REM IF WORDS MATCH COLORS SQUARES CYN
                                              :rem 33
610 FORJ=0TO4:FORI=0TO4                      :rem 186
620 POKE7725+B(M)+I+J*22,160:POKE7725+SC+B(M)+I+J*
    22,3:NEXTI:NEXTJ                          :rem 70
640 IFM=G2THEN660                            :rem 247
650 M=G2:GOTO610                             :rem 168
655 POKE36878,15:POKE36876,225:FORK=1TO1000:NEXT:P
    OKE36876,0:POKE36878,0                    :rem 12
660 RETURN                                   :rem 124
1000 FORJ=0TO4:FORI=0TO4:REM REVEALS WORD BEHIND S
     QUARE                                   :rem 203
1010 POKE7725+B(M)+I+J*22,160:POKE7725+SC+B(M)+I+J
     *22,1:NEXTI:NEXTJ                       :rem 110
1020 PRINT"{HOME}"SPC((B(M)+1)/2)SPC(B(M)/2)SPC(90
     )D$(M)                                  :rem 234
1030 RETURN                                  :rem 164
```

Program 2. Merry-Go-Match, 64 Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 SC=1024:CS=53281:CB=53280:CM=55296:CV=54272:ES=
   1984:EM=56256                             :rem 37
15 PRINT"{CLR}{BLU}":POKECS,7:POKECB,7:PRINTSPC(24
   0)TAB(5)                                  :rem 141
17 PRINT"M E R R Y - G O - M A T C H "       :rem 37
18 PRINTSPC(240)TAB(11)"{RVS}PRESS ANY KEY{OFF}"
                                             :rem 147
19 GETDX1$:IF DX1$=""THEN19                  :rem 15
20 DIMK$(16),C1(16),D$(16),A$(16),X(80),Y(80)
                                             :rem 203
30 FORI=1TO16:READK$(I):NEXT:FORI=1TO16:READC1(I):
   NEXT                                      :rem 162
40 FORT=1TO80:READX(T):NEXT                  :rem 228
```

```
50 FORT=1TO80:READY(T):NEXT                    :rem 230
90 FORI=1TO8:REM READS WORDS                   :rem 254
92 GS=0                                        :rem 114
94 READA$(I):A$(I+8)=A$(I):NEXT:POKECS,14:POKECB,1
   :PRINT"{WHT}"                               :rem 196
96 PRINT"{CLR}"SPC(80)TAB(4)                   :rem 206
98 PRINT"PLEASE STAND BY WHILE{19 SPACES}I COUNT D
   OWN TO ZERO "                               :rem 119
100 FORI=1TO16:REM ARRANGES WORDS IN RANDOM ORDER
                                               :rem 13
110 PRINTTAB(8);17-I                           :rem 152
120 J=INT(RND(X)*16+1):IFA$(J)="0"THEN120      :rem 12
125 D$(I)=A$(J)                                :rem 215
130 A$(J)="0":NEXT:PRINTTAB(8);0               :rem 80
140 FOR I=1TO800:NEXT                          :rem 231
270 PRINT"{CLR}{WHT}":POKECS,6:POKECB,6        :rem 179
280 FORH=1TO40:FORT=SC+X(H)TOSC+Y(H):POKET+CV,14:P
    OKET,160:NEXT:NEXT                         :rem 124
290 FORH=41TO80:FORT=SC+X(H)TOSC+Y(H):POKET+CV,4:P
    OKET,160:NEXT:NEXT                         :rem 132
400 F$="{HOME}{2 DOWN}"                        :rem 180
412 X=10:Y=40                                  :rem 193
414 FORT=1TO4                                  :rem 25
420 FORR=XTOYSTEP10:PRINTF$TAB(3)SPC(R-10);K$(R/10
    ):NEXT                                     :rem 246
430 X=X+40:Y=Y+40:F$=F$+"{4 DOWN}"             :rem 201
440 NEXT:IFTRTHENPRINTF$:PRINT"{3 DOWN}{22 SPACES}
    ":GOTO580                                  :rem 186
500 FORD=SC+2TOSC+35                           :rem 188
510 POKED+CV,0:POKED,102:NEXT                  :rem 14
520 FORD=SC+2TOES-121STEP40                    :rem 142
530 POKED+CV,0:POKED,102:NEXT                  :rem 16
540 FORD=SC+35TOES-84STEP40                    :rem 158
550 POKED+CV,0:POKED,102:NEXT                  :rem 18
560 FORD=ES-118TOES-86                         :rem 56
570 POKED+CV,0:POKED,102:NEXT                  :rem 20
580 SG=1:PRINTF$                               :rem 214
582 IFTR<1THENTR=0                             :rem 131
685 INPUT"{3 DOWN}FIRST GUESS{3 SPACES}";G$:rem 95
690 IFVAL(G$)<1ORVAL(G$)>16THEN 580            :rem 235
691 G1=VAL(G$)                                 :rem 244
693 IFD$(G1)="0"THEN 580                       :rem 223
695 SB=G1                                      :rem 238
697 GOSUB800                                   :rem 190
700 SG=2:PRINTF$:INPUT"{3 DOWN}SECOND GUESS
    {6 SPACES}";G$:TR=TR+1                     :rem 26
705 IFVAL(G$)<1ORVAL(G$)>16THEN 700            :rem 226
710 G2=VAL(G$)                                 :rem 237
715 IFD$(G2)="0"ORG1=G2THEN 700                :rem 164
```

```
720 SB=G2:GOSUB800                                    :rem 54
730 IFD$(G1)=D$(G2)THEN20000                          :rem 234
740 GOSUB2000:PRINTF$:PRINT"{3 DOWN}NO MATCH
    {17 SPACES}":FORI=1TO1000:NEXT                    :rem 11
744 SG=3:GOSUB800                                     :rem 251
745 IFDITHENSB=G1:GOSUB800                            :rem 135
750 GOTO400                                           :rem 105
800 C1=6:C2=6                                         :rem 161
801 IFSG=1THENQ=G1                                    :rem 101
802 IFSG=2THENQ=G2                                    :rem 104
803 IFSG=3THENC1=14:C2=4:DI=1                         :rem 206
804 IFSG=4 THEN C1=3:C2=3:DI=1                        :rem 157
805 ONSBGOTO810,820,830,840,850,860,870,880,890,90
    0,910,920,930,940,950,960                         :rem 101
810 FORH=1TO5:FORT=SC+X(H)TOSC+Y(H):POKET,160:POKE
    T+CV,C1:NEXT:NEXT                                 :rem 91
815 IFSG<3THENPRINT"{HOME}{WHT}{3 DOWN}"TAB(3);"
    {2 SPACES}"D$(Q)                                  :rem 101
817 GOTO 990                                          :rem 123
820 FORH=41TO45:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C2:NEXT:NEXT                               :rem 197
825 IFSG<3THENPRINT"{HOME}{WHT}{3 DOWN}"TAB(11);"
    {2 SPACES}"D$(Q)                                  :rem 149
827 GOTO 990                                          :rem 124
830 FORH=6TO10:FORT=SC+X(H)TOSC+Y(H):POKET,160:POK
    ET+CV,C1:NEXT:NEXT                                :rem 142
835 IFSG<3THENPRINT"{HOME}{WHT}{3 DOWN}"TAB(20);"
    {2 SPACES}"D$(Q)                                  :rem 150
837 GOTO 990                                          :rem 125
840 FORH=46TO50:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C2:NEXT:NEXT                               :rem 200
845 IFSG<3THENPRINT"{HOME}{WHT}{3 DOWN}"TAB(28);"
    {2 SPACES}"D$(Q)                                  :rem 159
847 GOTO 990                                          :rem 126
850 FORH=51TO55:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C2:NEXT:NEXT                               :rem 202
855 IFSG<3THENPRINT"{HOME}{WHT}{8 DOWN}"TAB(3);"
    {2 SPACES}"D$(Q)                                  :rem 190
857 GOTO 990                                          :rem 127
860 FORH=11TO15:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C1:NEXT:NEXT                               :rem 194
865 IFSG<3THENPRINT"{HOME}{WHT}{8 DOWN}"TAB(11);"
    {2 SPACES}"D$(Q)                                  :rem 238
867 GOTO 990                                          :rem 128
870 FORH=56TO60:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C2:NEXT:NEXT                               :rem 205
875 IFSG<3THENPRINT"{HOME}{WHT}{8 DOWN}"TAB(20);"
    {2 SPACES}"D$(Q)                                  :rem 239
877 GOTO 990                                          :rem 129
```

```
880 FORH=16TO20:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C1:NEXT:NEXT                          :rem 197
885 IFSG<3THENPRINT"{HOME}{WHT}{8 DOWN}"TAB(28);"
    {2 SPACES}"D$(Q)                             :rem 248
887 GOTO 990                                     :rem 130
890 FORH=21TO25:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C1:NEXT:NEXT                          :rem 199
895 IFSG<3THENPRINT"{HOME}{WHT}{13 DOWN}"TAB(3);"
    {2 SPACES}"D$(Q)                              :rem 23
897 GOTO 990                                     :rem 131
900 FORH=61TO65:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C2:NEXT:NEXT                          :rem 200
905 IFSG<3THENPRINT"{HOME}{WHT}{13 DOWN}"TAB(11);"
    {2 SPACES}"D$(Q)                              :rem 62
907 GOTO 990                                     :rem 123
910 FORH=26TO30:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C1:NEXT:NEXT                          :rem 193
915 IFSG<3THENPRINT"{HOME}{WHT}{13 DOWN}"TAB(20);"
    {2 SPACES}"D$(Q)                              :rem 63
917 GOTO 990                                     :rem 124
920 FORH=66TO70:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C2:NEXT:NEXT                          :rem 203
925 IFSG<3THENPRINT"{HOME}{WHT}{13 DOWN}"TAB(28);"
    {2 SPACES}"D$(Q)                              :rem 72
927 GOTO 990                                     :rem 125
930 FORH=71TO75:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C2:NEXT:NEXT                          :rem 205
935 IFSG<3THENPRINT"{HOME}{WHT}{18 DOWN}"TAB(3);"
    {2 SPACES}"D$(Q)                             :rem 103
937 GOTO 990                                     :rem 126
940 FORH=31TO35:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C1:NEXT:NEXT                          :rem 197
945 IFSG<3THENPRINT"{HOME}{WHT}{18 DOWN}"TAB(11);"
    {2 SPACES}"D$(Q)                             :rem 151
947 GOTO 990                                     :rem 127
950 FORH=76TO80:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C2:NEXT:NEXT                          :rem 208
955 IFSG<3THENPRINT"{HOME}{WHT}{18 DOWN}"TAB(20);"
    {2 SPACES}"D$(Q)                             :rem 152
957 GOTO 990                                     :rem 128
960 FORH=36TO40:FORT=SC+X(H)TOSC+Y(H):POKET,160:PO
    KET+CV,C1:NEXT:NEXT                          :rem 200
965 IFSG<3THENPRINT"{HOME}{WHT}{18 DOWN}"TAB(28);"
    {2 SPACES}"D$(Q)                             :rem 161
990 RETURN                                       :rem 130
1640 DATA1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
                                                 :rem 18
1660 DATA14,4,14,4,14,4,14,4,14,4,14,4,14,4,14,4
                                                 :rem 67
```

```
2000 FORL=CVTOCV+24:POKEL,0:NEXT                    :rem 34
2005 POKECV+24,15                                   :rem 178
2010 POKECV+5,128                                   :rem 178
2020 POKECV+6,128                                   :rem 180
2030 POKECV+1,3                                      :rem 72
2040 POKECV,5                                       :rem 239
2050 POKECV+4,33                                    :rem 128
2060 FORT=1TO1100:NEXT                               :rem 79
2070 POKECV+4,32                                    :rem 129
2080 RETURN                                         :rem 170
2500 FORL=CVTOCV+24:POKEL,0:NEXT                     :rem 39
2510 POKECV+24,15                                   :rem 179
2520 POKECV+19,72                                   :rem 187
2530 POKECV+20,129                                  :rem 231
2540 POKECV+18,33                                   :rem 185
2550 FORT=1TO75:D=INT(RND(1)*198)+3                  :rem 11
2560 POKECV+15,D:POKECV+14,50:POKECV+18,17          :rem 227
2570 NEXT                                            :rem 13
2580 POKECV+18,32:RETURN                            :rem 214
4000 DATA 43,83,123,163,203,59,99,139,179,219
                                                    :rem 187
4010 DATA 251,291,331,371,411,267,307,347,387,427
                                                    :rem 125
4020 DATA 443,483,523,563,603,459,499,539,579,619
                                                    :rem 165
4030 DATA 651,691,731,771,811,667,707,747,787,827
                                                    :rem 167
4040 DATA 51,91,131,171,211,67,107,147,187,227
                                                    :rem 220
4050 DATA 243,283,323,363,403,259,299,339,379,419
                                                    :rem 148
4060 DATA 451,491,531,571,611,467,507,547,587,627
                                                    :rem 150
4080 DATA 643,683,723,763,803,659,699,739,779,819
                                                    :rem 191
4090 DATA 50,90,130,170,210,66,106,146,186,226
                                                    :rem 215
4100 DATA 258,298,338,378,418,274,314,354,394,434
                                                    :rem 150
4110 DATA 450,490,530,570,610,466,506,546,586,626
                                                    :rem 136
4120 DATA 658,698,738,778,818,674,714,754,794,834
                                                    :rem 192
4130 DATA 58,98,138,178,218,74,114,154,194,234
                                                    :rem 245
4140 DATA 250,290,330,370,410,266,306,346,386,426
                                                    :rem 119
4150 DATA 458,498,538,578,618,474,514,554,594,634
                                                    :rem 175
```

```
4160 DATA 650,690,730,770,810,666,706,746,786,826
                                                :rem 161
4500 REM ADD WORDS                               :rem 5
4510 DATAHOT,RUN,YOU,MAN,TOP,FOR,ONE,JAR        :rem 106
4520 DATABED,RAT,CAT,DOG,ALL,SIP,POP,MOM         :rem 26
4530 DATA STOP,TOPS,LOOK,BOOK,RICE,NICE,TREE,FREE
                                                :rem 154
4540 DATASAT,HIT,FAT,FAN,HAT,SUN,PIG,POT,PAN
                                                 :rem 73
4550 DATA"Z","X","A","S","Q","W","+","V"        :rem 219
5000 FORJ=0TO4:FORI=0TO4:REM REVEALS WORD BEHIND S
     QUARE                                      :rem 207
5010 POKE7725+B(M)+I+J*22,160:POKE7725+SC+B(M)+I+J
     *22,1:NEXTI:NEXTJ                          :rem 114
5020 PRINT"{HOME}"SPC((B(M)+1)/2)SPC(B(M)/2)SPC(90
     )                                          :rem 232
5030 RETURN                                     :rem 168
20000 GOSUB2500:PRINTF$:PRINT"{3 DOWN}{YEL}MATCH!
      {19 SPACES}{WHT}":FORK=1TO1000:NEXT       :rem 144
20005 SG=4:GOSUB800:IFDITHENSB=G1:GOSUB800:rem 117
20010 D$(G1)="0":D$(G2)="0"                     :rem 242
20020 FORI=1TO16                                :rem 156
20030 IFD$(I)<>"0"THEN400                        :rem 55
20040 NEXT                                       :rem 53
20045 PRINT"{CLR}"SPC(160)TAB(11)"!!YOU DID IT!!":
      PRINTSPC(120)TAB(12)TR" GUESSES"          :rem 15
20055 PRINTSPC(80)TAB(12)"{RVS}PRESS RETURN{OFF}"
                                                 :rem 8
20056 FORL=CVTOCV+24:POKEL,0:NEXT               :rem 93
20057 POKECV+24,15:POKECV+19,72:POKECV,129:POKECV+
      18,33                                      :rem 7
20058 D=105:FORT=1TO7:D=D+1:POKECV+15,D:POKECV+14,
      5:POKECV+18,33                            :rem 76
20060 NEXT                                      :rem 55
20065 GETRS$:IFRS$<>CHR$(13)THEN20045           :rem 146
20075 POKECV+18,32:POKECV+24,0                  :rem 218
30000 PRINT"{CLR}{WHT}":POKECS,2:POKECB,1:PRINTSPC
      (160)TAB(11)"PLAY AGAIN?"                 :rem 60
30010 GETRP$:IF RP$="" THEN 30010               :rem 205
30020 IF RP$=CHR$(89)THENRUN                    :rem 110
```

Hatch It

Neil Murray

"Hatch It" gives preschoolers a delightful introduction to computers — and since it is a no-lose game, the experience will always be a good one. Versions are included for the unexpanded VIC and the 64.

"Hatch It" was developed to provide musical and graphic entertainment for young children. The game evolved to satisfy my three-year-old son's demand for continuous music and my own desire for a program that asks the player to create rather than destroy.

In Hatch It, the child identifies with a smiley character named Sunny, whose job is to hatch the eggs that appear on the screen. To start the game, the child types in his or her name and then chooses either the keyboard or the joystick for moving Sunny around the screen. Each time Sunny arrives at (and hatches) an egg, a butterfly flaps up, a spider drifts down, or an inchworm crawls away. The bugs come in different colors and move at different speeds. There is no score, but when all the eggs have been hatched, the music changes from "Alouette" to "Frère Jacques," the screen clears, and an enlarged Sunny blinks his blue eyes and congratulates the player. The child will need your help to start the game, especially the first time.

This game offers a choice of joystick or keyboard control, and I suggest that you encourage the child to try both. In my experience, young children want to use the joystick, but find it frustrating to manipulate. The keyboard is less intriguing but often proves more satisfying. Here are the control keys and their functions:

< — move left
> — move right
A — move up
Z — move down

The program uses a machine language wedge inserted into the VIC's interrupt routine, so the only way to shut off the program is to press the RUN/STOP and RESTORE keys.

Music

Lines 350-430 load locations 830-996 of the VIC's cassette buffer with a machine language wedge that allows two different songs to

be played at different times during the game. The wedge is a modified version of the "VIC Musician" (*COMPUTE!*, July 1983). Initially, a SYS 830 tells the wedge to look for "Alouette" beginning at RAM location 7552 ($1D80), while a SYS 842 tells it to look for "Frère Jacques" starting at location 933 ($3A5) in the tape buffer. Then the wedge cuts the volume on the last two cycles of each note, which has the effect of separating successive notes from each other.

Making It Fit

Each note of each song requires two data entries, one for the note and one for the duration, so it was difficult to squeeze Hatch It into an unexpanded VIC. However, two memory-saving tricks saved the day. The VIC requires at least 512 bytes (locations 7168 to 7679 in this case) to be set aside for a special character set, but since the program doesn't keep score, the last hundred-odd bytes of this block (which includes the numbers) are never put to use. It was therefore possible to POKE the data for "Alouette" into the 128-byte block that runs from locations 7552 to 7679.

The other trick was the substitution of relational operations for IF-THEN statements. When BASIC evaluates relations such as (X=25), it assigns them a value of 0 if false and a value of −1 if true. This enables the programmer to include parenthetic relations as factors in equations. For example, line 230 instructs the BASIC interpreter to choose alternate POKE values for the variable CH so that the butterfly character appears to flap its wings. Normally, one might use three lines of code to swap back and forth between POKE values of 29 and 30:

```
230 IF CH=29 THEN CH=30: GOTO 240
235 IF CH=30 THEN CH=29
240 GOSUB 280: GOTO 220
```

But using the 0 and −1 values from tests of the variable CH, it's possible to substitute a one-line formula:

```
230 CH = CH + (CH=30) - (CH=29) : GOSUB 280: GOTO
    {SPACE}220
```

When CH has a value of 29, then (CH=30) is false (has a value of 0) and (CH=29) is true (has a value of −1). Therefore, CH = 29 + 0 − (−1) = 30, precisely the change needed.

Hatch It on the Expanded VIC

Two steps are required to run this program on an expanded VIC. Before typing or loading the program, enter POKE 44,33: POKE 8448,00 in direct mode and hit RETURN to move the start of BASIC high enough to protect the screen and the special character set. Second, delete line 10 of the listing and replace it with the following:

```
10 POKE 648,30:POKE36866,150:POKE36869,240:PRINTCH
   R$(147)
```

A word of caution: Do not try to run Hatch It with the Super Expander cartridge plugged in. The Super Expander has its own special interrupt program that will render the program voiceless.

Program 1. Hatch It, VIC Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 POKE52,28:POKE56,28                          :rem 247
20 CD=30720:DIMH%(3):H%(0)=1:H%(1)=-1:H%(2)=22:H%(
   3)=22:W=-1                                   :rem 68
30 PRINTCHR$(147)TAB(51)CHR$(18)"HATCHIT!":PRINT:I
   NPUT"NAME";N$:PRINT:PRINT"HI, "N$"!"         :rem 154
40 PRINT:INPUT"J=JOY, K=KEYS";JK$:JK=1-(JK$="J"):P
   RINT:PRINT"PLEASE WAIT...."                  :rem 20
50 GOSUB350                                     :rem 125
60 POKE36864,4:POKE36869,255:POKE36879,13       :rem 75
70 GOSUB340:FORI=7704TO8144STEP22:Z=INT(RND(1)*19)
   :POKEI+Z+CD,6:POKEI+Z,42:NEXT                :rem 159
80 NP=7933+(PEEK(7933)<>32):GOSUB150            :rem 16
90 ONJKGOSUB160,180:NP=OP+DP:PK=PEEK(NP):IFNP<7681
   ORNP>8185ORPK=0THEN90                        :rem 29
100 IFPK=32THENPOKEOP,32:GOSUB150:GOTO90        :rem 23
110 OL=NP:OC=34:POKEOP,32:GOSUB150              :rem 97
120 HC=INT(RND(1)*5)+1:H=INT(RND(1)*3)+1:ONHGOSUB2
    40,210,240                                  :rem 136
130 CT=CT+1:IFCT=21THENCT=0:GOSUB290:GOTO70
                                                :rem 189
140 GOTO90                                      :rem 55
150 POKENP+CD,7:POKENP,34:OP=NP:RETURN          :rem 224
160 GETA$:IFA$=""THEN160                        :rem 81
161 A=ASC(A$):IF A<>44ANDA<>46ANDA<>65ANDA<>90THEN
    160                                         :rem 61
170 DP=(A=44)-(A=46)+22*(A=65)-22*(A=90):RETURN
                                                :rem 6
180 POKE37154,127:J3=-((PEEK(37152)AND128)=0):POKE
    37154,255:P=PEEK(37137)                     :rem 31
190 J1=-((PAND8)=0):J2=((PAND16)=0):J0=((PAND4)=0)
    :DP=J2+J3+22*(J0+J1):IFDP=0THEN180          :rem 183
```

```
200 FORT=1TO80:NEXT:RETURN                          :rem 217
210 T=INT(RND(1)*200)+10:W=-W:CH=29                 :rem 205
220 NL=OL+W:IFPEEK(NL)=0THENPOKEOL,32:RETURN
                                                    :rem 114
230 CH=CH+(CH=30)-(CH=29):GOSUB280:GOTO220:rem 155
240 HI=H-2:CH=29+2*HI:T=INT(RND(1)*20)+15  :rem 26
250 Z=INT(RND(1)*4):NL=OL+HI*H%(Z):PK=PEEK(NL):IFN
    L<7681ORNL>8185THENPOKEOL,32:RETURN     :rem 98
260 IFPK<>32THEN250                                 :rem 103
270 CH=CH+(CH=28)-(CH=27):GOSUB280:GOTO250:rem 167
280 POKEOL,OC:OC=32:POKENL+CD,HC:POKENL,CH:OL=NL:F
    ORI=1TOT:NEXT:RETURN                            :rem 196
290 SYS842:GOSUB340:PRINTTAB(67)CHR$(156)"SUPER, "
    N$"!"TAB(141)CHR$(158)"@@@@"                    :rem 155
300 PRINTTAB(8)"@@@@@@"SPC(16)"@ @@ @"SPC(15)"@@@@
    @@@@"SPC(14)"@@ @@ @@"SPC(15)"@@{2 SPACES}@@"
                                                    :rem 72
310 PRINTTAB(9)"@@@@"SPC(16)"@@@{2 SPACES}@@@":POK
    E7953+CD,6:POKE7956+CD,6                        :rem 213
320 FORI=1TO34:POKE7953,42:POKE7956,42:FORT=1TO200
    :NEXTT                                          :rem 203
330 POKE7953,32:POKE7956,32:FORT=1TO200:NEXTT:NEXT
    I:SYS830:RETURN                                 :rem 152
340 PRINTCHR$(147):FORI=768TO8164STEP22:POKEI+CD,
    5:POKEI,0:NEXT:RETURN                           :rem 10
350 FORI=830TO996:READJ:POKEI,J:NEXT                :rem 44
360 DATA120,169,128,133,0,169,29,133,1,76,83,3,120
    ,169,165,133,0,169,3,133,1,169,5,141    :rem 182
370 DATA60,3,169,6,141,61,3,169,105,141,20,3,169,3
    ,141,21,3,88,96,72,152,72,206,61,3      :rem 78
380 DATA173,61,3,201,3,16,37,201,0,240,6,110,14,14
    4,76,155,3,172,60,3,200                         :rem 16
390 DATA177,0,141,61,3,200,177,0,201,1,240,17,141,
    12,144,169,15,141,14,144,140,60,3,104 :rem 187
400 DATA168,104,76,191,234,160,255,208,243  :rem 73
410 DATA30,215,30,219,30,223,30,215,30,219,30,223,
    30,215,30,223,30,225,60,228,30,223      :rem 32
420 DATA30,225,60,228,15,228,15,231,15,228,15,225,
    30,223,30,215,15,228,15,231,15,228      :rem 62
430 DATA15,225,30,223,30,215,30,215,30,201,60,215,
    30,215,30,201,60,215,1,1                        :rem 41
440 FORI=7168TO7529:POKEI,PEEK(I+25600):NEXT
                                                    :rem 147
450 FORI=1TO7:READJ:FORX=JTOJ+7:READY:POKEX,Y:NEXT
    :NEXT                                           :rem 231
460 DATA7168,255,255,255,255,255,255,255,255
                                                    :rem 202
470 DATA7384,0,129,90,102,126,90,24,36      :rem 132
480 DATA7392,0,24,0,255,126,90,24,0         :rem 229
```

```
490 DATA7400,0,0,0,0,0,60,102,195          :rem 106
500 DATA7408,0,0,0,24,60,36,102,66         :rem 166
510 DATA7416,0,0,56,68,170,170,170,170     :rem 123
520 DATA7440,60,126,90,255,219,102,60,102  :rem 12
530 FORI=7552TO7633:READJ:POKEI,J:NEXT:SYS830
                                           :rem 99
540 DATA45,217,15,221,30,225,30,225,22,221,7,217,2
    2,221,7,225,30,217,30,203,45,217       :rem 208
550 DATA15,221,30,225,30,225,22,221,7,217,22,221,7
    ,225,60,217,22,203,7,203,22,217        :rem 153
560 DATA7,215,22,217,7,225,30,229,22,229,7,232,22,
    229,7,227,22,225,7,221,30,217,22,229   :rem 175
570 DATA7,229,30,229,22,151,7,151,30,151,120,229,1
    ,1:RETURN                              :rem 53
```

Program 2. Hatch It, 64 Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 PRINT"{CLR}"                           :rem 245
110 POKE56,48:CLR                          :rem 223
120 SC=1024:EC=1984:CS=53281:CB=53280:CV=54272:EM=
    56295:JS=56320                         :rem 76
130 NJ=0:N=1:E=8:S=2:WE=4:NE=16:NW=16:SE=16:SW=16:
    FB=16:ES=2023                          :rem 244
132 POKECS,13:POKECB,0:PRINT"{BLK}{5 DOWN}
    {12 RIGHT}{RVS}H A T C H I T{OFF}"     :rem 42
134 PRINT "{3 DOWN}{RIGHT}PLEASE WAIT 70 SECONDS W
    HILE WE"                               :rem 51
135 PRINT"{3 DOWN}{5 RIGHT}REDEFINE THE CHARACTER
    {SPACE}SET."                           :rem 248
140 DIMH%(3):H%(0)=1:H%(1)=-1:H%(2)=22:H%(3)=22:W=
    -1                                     :rem 125
150 POKE56334,PEEK(56334)AND254            :rem 223
160 POKE1,PEEK(1)AND251                    :rem 53
170 FORI=0TO4095:POKEI+12288,PEEK(I+53248):NEXT
                                           :rem 33
180 POKE1,PEEK(1)OR4                       :rem 161
190 POKE56334,PEEK(56334)OR1               :rem 71
200 PRINT"{CLR}"TAB(51)CHR$(18)"HATCHIT!":PRINT:IN
    PUT"NAME";N$:PRINT                     :rem 19
205 PRINT "HI, "N$"!"                      :rem 252
210 PRINT:INPUT"J=JOY, K=KEYS";JK$:JK=1-(JK$="J"):
    PRINT:PRINT"PLEASE WAIT...."           :rem 67
220 FORI=1TO7:READJ:FORX=JTOJ+7:READY:POKEX,Y:NEXT
    :NEXT                                  :rem 226
230 POKE53272,29                           :rem 94
240 POKECS,0:POKECB,5                      :rem 6
250 GOSUB600:FORI=SC+42TO1946STEP40:Z=INT(RND(1)*3
    7):POKEI+Z+CV,6:POKEI+Z,42             :rem 191
```

```
260 NEXT:GOSUB790:SYS49152                          :rem 111
270 NP=1444+(PEEK(1444)<>32):GOSUB340               :rem 48
280 ONJKGOSUB350,680:NP=OP+DP:PK=PEEK(NP):IFNP<102
    5ORNP>2022ORPK=0THEN280                         :rem 103
290 IFPK=32THEN POKEOP,32:GOSUB340:GOTO280          :rem 83
300 OL=NP:OC=34:POKEOP,32:GOSUB340                  :rem 99
310 HC=INT(RND(1)*5)+1:H=INT(RND(1)*3)+1:ONHGOSUB4
    50,420,450                                      :rem 146
320 CT=CT+1:IFCT=21THENCT=0:GOSUB520:GOTO250
                                                    :rem 234
330 GOTO280                                         :rem 105
340 POKENP+CV,7:POKENP,34:OP=NP:RETURN              :rem 243
350 GETA$:IFA$=""ORA$=CHR$(17)ORA$=CHR$(29)THEN350
                                                    :rem 80
360 A=ASC(A$):IFA<>44ANDA<>46ANDA<>65ANDA<>90THEN3
    50                                              :rem 63
370 DP=(A=44)-(A=46)+40*(A=65)-40*(A=90):RETURN
                                                     :rem 8
400 IFDP=0THEN380                                   :rem 238
410 FORT=1TO80:NEXT:RETURN                          :rem 220
420 T=INT(RND(1)*200)+10:W=-W:CH=29                 :rem 208
430 NL=OL+W:IFPEEK(NL)=0THENPOKEOL,32:RETURN
                                                    :rem 117
440 CH=CH+(CH=30)-(CH=29):GOSUB500:GOTO430:rem 156
450 HI=H-2:CH=29+2*HI:T=INT(RND(1)*20)+15  :rem 29
460 Z=INT(RND(1)*4):NL=OL+HI*H%(Z):PK=PEEK(NL)
                                                    :rem 253
470 IFNL<1025ORNL>2022THENRETURN                     :rem 22
480 IFPK<>32THEN460                                 :rem 110
490 CH=CH+(CH=28)-(CH=27):GOSUB500:GOTO460:rem 169
500 POKEOL,OC:OC=32:POKENL+CV,HC:POKENL,CH:OL=NL:F
    ORI=1TOT:NEXT:RETURN                            :rem 209
520 GOSUB600:POKE840,1:SYS49152                      :rem 77
522 PRINT"{CLR}":PRINTTAB(2)CHR$(156)"SUPER, "N$"!
    "TAB(255)CHR$(158)"@@@@"                        :rem 218
530 PRINTTAB(14)"@@@@@@"SPC(34)"@ @@ @"SPC(33)"@@@
    @@@@@"SPC(32)"@@ @@ @@"                         :rem 153
540 PRINTSPC(14)"@@{2 SPACES}@@"                     :rem 6
550 PRINTTAB(15)"@@@@"SPC(34)"@@@{2 SPACES}@@@":PO
    KE1399+CV,6:POKE1402+CV,6                       :rem 22
560 FORI=1TO34:POKE1399,42:POKE1402,42:FORT=1TO200
    :NEXTT                                          :rem 187
570 POKE1399,32:POKE1402,32:FORT=1TO200:NEXTT:NEXT
    I                                               :rem 154
590 RETURN                                          :rem 126
600 PRINT"{CLR}":FORI=SC TO EC STEP 40:POKEI+CV,5:
    POKEI,0:NEXT:RETURN                             :rem 122
610 DATA 12288,255,255,255,255,255,255,255,255
                                                    :rem 246
```

```
620 DATA 12504,0,129,90,102,126,90,24,36  :rem 167
630 DATA 12512,0,24,0,255,126,90,24,36     :rem 65
640 DATA 12520,0,0,0,0,0,60,102,195       :rem 150
650 DATA 12528,0,0,0,24,60,36,102,66      :rem 219
660 DATA 12536,0,0,56,68,170,170,170,170  :rem 176
670 DATA 12560,60,126,90,255,219,102,60,102:rem 65
680 D3=15-PEEK(JS)AND15:IF D3=NJ THEN D1=0:D2=0
                                          :rem 116
690 IF D3=N{2 SPACES}THEN D1=0:D2=-1      :rem 140
700 IF D3=NE THEN X=0:Y=0                  :rem 97
710 IF D3=E{2 SPACES}THEN D1=1:D2=0        :rem 79
720 IF D3=SE THEN X=0:Y=0                 :rem 104
730 IF D3=S{2 SPACES}THEN D1=0:D2=1        :rem 95
740 IF D3=SW THEN X=0:Y=0                 :rem 124
750 IF D3=WE THEN D1=-1:D2=0              :rem 215
760 IF D3=NW THEN X=0:Y=0                 :rem 121
770 DP=D1+40*D2:IFDP=0THEN680             :rem 170
780 RETURN                                :rem 127
790 I=49152                                :rem 43
800 CK=0                                  :rem 147
802 READ A:IF A=256 THEN 1460             :rem 213
805 CK=CK+A                                :rem 98
810 POKE I,A:I=I+1:GOTO 802               :rem 240
820 DATA 169,0,160,24,136,153,0           :rem 236
830 DATA 212,208,250,169,15,141,24        :rem 133
840 DATA 212,169,18,141,5,212,169          :rem 92
850 DATA 241,141,6,212,120,169,91          :rem 84
860 DATA 141,20,3,169,192,141,21           :rem 32
870 DATA 3,88,173,72,3,240,18             :rem 154
880 DATA 169,204,141,64,3,169,192         :rem 103
890 DATA 141,65,3,169,0,141,255           :rem 249
900 DATA 207,76,75,192,169,75,141         :rem 107
910 DATA 64,3,169,193,141,65,3            :rem 202
920 DATA 169,0,141,255,207,173,64          :rem 93
930 DATA 3,133,253,173,65,3,133           :rem 242
940 DATA 254,169,5,141,254,207,96         :rem 104
950 DATA 174,62,3,173,253,207,208          :rem 95
960 DATA 60,169,5,141,254,207,173          :rem 97
970 DATA 255,207,208,26,160,0,177          :rem 95
980 DATA 253,201,255,240,74,141,0          :rem 83
990 DATA 212,200,177,253,141,1,212        :rem 128
1000 DATA 169,17,141,4,212,200,177        :rem 123
1010 DATA 253,170,202,240,11,169,1        :rem 115
1020 DATA 141,255,207,142,62,3,76          :rem 79
1030 DATA 49,234,169,16,141,4,212          :rem 82
1040 DATA 169,1,141,253,207,206,254       :rem 180
1050 DATA 207,240,3,76,49,234,169          :rem 93
1060 DATA 0,141,253,207,141,255,207       :rem 173
1070 DATA 24,165,253,105,3,133,253        :rem 128
```

```
1080 DATA 169,0,101,254,133,254,76          :rem 135
1090 DATA 49,234,173,64,3,133,253            :rem 91
1100 DATA 173,65,3,133,254,76,49,234        :rem 238
1110 REM MUSIC DATA                          :rem 66
1120 DATA 0,0,20,30,25,30                   :rem 169
1130 DATA 49,28,10,165,31,20,165             :rem 29
1140 DATA 31,20,49,28,10,30,25              :rem 176
1150 DATA 10,49,28,10,165,31,10             :rem 227
1160 DATA 30,25,20,209,18,20,30             :rem 223
1170 DATA 25,30,49,28,10,165,31             :rem 237
1180 DATA 20,165,31,20,49,28,10             :rem 232
1190 DATA 30,25,10,49,28,10,165             :rem 236
1200 DATA 31,10,30,25,40,0,0                 :rem 56
1210 DATA 10,30,25,10,49,28,10              :rem 170
1220 DATA 165,31,10,135,33,10,162            :rem 65
1230 DATA 37,10,162,37,10,162,37             :rem 26
1240 DATA 20,162,37,10,62,42,10             :rem 222
1250 DATA 162,37,10,135,33,10,165            :rem 74
1260 DATA 31,10,48,28,10,30,25              :rem 177
1270 DATA 20,162,37,10,162,37,10             :rem 22
1280 DATA 162,37,20,209,18,10,209            :rem 82
1290 DATA 18,10,209,18,20,162,37             :rem 33
1300 DATA 80,255,0,0,10,30,25               :rem 117
1310 DATA 20,49,28,20,165,31,20             :rem 228
1320 DATA 30,25,20,30,25,20,49              :rem 173
1330 DATA 28,20,165,31,20,30,25             :rem 225
1340 DATA 20,165,31,20,135,33,20             :rem 15
1350 DATA 162,37,40,0,0,5,165               :rem 131
1360 DATA 31,20,135,33,20,162,37             :rem 22
1370 DATA 40,0,0,5,167,37,10                 :rem 79
1380 DATA 62,42,10,167,37,10,135             :rem 31
1390 DATA 33,10,165,31,20,30,25             :rem 226
1400 DATA 20,167,37,10,62,42,10             :rem 225
1410 DATA 167,37,10,135,33,10,165            :rem 77
1420 DATA 31,20,30,25,20,30,25              :rem 165
1430 DATA 20,209,18,20,30,25,40             :rem 224
1440 DATA 0,0,10,30,25,20,209               :rem 115
1450 DATA 18,20,30,25,40,255,256             :rem 30
1460 IF CK=39469 THEN RETURN                 :rem 67
1470 PRINT "{CLR}{WHT}ERROR IN DATA LINES 820 TO 1
     450":STOP                              :rem 114
```

Puzzle Solver

Steve Gibson

Hidden word puzzles have always been popular educational tools, but there are usually two or three words that you just can't find. "Puzzle Solver" lets your Commodore computer track them down; it will run on the VIC with at least 3K expansion or on the Commodore 64.

If you enjoy solving puzzles, you're probably familiar with the so-called "word find" puzzles commonly found in newspapers and magazines. They consist of a square matrix of letters in which several words are hidden. The words may be present in any configuration, including up, down, left, right, or diagonally. In a large matrix, words can be difficult to find, particularly those that are leftward, upward, or diagonal.

"Puzzle Solver" finds such words. It locates them after you enter the matrix and word list. Matrix size is limited to 50 x 50 letters (line 100). The program will run on any Commodore computer with sufficient memory; it occupies 5048 bytes of memory, without variables.

Puzzle Solver makes good use of the string-handling capabilities of BASIC. The program works by searching through the matrix for the first letter of a string (a word in the word list), using lines 188-210. When found, the program determines if there is enough room in the matrix for the string in the "northern" (upward) direction without going beyond the top boundary of the matrix (lines 226-264). If there is, the next letter of the string is compared to the next letter in the matrix in the northern direction. This continues until the entire string is found in the northern direction or until a mismatch between string and matrix occurs. When a mismatch occurs, a similar search is started in the northeastern direction. Continued mismatches cause the program to ultimately look in all eight directions (N, NE, E, SE, S, SW, W, NW) until the string is found (lines 268-344). If it isn't, the program locates the next letter in the matrix that matches the first letter in the string, and the process begins all over again.

Once all strings have been searched for, the results can either be listed to the screen or printed. If the screen option is chosen, the matrix will appear on the screen with words in the list sequentially highlighted in reverse video (lines 396-454). If the matrix contains more columns than your screen, the program will

give you the option of listing the X-Y coordinates of the starting and ending letters of the words in the word list. These same coordinates will be printed when the PRINT option is chosen.

When running the program, you are first asked how many screen columns to use for your computer. Answer with 22 for the VIC or 40 for the 64. The next question concerns the size of the matrix; answer with the number of letters on a side.

The program then asks you to input the rows of the matrix (that is, the letters in the horizontal direction). It is *very* important that you enter the *bottom* row first. Do not start with the top row.

You'll be asked the number of hidden strings; then, following the prompts, type the strings in. You'll have a chance to correct errors. Finally, after all questions are answered, the program will search for the strings in the matrix. It will keep you updated on which string is being searched for; when it is through, the output options will be presented.

The printer output option (lines 456-482) is set up for a standard Commodore printer (device 4). You may have to modify these lines if you have a non-Commodore printer.

Puzzle Solver for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 DIMXB(50),YB(50),XE(50),YE(50),ST$(50),R$(50)
                                          :rem 60
102 GOTO112                               :rem 96
104 REM**********SUBROUTINES**********     :rem 36
106 PRINTCHR$(18)MID$(R$(CT),KK,1)CHR$(146);:RETUR
    N                                     :rem 237
108 PRINTMID$(R$(CT),KK,1);:RETURN        :rem 71
110 REM**********END SUBROUTINES*****      :rem 38
112 PRINTCHR$(147)"INPUT SCREEN COLUMNS ON YOUR CO
    MPUTER - EG. 22 OR 40"                :rem 111
114 INPUTSS                               :rem 204
116 REM**********INPUT ROUTINE*******      :rem 252
118 FORCT=1TO50:XB(CT)=0:YB(CT)=0:XE(CT)=0:YE(CT)=
    0:NEXTCT:PRINTCHR$(147)               :rem 255
120 INPUT"MATRIX SIZE";MS                  :rem 82
122 FORCT=1TOMS                           :rem 196
124 PRINT"INPUT ROW "CT" COUNTING FROM BOTTOM OF M
    ATRIX : "                             :rem 223
126 INPUTR$(CT)                           :rem 135
128 IFLEN(R$(CT))<>MSTHENPRINT"IMPROPER ENTRY RE-"
    ;:GOTO124                             :rem 219
130 NEXTCT                                :rem 106
132 PRINT"DO YOU WANT TO CHANGE A ROW - Y/N"
                                          :rem 182
```

```
134 GETYN$:IFYN$=""THEN134                          :rem 31
136 IFYN$<>"N"ANDYN$<>"Y"THEN132                    :rem 122
138 IFYN$="N"THEN148                                :rem 145
140 INPUT"WHICH ROW";RN                             :rem 175
142 PRINT"INPUT ROW "RN                             :rem 144
144 INPUTR$(RN)                                     :rem 144
146 GOTO132                                         :rem 106
148 PRINT"{CLR}HOW MANY STRINGS TO FIND":INPUTNS
                                                    :rem 125
150 FORCT=1TONS                                     :rem 198
152 PRINT"INPUT STRING "CT                          :rem 103
154 INPUTST$(CT)                                    :rem 221
156 TS$=""                                          :rem 232
158 FORCO=1TOLEN(ST$(CT))                           :rem 11
160 IFASC(MID$(ST$(CT),CO,1))=32THEN164             :rem 215
162 TS$=TS$+MID$(ST$(CT),CO,1)                      :rem 180
164 NEXTCO                                          :rem 108
166 ST$(CT)=TS$                                     :rem 88
168 NEXTCT                                          :rem 117
170 PRINT"DO YOU WANT TO CHANGE A STRING - Y/N"
                                                    :rem 151
172 GETYN$:IFYN$=""THEN172                          :rem 35
174 IFYN$<>"N"ANDYN$<>"Y"THEN170                    :rem 126
176 IFYN$="N"THEN186                                :rem 149
178 INPUT"WHICH STRING ";SN                         :rem 154
180 PRINT"INPUT STRING "SN                          :rem 114
182 INPUTST$(SN)                                    :rem 232
184 GOTO170                                         :rem 110
186 REM**********GET MATCHING LETTER IN MATRIX
                                                    :rem 142
188 PRINTCHR$(147):REM CLEAR HOME                   :rem 202
190 SV=0:RO=1:LE=0                                  :rem 49
192 SV=SV+1:PRINT"SEARCHING FOR "ST$(SV)            :rem 210
194 FL$=LEFT$(ST$(SV),1)                            :rem 83
196 LE=LE+1                                         :rem 91
198 RL$=MID$(R$(RO),LE,1)                           :rem 114
200 IFFL$=RL$THEN214                                :rem 156
202 IFLE<>MSTHEN196                                 :rem 157
204 IFRO<>MSTHENLE=0:RO=RO+1:GOTO196                :rem 53
206 PRINT"FIRST LETTER OF STRING "ST$(SV)" NOT FOU
    ND IN ANY ROW"                                  :rem 26
208 IFSV=NSTHENGOTO362                              :rem 179
210 LE=0:RO=0:GOTO192                               :rem 232
212 REM**********MATRIX SEARCH ROUTINE              :rem 206
214 FORTT=1TO8                                      :rem 111
216 FORCT=0TOLEN(ST$(SV))-1                         :rem 122
218 SV$=MID$(ST$(SV),CT+1,1)                        :rem 53
220 IFCT>0THENONTTGOTO270,280,290,300,310,320,330,
    340                                             :rem 192
```

```
222 ONTTGOTO226,230,236,240,246,250,256,260:rem 16
224 REM CHECK IF LENGTH OF STRING WILL EXCEED MATR
    IX BOUNDRIES                                :rem 253
226 IFLEN(ST$(SV))>MS+1-ROTHENNEXTTT             :rem 60
228 GOTO270                                     :rem 110
230 IFRO>LEANDLEN(ST$(SV))>MS+1-ROTHENNEXTTT
                                                :rem 122
232 IFLE>=ROANDLEN(ST$(SV))>MS+1-LETHENNEXTTT
                                                :rem 169
234 GOTO280                                     :rem 108
236 IFLEN(ST$(SV))>MS+1-LETHENNEXTTT             :rem 45
238 GOTO290                                     :rem 113
240 IFLE+RO<=MS+1ANDLEN(ST$(SV))>ROTHENNEXTTT
                                                :rem 180
242 IFLE+RO>MSANDLEN(ST$(SV))>MS-LE+1THENNEXTTT
                                                 :rem 56
244 GOTO300                                     :rem 102
246 IFLEN(ST$(SV))>ROTHENNEXTTT                  :rem 21
248 GOTO310                                     :rem 107
250 IFRO>LEANDLEN(ST$(SV))>LETHENNEXTTT          :rem 67
252 IFRO<=LEANDLEN(ST$(SV))>ROTHENNEXTTT        :rem 144
254 GOTO320                                     :rem 105
256 IFLEN(ST$(SV))>LETHENNEXTTT                   :rem 6
258 GOTO330                                     :rem 110
260 IFLE+RO<=MS+1ANDLEN(ST$(SV))>LETHEN354      :rem 91
262 IFLE+RO>MS+1ANDLEN(ST$(SV))>MS-RO+1THEN354
                                                :rem 91
264 GOTO340                                     :rem 108
266 REM CHECK ALL DIRECTIONS FOR STRING         :rem 107
268 REM LOOK NORTH                               :rem 68
270 IFSV$<>MID$(R$(RO+CT),LE,1)THENNEXTTT        :rem 24
272 NEXTCT                                      :rem 113
274 XE(SV)=LE:YE(SV)=RO+LEN(ST$(SV))-1           :rem 48
276 GOTO348                                     :rem 119
278 REM LOOK NORTH EAST                         :rem 114
280 IFSV$<>MID$(R$(RO+CT),LE+CT,1)THENNEXTTT
                                                :rem 219
282 NEXTCT                                      :rem 114
284 XE(SV)=LE+LEN(ST$(SV))-1:YE(SV)=RO+LEN(ST$(SV)
    )-1                                         :rem 175
286 GOTO348                                     :rem 120
288 REM LOOK EAST                               :rem 232
290 IFSV$<>MID$(R$(RO),LE+CT,1)THENNEXTTT        :rem 26
292 NEXTCT                                      :rem 115
294 XE(SV)=LE+LEN(ST$(SV))-1:YE(SV)=RO           :rem 50
296 GOTO348                                     :rem 121
298 REM LOOK SOUTH EAST                         :rem 124
300 IFSV$<>MID$(R$(RO-CT),LE+CT,1)THENNEXTTT
                                                :rem 214
```

```
302 NEXTCT                                          :rem 107
304 XE(SV)=LE+LEN(ST$(SV))-1:YE(SV)=RO-LEN(ST$(SV)
    )+1                                             :rem 168
306 GOTO348                                         :rem 113
308 REM LOOK SOUTH                                   :rem 71
310 IFSV$<>MID$(R$(RO-CT),LE,1)THENNEXTTT            :rem 21
312 NEXTCT                                          :rem 108
314 XE(SV)=LE:YE(SV)=RO-LEN(ST$(SV))+1               :rem 43
316 GOTO348                                         :rem 114
318 REM LOOK SOUTH WEST                             :rem 139
320 IFSV$<>MID$(R$(RO-CT),LE-CT,1)THENNEXTTT
                                                    :rem 218
322 NEXTCT                                          :rem 109
324 XE(SV)=LE-LEN(ST$(SV))+1:YE(SV)=RO-LEN(ST$(SV)
    )+1                                             :rem 170
326 GOTO348                                         :rem 115
328 REM LOOK WEST                                   :rem 249
330 IFSV$<>MID$(R$(RO),LE-CT,1)THENNEXTTT            :rem 23
332 NEXTCT                                          :rem 110
334 XE(SV)=LE-LEN(ST$(SV))+1:YE(SV)=RO               :rem 45
336 GOTO348                                         :rem 116
338 REM LOOK NORTH WEST                             :rem 133
340 IFSV$<>MID$(R$(RO+CT),LE-CT,1)THEN354           :rem 143
342 NEXTCT                                          :rem 111
344 XE(SV)=LE-LEN(ST$(SV))+1:YE(SV)=RO+LEN(ST$(SV)
    )-1                                             :rem 172
346 REM**********END MATRIX SEARCH ROUTINE:rem 173
348 XB(SV)=LE:YB(SV)=RO:PRINT"FOUND "ST$(SV):PRINT
                                                    :rem 193
350 IFSV=NSTHEN362                                  :rem 120
352 LE=0:RO=1:GOTO192                               :rem 240
354 IFLE=MSANDRO=MSTHENPRINT"NOT FOUND":IFSV=NSTHE
    N362                                            :rem 113
356 IFLE=MSANDRO=MSTHENLE=0:RO=1:GOTO192            :rem 113
358 IFLE=MSTHENLE=0:RO=RO+1:GOTO196                 :rem 242
360 GOTO196                                         :rem 114
362 PRINT:PRINT"OUTPUT TO PRINTER OR SCREEN OR NEX
    T PUZZLE OR END - P/S/N/E ?"                    :rem 189
364 GETSP$:IFSP$=""THEN364                           :rem 33
366 IFSP$<>"S"ANDSP$<>"P"ANDSP$<>"N"ANDSP$<>"E"THE
    N362                                            :rem 187
368 IFSP$="P"THENCD=1:GOTO458                          :rem 0
370 IFSP$="N"THEN118                                :rem 136
372 IFSP$="E"THENEND                                :rem 190
374 IFMS<=SSTHEN386                                 :rem 188
376 PRINT"MATRIX OVERSIZE FOR SCREEN"               :rem 100
378 PRINT"WOULD YOU RATHER SEE THE STRING COORDINA
    TES - Y/N ?"                                    :rem 211
380 GETYN$:IFYN$=""THEN380                           :rem 37
```

```
382 IFYN$<>"Y"ANDYN$<>"N"THEN376          :rem 135
384 IFYN$="Y"THENPRINTCHR$(147):CD=Ø:GOTO458
                                          :rem 191
386 FORTT=1TONS                           :rem 226
388 PRINT:PRINT"PRESS ANY KEY TO SEE "CHR$(18)ST$(
    TT)CHR$(146)                          :rem 133
390 GETAK$:IFAK$=""THEN390                :rem 241
392 IFXB(TT)=ØTHENPRINTST$(TT)" NOT FOUND":GOTO452
                                          :rem 108
394 PRINTCHR$(147)                         :rem 27
395 REM**********ROUTINE TO PRINT REVERSE VIDEO
                                           :rem 18
396 FORCT=MST01STEP-1                     :rem 107
398 IFYB(TT)<CTANDYE(TT)<CTTHENPRINTR$(CT);:GOTO45
    Ø                                      :rem 56
400 IFYB(TT)>CTANDYE(TT)>CTTHENPRINTR$(CT);:GOTO45
    Ø                                      :rem 44
402 IFYB(TT)=YE(TT)THEN430                 :rem 83
404 IFXB(TT)=XE(TT)THEN442                 :rem 86
406 FORKK=1TOMS                           :rem 200
408 IFXB(TT)>XE(TT)THEN418                 :rem 94
410 IFKK>XE(TT)ORKK<XB(TT)THENGOSUB108:GOTO426
                                          :rem 235
412 IFYB(TT)<YE(TT)ANDKK=XE(TT)-(YE(TT)-CT)THENGOS
    UB106:GOTO426                         :rem 247
414 IFYB(TT)>YE(TT)ANDKK=XE(TT)+(YE(TT)-CT)THENGOS
    UB106:GOTO426                         :rem 249
416 GOSUB108:GOTO426                      :rem 195
418 IFKK>XB(TT)ORKK<XE(TT)THENGOSUB108:GOTO426
                                          :rem 243
420 IFYB(TT)<YE(TT)ANDKK=XB(TT)+(YB(TT)-CT)THENGOS
    UB106:GOTO426                         :rem 238
422 IFYB(TT)>YE(TT)ANDKK=XB(TT)-(YB(TT)-CT)THENGOS
    UB106:GOTO426                         :rem 244
424 GOSUB108                              :rem 179
426 NEXTKK                                :rem 113
428 GOTO450                               :rem 112
430 FORKK=1TOMS                           :rem 197
432 IFXB(TT)>XE(TT)AND(KK>XB(TT)ORKK<XE(TT))THENGO
    SUB108:GOTO438                        :rem 125
434 IFXB(TT)<XE(TT)AND(KK<XB(TT)ORKK>XE(TT))THENGO
    SUB108:GOTO438                        :rem 125
436 GOSUB106                              :rem 180
438 NEXTKK                                :rem 116
440 GOTO450                               :rem 106
442 FORKK=1TOMS                           :rem 200
444 IFKK=XB(TT)THENGOSUB106:GOTO448       :rem 234
446 GOSUB108                              :rem 183
448 NEXTKK                                :rem 117
```

```
450 PRINT:NEXTCT                                    :rem 54
452 NEXTTT                                          :rem 130
454 GOTO362                                         :rem 113
456 REM**********PRINT ROUTINE******                 :rem 214
458 IFCD=1THENOPEN4,4:CMD4                          :rem 92
460 FORCT=1TONS                                     :rem 202
462 PRINT:PRINT"STRING - "ST$(CT)                   :rem 235
464 IFXB(CT)=0THENPRINT"STRING NOT FOUND":GOTO478
                                                    :rem 118
466 PRINT:PRINT"BEGINNING LETTER"                   :rem 153
468 PRINT"X,Y COORDINATES = "XB(CT)","YB(CT)
                                                    :rem 61
470 PRINT:PRINT"ENDING LETTER"                      :rem 184
472 PRINT"X,Y COORDINATES = "XE(CT)","YE(CT)
                                                    :rem 62
474 IFCD=K0THENPRINT:PRINT"PRESS ANY KEY TO CONTIN
    UE"                                             :rem 154
476 IFCD=0THENGETAK$:IFAK$=""THEN476                :rem 173
477 IFNS=1THEN480                                   :rem 11
478 NEXTCT                                          :rem 121
480 IFCD=1THENPRINT#4:CLOSE4                        :rem 23
482 GOTO362                                         :rem 114
```

Chapter 3

Applications

File Cabinet

Mike Webster

Everyone has something to catalog: a record collection, a library, a stack of twelve dozen program disks. The following program lets your VIC or 64 handle the filing chores for you.

"File Cabinet" is designed as a general-purpose program for data storage and retrieval. It can be easily modified to suit your needs. As presented here, the program features six functions — ADD, REVIEW, FIND, CHANGE, DELETE, and SAVE — and the amount of data that can be stored is dependent only on the length of each entry and on the amount of free memory available.

The program provides you with five different INPUTs per entry. For the sake of clarity, each of the INPUTs will be called a *category,* and each group of five categories will be called a *page*.

You can use File Cabinet to catalog your record collection, your program files, or anything else you might want to recall or search by category. One good application might be for cataloging your personal library. In that case, a typical entry page could include a book's title, its author or publisher, the type of book, and two notations describing the material covered in the book. For example, one page might look like this:

COMPUTE!'s FIRST BOOK OF VIC
COMPUTE!
REFERENCE
PROGRAMMING
GAMES

Another entry could resemble this one:

MACHINE LANGUAGE FOR BEGINNERS
RICHARD MANSFIELD
REFERENCE
PROGRAMMING
MACHINE LANGUAGE

If you want to make a null entry for a category, use an asterisk. That eliminates the possibility of any carryover from the previous page. You can use up to 42 characters per page entry; any more than that will not store in the DATA statements.

To see how File Cabinet works, type in the two examples given above. You can then FIND either one by entering any of the appropriate categories. For instance, telling the computer to FIND

games will turn up *First Book of VIC*. Similarly, telling it to FIND "programming" will call both *First Book of VIC* and *Machine Language for Beginners* to the screen. They will be displayed one at a time; to bring up the next one, just hit RETURN. Using the same approach, you could locate every book in your library that was written by Richard Mansfield or every book that dealt with the VIC.

The crunched program occupies almost 1900 bytes of RAM. A page entry containing 25 characters fills about 36 bytes, and a page entry with the maximum of 42 characters fills about 53 bytes. Thus, an unexpanded VIC will hold about fifty-one 25-character pages or thirty-two 42-character pages. With 3K expansion, the VIC can handle one hundred thirty-three 25-character pages or ninety 42-character pages. If you have the 8K expander, the capacity increases even further to two hundred seventy-five 25-character pages and one hundred eighty-seven 42-character pages. The Commodore 64, of course, will hold substantially more.

To customize the program for your own needs, you may wish to change the number of categories per page. In the VIC version you can do this by modifying seven areas of the program: INPUT (lines 75-95); DATA (line 120); FIND (lines 155-173); CHANGE (lines 210-230); READ (line 505); SCREEN DISPLAY (lines 520-540); and DATA (line 60000).

Be sure to crunch the program lines when you type this program in. Do not leave unneeded spaces, and abbreviate commands (? for PRINT, for example) wherever possible. Otherwise, some lines may not fit.

Program 1. File Cabinet, VIC Version

Refer to "The Automatic Proofreader" (Appendix I) before typing this program in.

```
7 POKE36879,47:PRINT"{YEL}"                    :rem 183
10 PRINT"{CLR}":PRINTTAB(31)"{RVS}MENU         :rem 129
15 PRINTTAB(47)"{RVS}A";"{OFF}DD NEW ENTRIES"SPC(5
   1)"{RVS}R";"{OFF}EVIEW ALL ENTRIES           :rem 97
20 PRINTTAB(47)"{RVS}F";"{OFF}IND AN ENTRY"SPC(53)
   "{RVS}C";"{OFF}HANGE AN ENTRY"              :rem 248
25 PRINTSPC(47)"{RVS}D";"{OFF}ELETE AN ENTRY"SPC(2
   2)TAB(47)"{RVS}S";"{OFF}AVE PROGRAM         :rem 139
30 GETA$:IFA$=""THEN30                         :rem 233
35 IFA$="A"THEN70                              :rem 180
40 IFA$="R"THEN130                             :rem 238
45 IFA$="F"THEN150                             :rem 233
50 IFA$="C"THEN190                             :rem 230
55 IFA$="D"THEN240                             :rem 232
60 IFA$="S"THEN360                             :rem 246
```

```
65 GOTO3Ø                                          :rem 7
7Ø GOSUB5ØØ:LN=256*PEEK(64)+PEEK(63)-1           :rem 156
75 PRINT"{CLR}":PRINTSPC(22):INPUTA$              :rem 97
8Ø PRINTSPC(22):INPUTB$                          :rem 192
85 PRINTSPC(22):INPUTC$                          :rem 198
9Ø PRINTSPC(22):INPUTD$                          :rem 195
95 PRINTSPC(22):INPUTE$                          :rem 2Ø1
1ØØ GOSUB575:PRINT"{CLR}"                         :rem 8Ø
1Ø5 PRINTSPC(178)"DEPRESS {RVS}RETURN{OFF} KEY":PR
    INTSPC(24)"TO SAVE INFORMATION"              :rem 129
11Ø PRINT"[22 I]"                                 :rem 79
115 PRINTSPC(24)"DEPRESS {RVS}RETURN{OFF} KEY":PRI
    NTSPC(27)"TO CONTINUE"                        :rem 59
12Ø PRINT"{CLR}{3 DOWN}"LN"DATA"A$","B$","C$","D$"
    ,"E$","LN:PRINT"{WHT} RUN1Ø{BLU}"            :rem 116
125 PRINT"{7 UP}":END                            :rem 113
13Ø GOSUB5ØØ                                     :rem 169
135 IFA$="END"THENGOSUB59Ø:GOTO1Ø                  :rem 6
14Ø IFA$<>"END"THENGOSUB515:GOSUB5Ø5:GOTO135
                                                 :rem 2ØØ
15Ø PRINT"{CLR}FIND":PRINT"[4 U]":INPUTFF$:GOSUB5Ø
    Ø                                            :rem 2Ø7
155 IFFF$=A$THENGOSUB515                         :rem 198
16Ø IFFF$=B$THENGOSUB515                         :rem 195
165 IFFF$=C$THENGOSUB515                         :rem 2Ø1
17Ø IFFF$=D$THENGOSUB515                         :rem 198
173 IFFF$=E$THENGOSUB515                         :rem 2Ø2
175 IFA$="END"THENGOSUB59Ø                         :rem 54
18Ø GOSUB5Ø5:GOTO155                             :rem 193
19Ø PRINT"{CLR}ENTRY # TO BE CHANGED":INPUTCC:GOSU
    B5ØØ                                         :rem 1Ø2
195 IFLN=ØTHENGOSUB59Ø                           :rem 13Ø
2ØØ IFCC<>LNTHENGOSUB5Ø5:GOTO195                   :rem 22
2Ø5 IFCC=LNTHENPRINT"{CLR}"                        :rem 22
21Ø PRINTA$:INPUTA$                              :rem 18Ø
215 PRINTB$:INPUTB$                              :rem 187
22Ø PRINTC$:INPUTC$                              :rem 185
225 PRINTD$:INPUTD$                              :rem 192
23Ø PRINTE$:INPUTE$                              :rem 19Ø
235 GOTO1ØØ                                      :rem 1ØØ
24Ø PRINT"{CLR}ENTER ITEM NUMBER":PRINTSPC(22)"TO
    {SPACE}BE DELETED"                            :rem 55
245 PRINT:INPUTI:LN=I                            :rem 149
25Ø PRINT"{CLR}":FORI=1TO8:PRINT:NEXT            :rem 238
252 PRINTTAB(2)"DEPRESS {RVS}RETURN{OFF} KEY":PRIN
    TSPC(25)"TO DELETE ENTRY"                    :rem 216
255 PRINT"[22 O]"                                 :rem 83
26Ø PRINTSPC(24)"DEPRESS {RVS}RETURN{OFF} KEY":PRI
    NTSPC(27)"TO CONTINUE                         :rem 26
```

```
270 PRINTCHR$(19)CHR$(17)CHR$(17)CHR$(17)LN:PRINT"
    {WHT} RUN10{BLU}"                          :rem 47
275 PRINT"{5 UP}"                              :rem 68
280 END                                        :rem 113
360 POKE36879,25:PRINT"{CLR}{BLK}"SPC(150)"
    {7 RIGHT}{RVS}T{OFF}APE{2 SPACES}OR{2 SPACES}
    {RVS}D{OFF}ISK?"                           :rem 243
361 GOTO1000                                   :rem 148
365 END                                        :rem 117
500 RESTORE                                    :rem 185
505 READA$,B$,C$,D$,E$,LN                      :rem 47
510 RETURN                                     :rem 118
515 PRINT"{CLR}ENTRY #"LN                      :rem 78
520 PRINTSPC(22)A$                             :rem 36
525 PRINTSPC(22)B$                             :rem 42
530 PRINTSPC(22)C$                             :rem 39
535 PRINTSPC(22)D$                             :rem 45
540 PRINTSPC(22)E$                             :rem 42
545 PRINTSPC(22)"{RVS}PRESS 'C' TO CONTINUE":PRINT
    SPC(22)"{RVS}PRESS 'M' FOR MENU"           :rem 62
550 GETCM$:IFCM$=""THEN550                     :rem 245
555 IFCM$="C"THEN570                           :rem 113
560 IFCM$="M"THEN10                            :rem 60
565 GOTO550                                    :rem 115
570 RETURN                                     :rem 124
575 PRINTTAB(44)"{RVS}PRESS KEY TO CONTINUE"
                                               :rem 146
580 GETZ$:IFZ$=""THEN580                       :rem 143
585 RETURN                                     :rem 130
590 PRINTCHR$(147):FORI=1TO4:PRINTCHR$(17):NEXT:PR
    INT"{3 SPACES}ENTRY NOT FOUND"             :rem 204
595 GOSUB575:GOTO10:RETURN                     :rem 178
1000 GETR$:IFR$=""THEN1000                     :rem 199
1010 IFR$=CHR$(84)THEN1100                     :rem 179
1015 IFR$=CHR$(68)THEN1200                     :rem 187
1020 GOTO 360                                  :rem 149
1100 PRINT"{CLR}"SPC(134)TAB(2);               :rem 138
1110 INPUT"FILENAME";F$                        :rem 125
1120 PRINT"{CLR}"SPC(110)                      :rem 241
1125 PRINT"PRESS {RVS}RETURN{OFF}"SPC(208):rem 124
1130 PRINT"SA";:PRINTCHR$(34);:PRINTF$;:PRINTCHR$(
     34);                                      :rem 199
1140 PRINT"{HOME}{12 DOWN}":END                :rem 135
1200 PRINT"{CLR}"SPC(134)TAB(2);               :rem 139
1210 INPUT"FILENAME";F$                        :rem 126
1220 PRINT"{CLR}"SPC(110)                      :rem 242
1225 PRINT"PRESS {RVS}RETURN{OFF}"SPC(208):rem 125
1230 PRINT"SA";:PRINTCHR$(34);:PRINTF$;:PRINTCHR$(
     34);",8"                                  :rem 112
```

```
1240 PRINT"{HOME}{12 DOWN}":END                    :rem 136
59998 DATAENTRY2,A,C,D,E, 59998                    :rem 247
59999 DATAENTRY1,A,B,C,D, 59999                    :rem 245
60000 DATAEND,END,END,END,END,0                     :rem 79
```

Program 2. File Cabinet, 64 Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
7 CS=53281:CB=53280                                 :rem 11
9 POKECS,2:POKECB,7                                :rem 173
10 PRINT"{CLR}{YEL}";SPC(120)TAB(14)"{RVS}MENU
                                                    :rem 94
12 PRINTSPC(80);TAB(6)"{RVS}A";"{OFF}DD NEW ENTRIE
   S"SPC(105)"{RVS}R";"{OFF}EVIEW ALL ENTRIES
                                                    :rem 51
14 PRINTSPC(80)TAB(6)"{RVS}F";"{OFF}IND AN ENTRY"S
   PC(107)"{RVS}C";"{OFF}HANGE AN ENTRY"  :rem 149
16 PRINTSPC(80)TAB(6)"{RVS}D";"{OFF}ELETE AN ENTRY
   "SPC(105)"{RVS}S";"{OFF}AVE PROGRAM"   :rem 167
30 GETA$:IFA$=""THEN30                             :rem 233
35 IFA$="A"THEN70                                  :rem 180
40 IFA$="R"THEN130                                 :rem 238
45 IFA$="F"THEN150                                 :rem 233
50 IFA$="C"THEN190                                 :rem 230
55 IFA$="D"THEN240                                 :rem 232
60 IFA$="S"THEN360                                 :rem 246
65 GOTO30                                            :rem 7
70 GOSUB500:LN=256*PEEK(64)+PEEK(63)-1             :rem 156
75 PRINT"{CLR}":PRINTSPC(40):INPUTA$                :rem 97
80 PRINTSPC(40):INPUTB$                            :rem 192
85 PRINTSPC(40):INPUTC$                            :rem 198
90 PRINTSPC(40):INPUTD$                            :rem 195
95 PRINTSPC(40):INPUTE$                            :rem 201
100 GOSUB575:PRINT"{CLR}"                           :rem 80
105 PRINTSPC(250)"DEPRESS {RVS}RETURN{OFF} KEY":PR
    INTSPC(50)"TO SAVE INFORMATION"                :rem 119
110 PRINTTAB(2)"[35 I]"                          :rem 227
115 PRINTSPC(50)"DEPRESS {RVS}RETURN{OFF} KEY":PRI
    NTSPC(50)"TO CONTINUE"                          :rem 54
120 PRINT"{CLR}{3 DOWN}"LN"DATA"A$","B$","C$","D$"
    ,"E$","LN:PRINT"{WHT} RUN10{BLU}"              :rem 116
125 PRINT"{7 UP}":END                              :rem 113
130 GOSUB500                                       :rem 169
135 IFA$="END"THENGOSUB590:GOTO10                    :rem 6
140 IFA$<>"END"THENGOSUB515:GOSUB505:GOTO135
                                                   :rem 200
150 PRINT"{CLR}FIND":PRINT"[4 U]":INPUTFF$:GOSUB
    500                                            :rem 207
155 IFFF$=A$THENGOSUB515                           :rem 198
```

```
160 IFFF$=B$THENGOSUB515                          :rem 195
165 IFFF$=C$THENGOSUB515                          :rem 201
170 IFFF$=D$THENGOSUB515                          :rem 198
173 IFFF$=E$THENGOSUB515                          :rem 202
175 IFA$="END"THENGOSUB590                         :rem 54
180 GOSUB505:GOTO155                              :rem 193
190 PRINT"{CLR}ENTRY # TO BE CHANGED":INPUTCC:GOSU
    B500                                          :rem 102
195 IFLN=0THENGOSUB590                            :rem 130
200 IFCC<>LNTHENGOSUB505:GOTO195                   :rem 22
205 IFCC=LNTHENPRINT"{CLR}"                        :rem 22
210 PRINTA$:INPUTA$                               :rem 180
215 PRINTB$:INPUTB$                               :rem 187
220 PRINTC$:INPUTC$                               :rem 185
225 PRINTD$:INPUTD$                               :rem 192
230 PRINTE$:INPUTE$                               :rem 190
235 GOTO100                                       :rem 100
240 PRINT"{CLR}ENTER ITEM NUMBER TO BE DELETED"
                                                  :rem 145
245 PRINT:INPUTI:LN=I                             :rem 149
250 PRINT"{CLR}"                                  :rem 251
251 PRINTSPC(250)"DEPRESS {RVS}RETURN{OFF} KEY":PR
    INTSPC(50)"TO DELETE ENTRY"                    :rem 73
255 PRINTTAB(2)"[35 I]"                         :rem 237
260 PRINTSPC(50)"DEPRESS {RVS}RETURN{OFF} KEY":PRI
    NTSPC(50)"TO CONTINUE                          :rem 21
270 PRINTCHR$(19)CHR$(17)CHR$(17)CHR$(17)LN:PRINT"
    {WHT} RUN10{BLU}"                              :rem 47
275 PRINT"{5 UP}"                                  :rem 68
280 END                                           :rem 113
360 POKECB,11:POKECS,15:PRINT"{CLR}{BLK}"SPC(160)"
    {7 RIGHT}{RVS}T{OFF}APE{2 SPACES}OR{2 SPACES}
    {RVS}D{OFF}ISK?"                              :rem 244
361 GOTO1000                                      :rem 148
365 END                                           :rem 117
500 RESTORE                                       :rem 185
505 READA$,B$,C$,D$,E$,LN                          :rem 47
510 RETURN                                        :rem 118
515 PRINT"{CLR}ENTRY #"LN                          :rem 78
520 PRINTSPC(40)A$                                 :rem 36
525 PRINTSPC(40)B$                                 :rem 42
530 PRINTSPC(40)C$                                 :rem 39
535 PRINTSPC(40)D$                                 :rem 45
540 PRINTSPC(40)E$                                 :rem 42
545 PRINTSPC(40)"{RVS}PRESS 'C' TO CONTINUE":PRINT
    SPC(40)"{RVS}PRESS 'M' FOR MENU"               :rem 62
550 GETCM$:IFCM$=""THEN550                        :rem 245
555 IFCM$="C"THEN570                              :rem 113
560 IFCM$="M"THEN10                                :rem 60
```

```
565 GOTO550                                          :rem 115
570 RETURN                                           :rem 124
575 PRINTTAB(10)"{RVS}PRESS KEY TO CONTINUE"
                                                     :rem 139
580 GETZ$:IFZ$=""THEN580                             :rem 143
585 RETURN                                           :rem 130
590 PRINTCHR$(147):FORI=1TO4:PRINTCHR$(17):NEXT
                                                     :rem 194
593 PRINTTAB(10)"{3 SPACES}ENTRY NOT FOUND":GOSUB5
    75:GOTO10                                         :rem 41
595 RETURN                                           :rem 131
1000 GETR$:IFR$=""THEN1000                           :rem 199
1010 IFR$=CHR$(84)THEN1100                           :rem 179
1015 IFR$=CHR$(68)THEN1200                           :rem 187
1020 GOTO 360                                        :rem 149
1100 PRINT"{CLR}"SPC(160)TAB(2);                     :rem 137
1110 INPUT"FILENAME";F$                              :rem 125
1120 PRINT"{CLR}"SPC(120)                            :rem 242
1125 PRINT"PRESS {RVS}RETURN{OFF}"SPC(228):rem 126
1130 PRINT"SA";:PRINTCHR$(34);:PRINTF$;:PRINTCHR$(
     34);                                            :rem 199
1140 PRINT"{HOME}{6 DOWN}":END                        :rem 33
1200 PRINT"{CLR}"SPC(160)TAB(2);                     :rem 138
1210 INPUT"FILENAME";F$                              :rem 126
1220 PRINT"{CLR}"SPC(120)                            :rem 243
1225 PRINT"PRESS {RVS}RETURN{OFF}"SPC(228):rem 127
1230 PRINT"SA";:PRINTCHR$(34);:PRINTF$;:PRINTCHR$(
     34);:PRINT",8"                                  :rem 55
1240 PRINT"{HOME}{6 DOWN}":END                        :rem 34
59998 DATAENTRY2,A,C,D,E, 59998                      :rem 247
59999 DATAENTRY1,A,B,C,D, 59999                      :rem 245
60000 DATAEND,END,END,END,END,0                       :rem 79
```

3-D Clock

Bosco Tsang

What time is it? With this program, and your unexpanded VIC or 64, you may never have to ask again.

One of the more interesting features of a Commodore computer is its internal clock. The computer keeps time in hours, minutes, and seconds, and "3-D Clock" converts that time into an impressive three-dimensional screen display.

The primary display, which shows only hours and minutes, is made up of Commodore graphics characters. For those who are not satisfied without seconds, a smaller clock in the lower right-hand corner of the screen displays seconds too.

To use the 3-D clock, type in and run the program. You'll be asked to enter the correct hour and minute. Then, when you press the f1 key, the screen will display the time. The clock will run until you interrupt the program or turn off your computer.

Program 1. 3-D Clock, VIC Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 DATA"{RVS}{BLK}£{2 SPACES}{DOWN}{3 LEFT}
   {RIGHT} {DOWN}{3 LEFT} {RIGHT} {DOWN}{3 LEFT}
   {RIGHT} {DOWN}{3 LEFT}{2 SPACES}{OFF}£":rem 60
20 DATA"{RVS}£ {DOWN}{LEFT} {DOWN}{LEFT} {DOWN}
   {LEFT} {DOWN}{2 LEFT}£ {OFF}£"          :rem 180
30 DATA"{RVS}£{2 SPACES}{DOWN}{LEFT} {DOWN}
   {3 LEFT}£ {OFF}£{RVS}{DOWN}{3 LEFT} {DOWN}
   {LEFT}{2 SPACES}{OFF}£"                 :rem 217
40 DATA"{RVS}£{2 SPACES}{DOWN}{LEFT} {DOWN}
   {2 LEFT}{2 SPACES}{DOWN}{LEFT} {DOWN}{3 LEFT}£
    {OFF}£"                                :rem 240
50 DATA"{RVS}£{RIGHT}£{DOWN}{3 LEFT} {RIGHT}
   {DOWN}{3 LEFT}{3 SPACES}{DOWN}{LEFT} {DOWN}
   {LEFT}{OFF}£"                           :rem 200
60 DATA"{RVS}£ {OFF}£{RVS}{DOWN}{3 LEFT} {DOWN}
   {LEFT}{3 SPACES}{DOWN}{LEFT} {DOWN}{3 LEFT}£
   {OFF}£"                                 :rem 220
70 DATA"{RVS}£ {OFF}£{RVS}{DOWN}{3 LEFT} {DOWN}
   {LEFT}{3 SPACES}{DOWN}{3 LEFT} {RIGHT} {DOWN}
   {3 LEFT}{2 SPACES}{OFF}£"               :rem 139
80 DATA"{RVS}£{2 SPACES}{DOWN}{LEFT} {DOWN}{LEFT}
    {DOWN}{LEFT} {DOWN}{LEFT}{OFF}£"       :rem 116
```

```
90 DATA"{RVS}£{2 SPACES}{DOWN}{3 LEFT} {RIGHT}
   {DOWN}{3 LEFT}{3 SPACES}{DOWN}{3 LEFT} {RIGHT}
   {SPACE}{DOWN}{3 LEFT}{2 SPACES}{OFF}£" :rem 151
100 DATA"{RVS}£{2 SPACES}{DOWN}{3 LEFT}¯{RIGHT}
    {DOWN}{3 LEFT}{3 SPACES}{DOWN}{LEFT} {DOWN}
    {LEFT}{OFF}£"                          :rem 46
110 DATA"{DOWN}¯{RVS} {2 DOWN}{LEFT} "     :rem 210
120 CLR:DIMA$(9),M$(10):FORT=0TO9:READA$(T):NEXT:R
    EADPU$                                 :rem 124
130 PRINT"{CLR}{BLK}{4 SPACES}STEREO CLOCK"
                                           :rem 198
145 S1$="{HOME}{20 DOWN}{8 RIGHT}":SC$=LEFT$(S1$,8
    )                                      :rem 239
150 J2=60:J3=60:J4=60:J5=60:J6=60:J7=60    :rem 129
260 PRINT"{BLU}ENTER THE PRESENT TIME"      :rem 55
270 INPUT"{DOWN}HOUR";HO:IFHO<0ORHO>12THEN270
                                           :rem 251
275 HO$=RIGHT$(STR$(HO),2):IFHO<10THENHO$="0"+RIGH
    T$(HO$,1)                              :rem 186
280 INPUT"{DOWN}MIN.";ME:IFME<0ORME>59THEN280
                                           :rem 205
285 ME$=RIGHT$(STR$(ME),2):IFME<10THENME$="0"+RIGH
    T$(ME$,1)                              :rem 162
290 PT$=HO$+ME$:PRINT"{DOWN}HIT F1 KEY TO START"
                                           :rem 206
300 GETA$:IFA$<>"{F1}"THEN300              :rem 11
310 TI$=PT$+"00":PRINT"{CLR}":CV=3072O     :rem 211
312 POKE 7866+CV,0:POKE7866,160:POKE7910+CV,0:POKE
    7910,160                               :rem 30
320 T2$=TI$:IFLEFT$(T2$,2)="13"THENTI$="010000":T2
    $=TI$                                  :rem 7
330 T=2:J= VAL(MID$(T2$,1,1)):IF J<>J2 THEN J2=J:G
    OSUB 500                               :rem 139
340 T=6:J=VAL(MID$(T2$,2,1)):IF J<>J3THEN J3=J:GOS
    UB 500                                 :rem 147
350 T=12:J=VAL(MID$(T2$,3,1)):IF J<>J4THENJ4=J:GOS
    UB 500                                 :rem 196
355 T= 16:J=VAL(MID$(T2$,4,1)):IFJ<>J5THENJ5=J:GOS
    UB 500                                 :rem 208
370 GOTO320                                :rem 104
500 PRINTS1$LEFT$(T2$,2)":"MID$(T2$,3,2)" :rem 200
510 PRINTSC$TAB(T)"{3 SPACES}{DOWN}{3 LEFT}
    {3 SPACES}{DOWN}{3 LEFT}{3 SPACES}{DOWN}
    {3 LEFT}{3 SPACES}{DOWN}{3 LEFT}{3 SPACES}"
                                           :rem 61
520 PRINTSC$TAB(T)A$(J):RETURN             :rem 116
```

Program 2. 3-D Clock, 64 Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 DATA"{RVS}{BLK}£{2 SPACES}{DOWN}{3 LEFT}
   {RIGHT} {DOWN}{3 LEFT} {RIGHT} {DOWN}{3 LEFT}
   {RIGHT} {DOWN}{3 LEFT}{2 SPACES}{OFF}£" :rem 60
20 DATA"{RVS}£ {DOWN}{LEFT} {DOWN}{LEFT} {DOWN}
   {LEFT} {DOWN}{2 LEFT}£ {OFF}£"          :rem 180
30 DATA"{RVS}£{2 SPACES}{DOWN}{LEFT} {DOWN}
   {3 LEFT}£ {OFF}£{RVS}{DOWN}{3 LEFT} {DOWN}
   {LEFT}{2 SPACES}{OFF}£"                 :rem 217
40 DATA"{RVS}£{2 SPACES}{DOWN}{LEFT} {DOWN}
   {2 LEFT}{2 SPACES}{DOWN}{LEFT} {DOWN}{3 LEFT}£
    {OFF}£"                                :rem 240
50 DATA"{RVS}£{RIGHT}£{DOWN}{3 LEFT} {RIGHT}
   {DOWN}{3 LEFT}{3 SPACES}{DOWN}{LEFT} {DOWN}
   {LEFT}{OFF}£"                           :rem 200
60 DATA"{RVS}£ {OFF}£{RVS}{DOWN}{3 LEFT} {DOWN}
   {LEFT}{3 SPACES}{DOWN}{LEFT} {DOWN}{3 LEFT}£
   {OFF}£"                                 :rem 220
70 DATA"{RVS}£ {OFF}£{RVS}{DOWN}{3 LEFT} {DOWN}
   {LEFT}{3 SPACES}{DOWN}{3 LEFT} {RIGHT} {DOWN}
   {3 LEFT}{2 SPACES}{OFF}£"               :rem 139
80 DATA"{RVS}£{2 SPACES}{DOWN}{LEFT} {DOWN}{LEFT}
    {DOWN}{LEFT} {DOWN}{LEFT}{OFF}£"       :rem 116
90 DATA"{RVS}£{2 SPACES}{DOWN}{3 LEFT} {RIGHT}
   {DOWN}{3 LEFT}{3 SPACES}{DOWN}{3 LEFT} {RIGHT}
   {SPACE}{DOWN}{3 LEFT}{2 SPACES}{OFF}£":rem 151
100 DATA"{RVS}£{2 SPACES}{DOWN}{3 LEFT} {RIGHT}
    {DOWN}{3 LEFT}{3 SPACES}{DOWN}{LEFT} {DOWN}
    {LEFT}{OFF}£"                          :rem 46
110 DATA"{DOWN} {RVS} {2 DOWN}{LEFT} "     :rem 210
120 CLR:DIMA$(9),M$(10):FORT=0TO9:READA$(T):NEXT:R
    EADPU$                                 :rem 124
122 CV=54272:CS=53281:CB=53280:SC=1024:CM=55296
                                           :rem 99
125 POKECS,1                               :rem 186
130 PRINT"{CLR}{BLK}{4 SPACES}STEREO CLOCK"
                                           :rem 198
145 S1$="{HOME}{20 DOWN}{15 RIGHT}":SC$=LEFT$(S1$,
    8)                                     :rem 186
150 J2=60:J3=60:J4=60:J5=60:J6=60:J7=60    :rem 129
260 PRINT"{BLU}ENTER THE PRESENT TIME"     :rem 55
270 INPUT"{DOWN}HOUR";HO:IFHO<0ORHO>12THEN270
                                           :rem 251
275 HO$=RIGHT$(STR$(HO),2):IFHO<10THENHO$="0"+RIGH
    T$(HO$,1)                              :rem 186
280 INPUT"{DOWN}MIN.";ME:IFME<0ORME>59THEN280
                                           :rem 205
```

```
283 INPUT"{DOWN}SEC.";SE:IFSE<ØORSE>59THEN283
                                          :rem 220
284 SE$=RIGHT$(STR$(SE),2):IFSE<1ØTHENSE$="Ø"+RIGH
    T$(SE$,1)                             :rem 191
285 ME$=RIGHT$(STR$(ME),2):IFME<1ØTHENME$="Ø"+RIGH
    T$(ME$,1)                             :rem 162
29Ø PT$=HO$+ME$+SE$:PRINT"{DOWN}HIT F1 KEY TO STAR
    T"                                    :rem 181
3ØØ GETA$:IFA$<>"{F1}"THEN3ØØ             :rem 11
31Ø TI$=PT$:PRINT"{CLR}"                  :rem 248
312 POKE1358+CV,Ø:POKE1358,16Ø:POKE1368+CV,Ø:POKE1
    368,16Ø                               :rem 12
314 POKE1438+CV,Ø:POKE1438,16Ø:POKE1448+CV,Ø:POKE1
    448,16Ø                               :rem 10
32Ø T2$=TI$:IFLEFT$(T2$,2)="13"THENTI$="Ø10ØØØ":T2
    $=TI$                                  :rem 7
33Ø T=6:J= VAL(MID$(T2$,1,1)):IF J<>J2 THEN J2=J:G
    OSUB 5ØØ                              :rem 143
34Ø T=1Ø:J=VAL(MID$(T2$,2,1)):IF J<>J3THEN J3=J:GO
    SUB 5ØØ                               :rem 19Ø
35Ø T= 16:J=VAL(MID$(T2$,3,1)):IF J<>J4THENJ4=J:GO
    SUB 5ØØ                               :rem 2ØØ
355 T= 2Ø:J=VAL(MID$(T2$,4,1)):IFJ<>J5THENJ5=J:GOS
    UB 5ØØ                                :rem 2Ø3
36Ø T= 26:J=VAL(MID$(T2$,5,1)):IF J<>J6THENJ6=J:GO
    SUB 5ØØ                               :rem 2Ø8
365 T= 3Ø:J=VAL(MID$(T2$,6,1)):IF J<>J7THENJ7=J:GO
    SUB 5ØØ                               :rem 211
37Ø GOTO32Ø                               :rem 1Ø4
5ØØ PRINTS1$LEFT$(T2$,2)":"MID$(T2$,3,2)":"RIGHT$(
    T2$,2)                                 :rem 31
51Ø PRINTSC$TAB(T)"{3 SPACES}{DOWN}{3 LEFT}
    {3 SPACES}{DOWN}{3 LEFT}{3 SPACES}{DOWN}
    {3 LEFT}{3 SPACES}{DOWN}{3 LEFT}{3 SPACES}"
                                          :rem 61
52Ø PRINTSC$TAB(T)A$(J):RETURN            :rem 116
```

General-Purpose Bar Chart Routine

Sal Raciti
64 Translation by David Florance

This versatile bar chart routine can be adapted to your programs to give any set of data exciting visual appeal. It runs on the VIC or the 64.

Many computer applications produce numerical outputs, and for some applications a simple listing of those numbers is all you'll need. But in many cases a set of numbers will have the greatest meaning if they are represented graphically.

This subroutine, designed to be appended to your own programs, plots values on a multicolor bar graph. The subroutine tests to see if the values are all positive, all negative, or both positive and negative, and selects a plotting format accordingly. In addition, it scales all values so that they will fit on the screen.

Note that several variables are used:

HH The number of bars to be displayed, up to a maximum of 16. Row 1 will be printed at the bottom of the graph.

XX$ The title of the graph, which may also be displayed in reverse video. Maximum length is limited to 20 characters.

**ZZ$** Bar labels. A maximum of 16 two-character labels is allowed. The length of ZZ$ should equal 2*HH; for single character labels, make one of the two characters a space.

B(N) Bar values. N ranges from 1 to HH. These values may be positive, negative or both. Note that the elements of B(N) can either be read in from DATA statements or computed in an expression that is tied to N (for instance, WW = N*3.7).

Once values have been assigned to these variables, the bar chart subroutine can be called at any time. Lines 10000-10160 locate the maximum positive and negative values of B(N). Those values are assigned to variables ZZ and YY, respectively, and allow lines 10180-10355 to calculate an appropriate scale factor (not to exceed 10,000,000).

To determine the scale factor, ZZ and YY are first evaluated to find the power of 10↑(XX) that would simultaneously make ZZ/10↑XX < 10 and YY/10↑XX> − 10. For instance, if ZZ = 9000 and YY = − 100 then XX would equal 3. Then, since the printed X-axis will be calibrated only from − 1 to 1, line 10480 calculates the

actual scale factor as 10↑(XX+1) and prints the result on the screen.

Lines 10360-10380 then scale the values. Each B(N) is divided by 10↑XX (instead of by 10↑(XX+1), since the X-axis ranges from 1 to −1 instead of from 10 to −10).

Lines 10520-10540 print the appropriate X axis on the screen. If ZZ or YY equals 0 (that is, if the X-axis is calibrated from −1 to 0 or from 0 to 1), then each horizontal division is given two character spaces. Otherwise (if there are both positive and negative values for B(N)), the X-axis will be calibrated from −1 through 0 to 1, and each horizontal division will be given one character space. Lines 10560-10800 then print the appropriate X-axis labels; the Y-axis is printed by lines 10840-10920.

Line 10940 selects the value of SS to scale (for printing purposes) the values of B(N). If bar values are all positive or all negative, that value is 2. If B(N) takes on both positive and negative values, a value of 1 is used.

Lines 10960-11460 form a loop that prints the bars. Bars are made up of whole reverse spaces (CHR$(32)) and partial reverse spaces (CHR$(180), CHR$(161), or CHR$(170)). The row location of each bar is determined by lines 10970-10972, and the loop is repeated once for each bar. If B(N) equals zero, that bar location is skipped by line 11082 or 11084. Bar colors are determined by lines 11070 and 11080, and bar labels are printed by lines 11480-11700.

The string variables used in this subroutine are XX$, YY$, and ZZ$. Numeric variables used are B(N), N, and LL through ZZ. Be extremely careful if you use one of these variables in your main program, or you may get some startling results.

To see how this subroutine works, type in the demonstration program (one version works on either the VIC or the 64) and add the appropriate version of the bar graph subroutine.

Program 1. Bar Chart Demo

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
5 DIM B(16):PRINT"{CLR}"                                    :rem 167
7 POKE 53280,6:POKE53281,1                                  :rem 147
10 FOR N=1TO 16                                              :rem 14
20 WW=+N*N*3.6                                              :rem 255
30 B(N)=WW                                                   :rem 47
40 HH=N                                                     :rem 127
45 NEXTN                                                    :rem 246
50 XX$="{RVS}COMPUTE! BOOKS{OFF}"                            :rem 26
55 ZZ$="Y1Y2Y3Y4Y5Y6Y7Y8Y9Y0Y1Y2Y3Y4Y5Y6"                   :rem 149
60 GOSUB10000                                               :rem 215
70 GOTO 70                                                    :rem 7
```

This particular demonstration calculates bar values using an expression. In this case, all values are positive; to see how the routine handles negative numbers, change line 20 to read WW=5*(−1.2)↑N. You can use any expression that you wish, and the subroutine will automatically scale and plot the values for you.

To display a known set of values, you can read in bar values from DATA statements. For example, change line 20 in the demonstration program to:

20 READ WW

and add a line with 16 items of data, for instance:

100 DATA 112,276,164,301,184,427,200,358,199,495,256,143,460,382,234,105

Remember that your scaling factor can be no larger than 10,000,000. To see what happens when you introduce numbers that are too big, let WW=N↑N. That gives WW a maximum value of 16↑16, or roughly 1.8447*10↑19 — just a bit beyond acceptable scaling range!

Program 2. General-Purpose Bar Charts, VIC Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10000 REM BARCHART                                  :rem 28
10020 WW=0:XX=0:YY=0:ZZ=0                           :rem 25
10040 FORN=1TOHH                                   :rem 203
10060 IFB(N)>=0ANDB(N)>ZZTHENZZ=B(N)               :rem 185
10080 IFB(N)<0ANDB(N)<YYTHENYY=B(N)                :rem 118
10160 NEXTN                                        :rem 133
10180 IFZZ>10THENZZ=ZZ/10:WW=WW+1                  :rem 111
10200 IFZZ>10GOTO10180                               :rem 8
10220 IFYY<-10THENYY=YY/10:XX=XX+1                 :rem 147
10240 IFYY<-10GOTO10220                             :rem 48
10253 IFYY=0THENXX=WW                              :rem 115
10280 IFZZ<=10ANDZZ>0THENZZ=ZZ*10:WW=WW-1          :rem 157
10300 IFZZ<10ANDZZ>0GOTO10280                      :rem 253
10310 IFYY>=-10ANDYY<0THENYY=YY*10:XX=XX-1         :rem 192
10320 IFYY>-10ANDYY<0GOTO10310                      :rem 34
10330 IFZZ>10THENWW=WW+1                           :rem 253
10340 IFYY<-10THENXX=XX+1                           :rem 43
10352 IFZZ=0THENXX=XX                              :rem 119
10353 IFYY=0THENXX=WW                              :rem 116
10354 IFZZ<>0ANDYY<>0ANDWW>XXTHENXX=WW              :rem 82
10355 IFZZ=0ANDYY=0THENWW=-1:XX=-1                 :rem 157
10360 FORN=1TOHH:B(N)=B(N)/(10↑XX)                 :rem 248
10380 NEXTN                                        :rem 137
10400 N=LEN(XX$)                                   :rem 132
10410 PRINT"{BLU}";                                 :rem 33
```

```
10420 PRINT"{CLR}";:IFN>20THENPRINT"TITLE TOO LONG
      !":GOTO11720                          :rem 128
10430 IFHH>16ORHH<1THENPRINT"NO. OF BARS INCORRECT
      !":GOTO11720                           :rem 26
10435 IFXX>6THENPRINT"SCALE FACT. TOO LARGE!":GOTO
      11720                                  :rem 80
10440 N=INT((22-N)/2)                        :rem 81
10460 PRINTSPC(N)XX$                        :rem 225
10480 PRINT"{RVS}SCALE FACTOR{OFF}"10↑(XX+1):PRINT
      "{16 DOWN}"SPC(2)                      :rem 57
10520 FORN=1TO20                            :rem 160
10530 IFZZ=0ORYY=0THENPRINTCHR$(111)CHR$(112);:N=N
      +1                                    :rem 154
10535 IFZZ<>0ANDYY<>0THENPRINTCHR$(111);    :rem 246
10540 NEXTN                                 :rem 135
10560 IFZZ=0ANDYY=0GOTO10680                :rem 214
10600 IFZZ=0GOTO10720                       :rem 218
10620 IFYY=0GOTO10800                       :rem 217
10640 PRINTSPC(1)"-1"SPC(17)"+1";            :rem 12
10660 GOTO10840                              :rem 51
10680 PRINT"{HOME}{4 DOWN}NO BAR CHART VALUES!":GO
      TO11720                               :rem 106
10720 PRINTSPC(1)"-1"SPC(18)"0";            :rem 224
10760 GOTO10840                              :rem 52
10800 PRINTSPC(2)"0"SPC(17)"+1";            :rem 221
10840 PRINT"{HOME}{DOWN}"                   :rem 242
10860 NN=12:RR=165:IFZZ=0THENNN=21:RR=167     :rem 0
10880 IFYY=0THENNN=2:RR=165                 :rem 160
10900 FORN=1TO16                            :rem 167
10910 IFNN<>21THENPRINTSPC(NN)CHR$(RR)      :rem 136
10911 IFNN=21THENPRINTSPC(21)CHR$(167);      :rem 72
10920 NEXTN                                 :rem 137
10940 SS=1:IFYY=0ORZZ=0THENSS=2               :rem 0
10960 FORN=1TOHH                            :rem 214
10970 PRINT"{HOME}";                         :rem 32
10971 FORLL=1TO18-N:PRINT"{DOWN}";          :rem 205
10972 NEXTLL                                :rem 218
10980 IFB(N)=0GOTO11460                      :rem 20
11020 UU=B(N)*SS                            :rem 140
11040 IFB(N)<0THENTT=-UU-INT(-UU):UU=-UU    :rem 245
11045 MM=0                                    :rem 2
11050 IFZZ<>0ANDYY<>0THENMM=1                :rem 74
11060 IFB(N)>0THENTT=UU-INT(UU)             :rem 167
11070 IFN/2-INT(N/2)=0THENPRINT"{RED}";      :rem 19
11075 IFN/2-INT(N/2)<>0THENPRINT"{YEL}";    :rem 215
11080 FORQQ=1TOINT(UU)                      :rem 122
11082 IFSS=2ANDUU<1ANDB(N)>=0THENPP=1:GOTO11170
                                            :rem 147
11083 IFSS=2ANDUU<1ANDB(N)<0THENPP=22:GOTO11170
                                            :rem 136
```

```
11084 IFSS=1ANDUU<1ANDB(N)>=ØTHENPP=11:GOTO11170
                                                :rem 197
11085 IFSS=1ANDUU<1ANDB(N)<ØTHENPP=12:GOTO11170
                                                :rem 136
11086 PRINT"{RVS}";                              :rem 30
11100 IFYY=ØTHENPRINTSPC(1+QQ)CHR$(32):PP=1+QQ
                                                 :rem 94
11120 IFZZ=ØTHENPRINTSPC(22-QQ)CHR$(32):PP=22-QQ
                                                :rem 204
11140 IFMM=1ANDB(N)>=ØTHENPRINTSPC(11+QQ)CHR$(32):
      PP=11+QQ                                   :rem 12
11160 IFMM=1ANDB(N)<ØTHENPRINTSPC(12-QQ)CHR$(32):P
      P=12-QQ                                   :rem 213
11170 PRINT"{HOME}";                             :rem 25
11172 FORLL=1TO18-N:PRINT"{DOWN}";               :rem 199
11173 NEXTLL                                     :rem 212
11180 NEXTQQ                                     :rem 220
11185 PRINT"{OFF}";                              :rem 158
11200 IFTT>=.12ANDTT<.37THENTT=2                 :rem 204
11220 IFTT>=.37ANDTT<.62THENTT=3                 :rem 212
11240 IFTT>=.62ANDTT<.87THENTT=4                 :rem 220
11260 IFTT>=.87ANDTT<=1THENTT=5                  :rem 183
11285 IFB(N)>=ØGOTO11300                          :rem 74
11295 IFB(N)<ØGOTO11380                           :rem 20
11300 IFTT=2THENPRINTSPC(PP+1)CHR$(180)          :rem 117
11320 IFTT=3THENPRINTSPC(PP+1)CHR$(161)          :rem 119
11340 IFTT=4THENPRINTSPC(PP+1)"{RVS}"CHR$(170)"
      {OFF}"                                     :rem 166
11350 IFTT=5THENPRINTSPC(PP+1)"{RVS}"CHR$(32)"
      {OFF}"                                     :rem 117
11370 GOTO11460                                   :rem 49
11380 IFTT=2THENPRINTSPC(PP-1)CHR$(170)          :rem 126
11400 IFTT=3THENPRINTSPC(PP-1)"{RVS}"CHR$(161)"
      {OFF}"                                     :rem 164
11420 IFTT=4THENPRINTSPC(PP-1)"{RVS}"CHR$(180)"
      {OFF}"                                     :rem 168
11430 IFTT=5THENPRINTSPC(PP-1)"{RVS}"CHR$(32)"
      {OFF}"                                     :rem 118
11460 NEXTN                                      :rem 137
11480 PRINT"{BLU}",                               :rem 26
11500 PRINT"{HOME}";                              :rem 22
11560 FORN=18TO1STEP-1                            :rem 70
11565 IFN>HHTHENPRINT"{DOWN}";                   :rem 249
11580 IFN<=HHTHENYY$=MID$(ZZ$,2*N-1,2)           :rem 224
11590 IFN<=HHTHENPRINTYY$SPC(20);                 :rem 76
11640 NEXTN                                      :rem 137
11660 PRINT"{HOME}";                              :rem 29
11680 FORN=1TO20:PRINT"{DOWN}";                  :rem 255
11700 NEXTN                                      :rem 134
11720 RETURN                                     :rem 219
```

Program 3. General-Purpose Bar Chart, 64 Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10000 REM******BARCHART                          :rem 24
10020 WW=0:XX=0:YY=0:ZZ=0                        :rem 25
10040 FORN=1TOHH                                 :rem 203
10060 IFB(N)>=0ANDB(N)>ZZTHENZZ=B(N)             :rem 185
10080 IFB(N)<0ANDB(N)<YYTHENYY=B(N)              :rem 118
10160 NEXTN                                      :rem 133
10180 IFZZ>10THENZZ=ZZ/10:WW=WW+1                :rem 111
10200 IFZZ>10GOTO10180                            :rem 8
10220 IFYY<-10THENYY=YY/10:XX=XX+1               :rem 147
10240 IFYY<-10GOTO10220                          :rem 48
10253 IFYY=0THENXX=WW                            :rem 115
10280 IFZZ<=10ANDZZ>0THENZZ=ZZ*10:WW=WW-1        :rem 157
10300 IFZZ<10ANDZZ>0GOTO10280                    :rem 253
10310 IFYY>=-10ANDYY<0THENYY=YY*10:XX=XX-1       :rem 192
10320 IFYY>-10ANDYY<0GOTO10310                   :rem 34
10330 IFZZ>10THENWW=WW+1                         :rem 253
10340 IFYY<-10THENXX=XX+1                        :rem 43
10352 IFZZ=0THENXX=XX                            :rem 119
10353 IFYY=0THENXX=WW                            :rem 116
10354 IFZZ<>0ANDYY<>0ANDWW>XXTHENXX=WW           :rem 82
10355 IFZZ=0ANDYY=0THENWW=-1:XX=-1               :rem 157
10360 FORN=1TOHH:B(N)=B(N)/(10↑XX)               :rem 248
10380 NEXTN                                      :rem 137
10400 N=LEN(XX$)                                 :rem 132
10410 PRINT"{BLU}";                              :rem 33
10420 PRINT"{CLR}";:IFN>20THENPRINT"TITLE TOO LONG
      !":GOTO11720                               :rem 128
10430 IFHH>16ORHH<1THENPRINT"NO. OF BARS INCORRECT
      !":GOTO11720                               :rem 26
10435 IFXX>6THENPRINT"SCALE FACT. TOO LARGE!":GOTO
      11720                                      :rem 80
10440 N=INT((22-N)/2)                            :rem 81
10460 PRINTSPC(17)XX$                            :rem 251
10480 PRINT"{RVS}{7 RIGHT}SCALE FACTOR{OFF}"10↑(XX
      +1):PRINT"{16 DOWN}"SPC(9)                 :rem 11
10520 FORN=1TO20                                 :rem 160
10530 IFZZ=0ORYY=0THENPRINTCHR$(111)CHR$(112);:N=N
      +1                                         :rem 154
10535 IFZZ<>0ANDYY<>0THENPRINTCHR$(111);         :rem 246
10540 NEXTN                                      :rem 135
10560 IFZZ=0ANDYY=0GOTO10680                     :rem 214
10600 IFZZ=0GOTO10720                            :rem 218
10620 IFYY=0GOTO10800                            :rem 217
10640 PRINTSPC(19)"-1"SPC(9)"0"SPC(8)"+1"        :rem 190
10660 GOTO10840                                   :rem 51
10680 PRINT"{HOME}{4 DOWN}NO BAR CHART VALUES!":GO
      TO11720                                    :rem 106
```

```
10720 PRINTSPC(19)"-1"SPC(18)"0";                    :rem 25
10760 GOTO10840                                      :rem 52
10800 PRINTSPC(20)"0"SPC(18)"+1";                    :rem 14
10840 PRINT"{HOME}{DOWN}"                           :rem 242
10860 NN=12:RR=165:IFZZ=0THENNN=21:RR=167             :rem 0
10880 IFYY=0THENNN=2:RR=165                         :rem 160
10900 FORN=1TO16                                    :rem 167
10910 IFNN<>40THENPRINT"{7 RIGHT}"SPC(NN)CHR$(RR)
                                                    :rem 152
10911 IFNN=40THENPRINT"{7 RIGHT}"SPC(40)CHR$(167);
                                                     :rem 89
10920 NEXTN                                         :rem 137
10940 SS=1:IFYY=0ORZZ=0THENSS=2                       :rem 0
10960 FORN=1TOHH                                    :rem 214
10970 PRINT"{HOME}";                                 :rem 32
10971 FORLL=1TO18-N:PRINT"{DOWN}";                  :rem 205
10972 NEXTLL                                        :rem 218
10980 IFB(N)=0GOTO11460                              :rem 20
11020 UU=B(N)*SS                                    :rem 140
11040 IFB(N)<0THENTT=-UU-INT(-UU):UU=-UU            :rem 245
11045 MM=0                                            :rem 2
11050 IFZZ<>0ANDYY<>0THENMM=1                        :rem 74
11060 IFB(N)>0THENTT=UU-INT(UU)                     :rem 167
11070 IFN/2-INT(N/2)=0THENPRINT"{RED}";              :rem 19
11075 IFN/2-INT(N/2)<>0THENPRINT"{YEL}";            :rem 215
11080 FORQQ=1TOINT(UU)                              :rem 122
11082 IFSS=2ANDUU<1ANDB(N)>=0THENPP=1:GOTO11170
                                                    :rem 147
11083 IFSS=2ANDUU<1ANDB(N)<0THENPP=22:GOTO11170
                                                    :rem 136
11084 IFSS=1ANDUU<1ANDB(N)>=0THENPP=11:GOTO11170
                                                    :rem 197
11085 IFSS=1ANDUU<1ANDB(N)<0THENPP=12:GOTO11170
                                                    :rem 136
11086 PRINT"{RVS}";                                  :rem 30
11100 IFYY=0THENPRINT"{7 RIGHT}"SPC(1+QQ)CHR$(32):
      PP=1+QQ                                       :rem 109
11120 IFZZ=0THENPRINT"{7 RIGHT}"SPC(22-QQ)CHR$(32)
      :PP=22-QQ                                     :rem 219
11140 IFMM=1ANDB(N)>=0THENPRINT"{7 RIGHT}"SPC(11+Q
      Q)CHR$(32):PP=11+QQ                           :rem 27
11160 IFMM=1ANDB(N)<0THENPRINT"{7 RIGHT}"SPC(12-QQ
      )CHR$(32):PP=12-QQ                            :rem 228
11170 PRINT"{HOME}";                                 :rem 25
11172 FORLL=1TO18-N:PRINT"{DOWN}";                  :rem 199
11173 NEXTLL                                        :rem 212
11180 NEXTQQ                                        :rem 220
11185 PRINT"{OFF}";                                 :rem 158
11200 IFTT>=.12ANDTT<.37THENTT=2                    :rem 204
```

```
11220 IFTT>=.37ANDTT<.62THENTT=3          :rem 212
11240 IFTT>=.62ANDTT<.87THENTT=4          :rem 220
11260 IFTT>=.87ANDTT<=1THENTT=5           :rem 183
11285 IFB(N)>=0GOTO11300                   :rem 74
11295 IFB(N)<0GOTO11380                    :rem 20
11300 IFTT=2THENPRINT"{7 RIGHT}"SPC(PP+1)CHR$(180)
                                          :rem 132
11320 IFTT=3THENPRINT"{7 RIGHT}"SPC(PP+1)CHR$(161)
                                          :rem 134
11340 IFTT=4THENPRINT"{7 RIGHT}"SPC(PP+1)"{RVS}"CH
      R$(170)"{OFF}"                      :rem 181
11350 IFTT=5THENPRINT"{7 RIGHT}"SPC(PP+1)"{RVS}"CH
      R$(32)"{OFF}"                       :rem 132
11370 GOTO11460                            :rem 49
11380 IFTT=2THENPRINT"{7 RIGHT}"SPC(PP-1)CHR$(170)
                                          :rem 141
11400 IFTT=3THENPRINT"{7 RIGHT}"SPC(PP-1)"{RVS}"CH
      R$(161)"{OFF}"                      :rem 179
11420 IFTT=4THENPRINT"{7 RIGHT}"SPC(PP-1)"{RVS}"CH
      R$(180)"{OFF}"                      :rem 183
11430 IFTT=5THENPRINT"{7 RIGHT}"SPC(PP-1)"{RVS}"CH
      R$(32)"{OFF}"                       :rem 133
11460 NEXTN                               :rem 137
11480 PRINT"{BLU}",                        :rem 26
11500 PRINT"{HOME}";                       :rem 22
11560 FORN=18TO1STEP-1                     :rem 70
11565 IFN>HHTHENPRINT"{DOWN}";            :rem 249
11580 IFN<=HHTHENYY$=MID$(ZZ$,2*N-1,2)    :rem 224
11590 IFN<=HHTHENPRINTYY$SPC(38);          :rem 85
11640 NEXTN                               :rem 137
11660 PRINT"{HOME}";                       :rem 29
11680 FORN=1TO20:PRINT"{DOWN}";           :rem 255
11700 NEXTN                               :rem 134
11720 RETURN                              :rem 219
```

Advertiser

Robert Lykins

If you're a business person, you recognize your computer's value as a bookkeeping or file-managing tool. But have you thought of using it for marketing? "Advertiser" turns any VIC or 64 into a computer-controlled marquee that's easily adaptable to any promotion.

A 22- or 40-column display, like that of the VIC or 64, is sometimes considered less desirable than the 80-column displays found on more expensive computers. However, the larger characters of the Commodore displays are indisputably easier to read, particularly from a distance. That can be quite an asset for certain uses, particularly in advertising.

The following program is an example of the advertising potential of your computer. A large TV screen, placed behind a properly shaded window, provides an effective display medium. The displayed words are easily changed, giving Madison Avenue flexibility at a hometown price.

The display is composed of a moving, marquee-style line (B$); a flashing, large-lettered word (A$); and a seven-line capacity box (C$ array). Line 500 controls the speed of the display.

The marquee string, B$, can contain up to 255 characters, but concatenation (string addition) becomes necessary since a program line will accommodate a total of only 88 characters. The programming involved is not complex. Simply insert lines for additional strings (such as B1$, B2$, and B3$) between lines 130 and 140. Then, in a final line, add them together as follows:

```
139 B$ = B$ + B1$ + B2$ + B3$
```

Because of its length, line 130 must be typed with no space after the line number. The cursor must be returned to the statement with a cursor key before hitting RETURN or an error will occur.

Line 470 in the VIC version, which produces the right-to-left movement of the marquee, also requires special attention. The problem is that the DELete character ({ DEL }) cannot be printed while in the normal quote mode. Try it and you will delete the previously typed character. You must close the quotes after typing the { RIGHT } character, move the cursor one space to the left so that it rests on the quotation marks, and then type an INSert

(SHIFT-DELete). Now hit the DELete key and a reverse-video T should appear. That is the symbol for the DELete character in a PRINT statement. Then move the cursor to the right of the quotation marks and finish the line.

When you list line 470, don't be alarmed to find that the {RIGHT} character and the {DEL} character are missing. Unlike the other control characters, the DELETE executes on listing. Both it and the {RIGHT} character are still in the line; however, you will have to retype them if you make subsequent changes to the line. There is no {DEL} character in line 470 of the 64 version.

The flashing, large-lettered word, A$, may consist of up to five letters. The sample program uses the word CAFE. An alternate choice might be the word SALE. Line 220 causes the word to print in the multicolor mode so that line 480 will make it flash in changing auxiliary colors. Changing the symbol that makes up the letters of the word in line 300 will change the shape and color of the letters. Many interesting variations are possible, and you can try reverse-video characters, too.

The remaining portion of the display consists of seven lines enclosed in a box on the lower part of the screen. The box is drawn in lines 370 through 410. Refer to Appendix B, "How to Type In Programs," for information on the special characters used.

The display lines are defined by the C$ array in lines 140 through 200. In this example, the second, fourth, and sixth lines are empty strings, but they can easily be filled to display more information. The lines should consist of no more than 18 characters each, excluding control characters; closed quotes are not necessary.

Since control characters affect the LENgth of the string, you may sometimes need to add one or two {RIGHT} characters in order to maintain centering. If, for example, you want the string in line 140 to print as white, you would add a {WHT} control character after the {RVS} character. However, that will make the line print to the left one space. To counter this, you should also add a {RIGHT} character. Each *two* characters in the string will move the print position *one* space left of center. This is accomplished in line 590.

Program 1. Advertiser, VIC Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 REM VIC ADVERTISING                          :rem 151
110 PRINT"{CLR}                                  :rem 212
120 A$="CAFE                                     :rem 102
130 B$="HUNGRY? THIRSTY? FROM SNACKS TO FULL DINNE
    RS, COMPUTER MALL CAFE IS THE SPOT. "        :rem 174
140 C$(1)="{RVS}TODAY'S SPECIAL                  :rem 235
150 C$(2)="                                      :rem 223
160 C$(3)="{RVS}HAMBURGER, FRIES,                 :rem 97
170 C$(4)="                                      :rem 227
180 C$(5)="{RVS}AND SHAKE                         :rem 54
190 C$(6)="                                      :rem 231
200 C$(7)="{RVS}$2.75                            :rem 226
210 D$="{HOME}{13 DOWN}                           :rem 74
220 POKE646,9                                    :rem 200
230 FORA=1TOLEN(A$)                               :rem 99
240 B=ASC(MID$(A$,A,1))-64                        :rem 82
250 FORC=0TO7                                      :rem 8
260 D=PEEK(32768+B*8+C)                          :rem 214
270 E=64                                         :rem 133
280 FORF=0TO7                                      :rem 14
290 IFD<ETHEN320                                 :rem 179
300 PRINT"{HOME}"TAB(110+C*22)SPC(12-LEN(A$)*2+F/2
    +G)"Q                                         :rem 28
310 D=D-E                                        :rem 203
320 E=E/2                                        :rem 189
330 NEXT                                         :rem 213
340 NEXT                                         :rem 214
350 G=G+4                                        :rem 194
360 NEXT                                         :rem 216
370 PRINT"{DOWN}{WHT} O{18 T}P                   :rem 116
380 FORL=1TO7                                     :rem 22
390 PRINT" {G}{18 SPACES}{M}                     :rem 151
400 NEXT                                         :rem 211
410 PRINT" L{18 @}@                               :rem 82
420 FORA=2TO7                                      :rem 7
430 POKE36879,8+A                                :rem 167
440 FORB=1TO2                                      :rem 4
450 GOSUB560                                     :rem 180
460 FORC=1TOLEN(B$)                              :rem 107
470 PRINT"{HOME}{2 DOWN}{RVS}{WHT}{RIGHT}{DEL}"SPC
    (21)MID$(B$,C,1)                               :rem 4
480 POKE36878,16*(INT(D/10)+2)                    :rem 86
490 D=D+1:IFD=60THEND=0                           :rem 78
500 FORL=1TO60                                    :rem 63
510 NEXT                                         :rem 213
520 NEXT                                         :rem 214
```

```
530 NEXT                                        :rem 215
540 NEXT                                        :rem 216
550 GOTO420                                     :rem 105
560 PRINTD$                                     :rem 144
570 FORL=1TO7                                    :rem 23
580 POKE646,INT(RND(1)*6)+2                     :rem 247
590 PRINTTAB(12-LEN(C$(L))/2)C$(L)              :rem 124
600 NEXT                                        :rem 213
610 RETURN                                      :rem 119
```

Program 2. Advertiser, 64 Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 DIMCH(3,7):IR=56334:CC=646:S=54272:BK=53281:BO
    =53280:CR=53248                             :rem 130
110 PRINT"{CLR}";CHR$(142);CHR$(8)              :rem 223
120 DATA3,1,6,5:REM "CAFE"                      :rem 113
130 B$="HUNGRY? THIRSTY? FROM SNACKS TO FULL DINNE
    RS, COMPUTER MALL CAFE HITS"                :rem 245
135 B$=B$+" THE SPOT.{40 SPACES}"               :rem 102
140 C$(1)="{RVS}TODAY'S SPECIAL                 :rem 235
150 C$(2)="                                     :rem 223
160 C$(3)="{RVS}HAMBURGER, FRIES,                :rem 97
170 C$(4)="                                     :rem 227
180 C$(5)="{RVS}AND SHAKE                        :rem 54
190 C$(6)="                                     :rem 231
200 C$(7)="{RVS}$2.75                           :rem 226
210 D$="{HOME}{15 DOWN}"                        :rem 142
220 POKECC,12                                   :rem 216
230 POKEIR,PEEK(IR)AND254:POKE1,PEEK(1)AND251
                                                :rem 226
240 FORJ=0TO3:READA:FORI=0TO7:CH(J,I)=PEEK(CR+A*8+
    I):NEXTI,J                                  :rem 233
250 POKE1,PEEK(1)OR4:POKEIR,PEEK(IR)OR1         :rem 178
260 FORK=0TO3:PRINT"{HOME}{5 DOWN}";TAB(1+10*K);
                                                :rem 80
270 FORJ=0TO7:FORI=7TO0STEP-1:IF(CH(K,J)AND2↑I)=(2
    ↑I)THENPRINT"Q";:GOTO290                     :rem 64
280 PRINT" ";                                   :rem 166
290 NEXTI:PRINTCHR$(141);TAB(1+10*K);:NEXTJ,K:PRIN
    T                                           :rem 163
370 PRINT"{DOWN}{WHT}{9 RIGHT} O{18 T}P         :rem 121
380 FORL=1TO7                                    :rem 22
390 PRINT"{9 RIGHT} {G}{18 SPACES}{M}           :rem 156
400 NEXT                                        :rem 211
410 PRINT"{9 RIGHT} L{18 @}@                     :rem 87
420 FORA=2TO7                                     :rem 7
430 POKEBK,0:POKEBO,0+A                         :rem 113
```

```
440 FORB=1TO2                                          :rem 4
450 GOSUB560                                          :rem 180
455 PRINT"{HOME}{2 DOWN}{RVS}{WHT}{40 SPACES}"
                                                      :rem 187
460 FORC=1TOLEN(B$)                                   :rem 107
465 I=40-C:IFI<0THENI=0                                :rem 92
468 E=1:IFC>40THENE=C-40                              :rem 136
470 PRINT"{HOME}{2 DOWN}{RVS}{WHT}"SPC(I)MID$(B$,E
    ,40-I);                                           :rem 159
480 POKE53270,PEEK(53270)OR16:POKE53282,(INT(D/10)
    ):POKE53283,(INT(D/10))                           :rem 108
490 D=D+1:IFD=80THEND=0                                :rem 80
500 FORL=1TO60                                         :rem 63
510 NEXT                                              :rem 213
520 NEXT                                              :rem 214
530 NEXT                                              :rem 215
540 NEXT                                              :rem 216
550 GOTO420                                           :rem 105
560 PRINTD$                                           :rem 144
570 FORL=1TO7                                          :rem 23
580 POKECC,INT(RND(1)*6)+2                            :rem 221
590 PRINTTAB(21-LEN(C$(L))/2)C$(L)                    :rem 124
600 NEXT                                              :rem 213
610 RETURN                                            :rem 119
```

Chapter 4

Programming Aids

Remarkable REMs

Louis F. Sander

Would you like to dress up your REMs? Here is a routine for the VIC or 64 that will let you create REM statements centered inside custom borders — and they will list without line numbers or the keyword REM.

This routine dramatically sets off REM statements, highlighting them and enhancing the readability of your program listings. With minor modifications, the program works on both the VIC-20 and the Commodore 64. However, the REM and line number suppression features may not work with all printers.

Follow these instructions when typing in the program:

1. The { SHIFT-SPACE } characters in lines 63820, 63840, and 63860 are obtained by holding down the SHIFT key while typing the space bar.
2. For VIC-20s, the WIDTH in line 63800 should equal 22. For Commodore 64s, WIDTH should equal 40.

To use this subroutine, append it to a program and type in RUN 63800. Following the prompts, enter the line number for the first REMark statement and the character or characters you'd like to see repeated as a border. Then type in up to six lines of remarks, with each line having fewer characters than your screen width.

When you've entered your last REMark, respond to the next prompt by pressing RETURN. The program will then clear the screen and print a number of program lines, which will be REM statements containing a group of reverse-video T's. Notice that they're consecutively numbered from the starting line you selected above, and observe that your border entry has been repeated to fill an entire screen line.

If the lines look good to you, HOME your cursor and press RETURN once for each line. That enters them into your program. If you want more than one set of remarks, RUN 63800 again and enter a different starting line number. When you've entered your last remark, you can delete lines 63800 and up. Then, when you list your main program, your special REMarks will appear as described above, suitably impressing everyone who sees them and making them much easier to spot.

Despite its apparent complexity, the program is really quite simple. The computer interprets those reverse-video T's as DELetes. When they are listed, they wipe out what was printed

before them, namely the line numbers and REM keywords. If your printer responds to DELetes, your printed listings will be just like those on the screen; if it doesn't, you'll see the line numbers, the REMs, and maybe even the T's on your printed listings.

There are a few cautions to be observed when entering your borders and remarks. Since REMs can't contain SHIFTed characters, you can't use graphics or lowercase letters in your borders or your remarks. In addition, since INPUT statements won't handle commas or colons, you can't use those characters either, unless you put them in after the program lines are listed on the screen.

It's hard to delete or change your special REM lines, because their line numbers are invisible, but lines 63930-63990 solve the problem. When you RUN 63930, these lines go through all the REMs in memory, changing the DELetes to British pound signs (£). The process takes several seconds. But when it's finished, the REMs will list in the normal way and you can make all the changes you'd like. When you're finished, another RUN 63930 will restore the DELetes, and the line numbers and REMs will again be invisible.

This little program is a lot of fun to use, and it gives your listings a professional look as well as an air of mystery. I hope you will have as much fun with it as I have.

Remarkable REMs for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 REM ** REMARKABLE REMARK MAKER **          :rem 37
120 REM{4 SPACES}SEE REM STATEMENTS FOR        :rem 39
122 REM{2 SPACES}CHANGES FOR OTHER MACHINES:rem 35
124 REM                                        :rem 123
63800 WIDTH=40:SP$="{10 SPACES}":FORI=1TO2:SP$=SP$
      +SP$:NEXT                                :rem 193
63805 REM ** IN LINE 63800, SET WIDTH=NUMBER OF CH
      ARACTERS ON ONE SCREEN LINE              :rem 120
63810 INPUT"{CLR}1ST LN#";LN:LN=INT(LN):IFLN<0ORLN
      >63999THEN63810                          :rem 71
63815 REM ** IN LINES 63820-63860, ALL SPACES MUST
       BE SHIFTED SPACES                       :rem 226
63820 INPUT"{DOWN} BORDER{4 SHIFT-SPACE}{4 LEFT}";
      B$:FORJ=1TOWI/LEN(B$)+1:A$=A$+B$:NEXT
                                               :rem 195
63830 A$=LEFT$(A$,WI-1):RE$(1)=A$:RE$(8)=A$:rem 14
63840 FORI=2TO7:INPUT"{DOWN} REMARK{3 SHIFT-SPACE}
      {3 LEFT}";RE$(I)                         :rem 166
```

```
63850 IFLEN(RE$(I))>WITHENPRINT"{RVS}MAX"WI"{LEFT}
       CHARACTERS!":I=I-1:NEXT                    :rem 82
63860 IFRE$(I)="{SHIFT-SPACE}"THENRE$(I)=RE$(8):RE
      $(8)="":LL=LN+I-1:I=7:GOTO63880             :rem 254
63870 LL=LN+I:RE$(I)=LEFT$(SP$,WI/2-LEN(RE$(I))/2)
      +RE$(I)                                     :rem 222
63880 NEXT:PRINT"{CLR}";:LN=LN-1:FORI=1TO8:LN=LN+1
      :IFRE$(I)=""THENI=8:NEXT:END                :rem 250
63890 ND=7:IFLN>9THENND=8:IFLN>99THENND=9:IFLN>999
      THENND=10:IFLN>9999THENND=11                :rem 250
63900 A$="REM"+CHR$(34)+CHR$(34)+CHR$(20)+CHR$(18)
      +LEFT$("TTTTTTTTTTT",ND-1)                  :rem 32
63910 A$=A$+CHR$(146)+RE$(I):PRINTLN;A$:NEXT
                                                  :rem 28
63920 END                                         :rem 219
63930 SB=43:I=PEEK(SB)+256*PEEK(SB+1):REM ** RUN 6
      3930 TO HIDE OR UNHIDE                      :rem 233
63940 J=PEEK(I)+256*PEEK(I+1):IFPEEK(I+4)=143ANDPE
      EK(I+5)=34THENGOSUB63970                    :rem 99
63950 IFPEEK(J)=0ANDPEEK(J+1)=0THENEND            :rem 37
63960 I=J:GOTO63940                               :rem 81
63970 FORK=I+6TOI+17:IFPEEK(K)=20THENPOKEK,92:GOTO
      63990                                       :rem 136
63980 IFPEEK(K)=92THENPOKEK,20                    :rem 57
63990 NEXT:RETURN                                 :rem 100
```

Programming the 64's Function Keys

James Quinby

By using your Commodore's function keys, you can simplify many editing and programming tasks.

This program allows Commodore 64 users to program the eight function keys and to use them outside the realm of a BASIC program. This is especially useful during program development. For example, let's say you're writing an application program in BASIC. In the normal course of writing and debugging a program, you probably find that you enter certain commands many times. For instance, I find myself constantly typing PRINT PEEK, LIST, GOSUB, and RUN to name a few. This key definer will allow you to program the function keys so that any sequence of commands will be executed with a single keystroke.

How It's Done

The method used here was derived from a previous *COMPUTE!* article ("Programming VIC's Function Keys," by Jim Wilcox, November 1982). It uses a BASIC program to supply the key definitions and to load the necessary machine language.

The bulk of the machine language operates on the same principle as the "wedge." Every 1/60 second, the machine scans for an IRQ interrupt. You can take advantage of that by altering the standard vectored address to wedge in your own routine. That routine checks the last keystroke for f1 through f8, and if no function key was pressed it vectors back to the standard IRQ handler. If no function key is detected, then the appropriate key definition (ASCII string) is plucked from a table (which you set up using the program) and sent to the screen. The text will be placed at the current cursor location, so the user can either continue typing, hit RETURN, or have it executed automatically.

The remaining machine code provides a convenient means for activating and deactivating the function keys. This small routine can be invoked by a SYS, and it simply repoints the IRQ vector, wedging in or wedging out the new interrupt handler.

The new interrupt wedge is located at $C000, and the IRQ resetter is located at $02A7. Normally, both of these areas are un-

touched by BASIC. Function key definitions are located in a variable length table after the new interrupt wedge. Since the length of the table cannot be determined until after the keys have been defined, the program will let you know the highest used byte. This value is typically in the neighborhood of 49500.

Loading the Key Definition Table Program

You must first decide how to define your function keys. Key definitions are contained in lines 20-90 of the program listing. I suggest defining each key with some often-used BASIC command(s) that you don't like typing repetitively. Alternatively, choose some command sequence that is used infrequently but easily forgotten.

Guidelines for defining the keys are as follows:

1. Each key definition is subject to the standard DATA statement syntax rules.
2. Every definition must be enclosed within quotes.
3. The back arrow (←) can be used to cause an automatic RETURN when the function key is pressed. If you don't use the back arrow, you must hit RETURN after pressing the function key to cause the command to be entered. The key definitions are arranged in sequential order; the first DATA statement defines f1, the second defines f2, etc.
4. Null keys can be defined by using a null key definition DATA statement (that is, DATA " ").
5. If a REM appears within the quotes, it will appear on the screen when its key is entered.
6. Use the BASIC keyword abbreviations for long command sequences to squeeze more in.
7. All eight function keys must be defined, even if some are null.

Once you've chosen your key definitions, type in the attached program and replace statements 20 through 90 with your definitions. SAVE, and then RUN, the program. As the key definitions are being interpreted and saved, you will see them appear on the screen. During a slight pause, the two machine language programs will be POKEd and checksummed. A checksum error will halt the program and print an error message. In this case, you should double-check the DATA statements in lines 520-710.

Upon successful loading, the border should turn green, indicating that the function keys are active. Now try each function key, and its corresponding definition should appear on the

screen. If the last character in a definition is a ←, then the command will be executed. If not, you can continue typing to complete the command or hit RETURN yourself. Please note that you are now free to clear memory (type NEW) and enter any program you wish. The function keys will maintain their special operation.

Precautions

Wedging a routine into the IRQ interrupt process can be tricky business, especially if the pointers are reset from BASIC using POKEs. For that reason, the short machine language program at location $02A7 (679) is provided. If you should reset the computer using RUN/STOP-RESTORE, you'll notice that the function keys cease to function, since the system reset will restore the IRQ vector to its original value. However, you can easily reactivate the keys by calling the IRQ resetter with:

```
SYS 679
```

The border will once more turn green, indicating that the function keys are again active. A second SYS 679 will deactivate the keys and turn the border red. You must deactivate the keys if you load a different routine at $C000; failure to do so will probably lock up the computer.

A note on border colors: If you don't like green for keys-on and red for keys-off, you can change them to suit your own preference. Line 570 contains the standard color code for green (5) and line 590 contains that for red (2). Modify those codes however you wish. Beware, though, that the checksum for this DATA statement group will then have to be adjusted.

In the listing given here, the following functions are defined:

f1: LIST command; no autoRETURN
f2: RUN command; no autoRETURN
f3: LIST program to printer; no autoRETURN
f4: CLOSE printer after LIST; no autoRETURN
f5: In-memory program merge; no autoRETURN
f6: Second step in program merge; no autoRETURN
f7: PRINT PEEK; no autoRETURN
f8: Set screen colors; autoRETURN

Notice that I avoid autoRETURNs, since I don't trust my memory. I prefer to see each command before it is executed. If I hit the wrong function key, I can always delete the line.

The definitions for f1, f2, f7, and f8 are self-explanatory. The

definition for f3 contains the three commands, in abbreviated format, to OPEN the printer, CMD to it, and LIST a program. Typing f4 (shift-f3) will CLOSE the printer. The definition for f5 contains all the commands to perform an in-memory program merge, computing the ending address of the program currently in memory and resetting BASIC start-of-program pointers to that address. The commands in f6 (shift-f5) will again reset BASIC pointers, only this time to the start of the first program. Together, f5 and f6 let you combine, or merge, two programs by simply hitting two keys.

Since it's easy to define the function keys with this BASIC program, you may want to make several copies, each defining a different set of keys. For example, the set discussed here could be used to facilitate program editing functions. Another set could supply SID chip POKE locations and let you fill in the blanks where appropriate. The same thing could be used for the VIC-II chip locations, making sprite programming easier.

Programmable Function Keys for the 64

Refer to "The Automatic Proofreader" (Appendix I) before typing this program in.

```
10 REM----DEFINE-PF-KEYS-HERE---------     :rem 182
20 DATA"LIST"                              :rem 252
30 DATA"RUN"                               :rem 182
40 DATA"OP4,4:CMD4:LI":REM->LIST TO PRINTER
                                           :rem 146
45 REM IE: "OPEN4,4:CMD4:LIST"             :rem 215
50 DATA"PR4:CLO4":REM->CLOSE PRINTER AFTER LIST
                                           :rem 182
55 REM IE: "PRINT#4:CLOSE4"                 :rem 34
60 DATA"X=(PE(45)+256*PE(46))-2:HH=INT(X/256):LL=X
   -256*HH:PO43,LL:PO44,HH"                :rem 237
65 REM IE:"X=(PEEK(45)+256*PEEK(46))-2:HH=INT(X/25
   6):LL=X-256*HH:                         :rem 187
66 REM (CONT'D) ... POKE43,LL:POKE44,HH"   :rem 211
70 DATA"PO43,1:PO44,8":REM->RESET AFTER MERGE
                                           :rem 187
75 REM IE:"POKE43,1:POKE44,8"              :rem 132
80 DATA"PRINT PEEK("                       :rem 160
90 DATA"PO53280,2:PO53281,12:PO646,7{2 SPACES}:REM
   =>SET SCREEN COLORS<"                     :rem 3
100 REM ------------------------------------
                                           :rem 246
110 REM +++ C-64 FUNCTION KEY DEFINITION +++
                                           :rem 138
120 REM ------------------------------------
                                           :rem 248
```

```
130 REM---NOTE: IF YOU HIT 'RUN/STOP' 'RESTORE' YO
    U WILL LOSE THE PFKEYS!                    :rem 70
140 REM---TO GET THEM BACK WITHOUT RELOADING, ENTE
    R THE FOLLOWING:                           :rem 1
150 REM{2 SPACES}SYS 679                       :rem 31
160 GOSUB470: REM- DEFINE PFK ASC VALUES       :rem 48
170 LB=110:POKE251,LB:POKE253,LB:REM- SAVE FOR NEW
     INTERR. ROUTINE                           :rem 224
180 HB=192:POKE252,HB:POKE254,HB:REM-{5 SPACES}DIT
    TO                                         :rem 61
190 :                                          :rem 212
200 PRINT CHR$(147);"DEFINING F-KEYS AS FOLLOWS:"
                                               :rem 115
210 ADDR=(LB+256*HB)                           :rem 70
220 FOR PFK=1 TO 8                             :rem 165
230 READ PF$: REM- GET A PFKEY DEF.            :rem 83
240 PRINT"PF"PFK "="PF$                        :rem 25
250 LP=LEN(PF$)                                :rem 90
260 POKEADDR,FK(PFK):IFLP=0THEN330             :rem 104
270 :                                          :rem 211
280 FORI=1 TO LP:ADDR=ADDR+1                   :rem 128
290 C$=MID$(PF$,I,1):IFC$="←"THENC$=CHR$(13)
                                               :rem 179
300 POKEADDR,ASC(C$)                           :rem 152
310 NEXTI                                      :rem 28
320 :                                          :rem 207
330 ADDR=ADDR+1                                :rem 101
340 NEXT PFK                                   :rem 183
350 POKEADDR,0                                 :rem 62
360 :                                          :rem 211
370 GOSUB520: REM---LOAD IRQ RESETTER          :rem 208
380 IF CK<> 3958THENPRINT"{RVS}BAD CHECKSUM IN FIR
    ST GROUP OF DATA STMTS.":END               :rem 108
390 :                                          :rem 214
400 GOSUB620: REM---LOAD NEW I.H.              :rem 72
410 IF CK<> 14512THENPRINT"{RVS}BAD CHECKSUM IN SE
    COND GROUP OF DATA STMTS.":END             :rem 190
420 :                                          :rem 208
430 SYS679{2 SPACES}: REM---REPOINT IRQ VECTOR
                                               :rem 193
440 PRINT"{RVS}PF KEYS ACTIVATED; LAST BYTE USED"A
    DDR                                        :rem 209
450 END                                        :rem 112
460 :                                          :rem 212
470 REM---PFKEY ASCII VALUES SUBRTN            :rem 156
480 FK(1)=133: FK(2)=137: FK(3)=134: FK(4)=138
                                               :rem 246
490 FK(5)=135: FK(6)=139: FK(7)=136: FK(8)=140
                                                 :rem 6
```

```
500 RETURN                                          :rem 117
510 :                                               :rem 208
520 REM---POKE IRQ RESETTER                         :rem 139
530 CK=0                                            :rem 147
540 FORI=1TO41:READA:POKE678+I,A:CK=CK+A:NEXT
                                                    :rem 61
550 RETURN                                          :rem 122
560 DATA120,173,20,3,240,18,169,0,141,20,3,169,192
    ,141,21,3,169                                   :rem 58
570 DATA 5 : REM GREEN BORDER? CODE                 :rem 83
580 DATA141,32,208,76,206,2,169,49,141,20,3,169,23
    4,141,21,3,169                                  :rem 128
590 DATA 2 : REM RED BORDER CODE                    :rem 188
600 DATA141,32,208,88,96                            :rem 212
610 :                                               :rem 209
620 REM---SUBR. TO POKE NEW INTERRUPT{3 SPACES}HAN
    DLER                                            :rem 244
630 CK=0                                            :rem 148
640 FORI=49152TO49261:READB:POKEI,B:CK=CK+B:NEXT
                                                    :rem 230
650 RETURN                                          :rem 123
660 DATA165,2,240,51,160,0,177,251,32,91,192,176,4
    ,201,0,208,15,169,0,133,2                       :rem 135
670 DATA165,253,133,251,165,254,133,252,76,49,234,
    166,198,177,251,157,119,2                       :rem 194
680 DATA230,198,32,103,192,165,198,201,11,144,212,
    230,2,76,49,234,165,215,32                      :rem 208
690 DATA91,192,176,3,76,49,234,165,8,41,1,208,247,
    160,0,177,251,197,215,208                       :rem 185
700 DATA6,32,103,192,76,6,192,32,103,192,76,73,192
    ,201,133,144,6,201,141,176                      :rem 199
710 DATA2,56,96,24,96,230,251,208,2,230,252,96
                                                    :rem 13
```

Calculated GOTO for the VIC and 64

Louis Buscaslia-Zeppa

Commodore's dynamic keyboard lets you simulate input without ever touching a key. This routine, which runs on any VIC or 64, takes advantage of that feature to calculate GOTO statements within a BASIC program.

When using your VIC, you must type in complete commands before the computer will respond. Right?

Wrong. The VIC has a ten-character keyboard buffer (locations 631-640) that is used by the GET command, and you can POKE values into that buffer that can be used as if they were typed in.

This program lets you calculate a value, POKE it into the keyboard buffer, and PEEK into the buffer to use the value as a GOTO line number. The loop index X is used in line 25 to calculate the GOTO number. Line 30 or 35 prints GOTO and a line number on the screen. Note that Z, the GOTO line number, may be a numeric literal (line 30) or an expression (line 35). The GOTO command itself is printed in the screen color, making it invisible; to see what is happening, change the color indicated in the literal.

Lines 40-45 place three CURSOR UP commands (145) and a RETURN (13) into the keyboard buffer. POKE 198,4 tells the computer how many characters are in the buffer.

To execute the END statement in line 45, the computer skips a line, prints READY, sets the cursor, and goes looking to see what's in the buffer. Then it executes three CURSOR UPs, covering the GOTO statement, and one RETURN. The GOTO statement is then executed with the calculated line number. After the gone-to line has been executed, GOTO 50 (line 35) sends control back into the loop.

Calculated GOSUBs can also be used but require a slightly different approach. Using RETURN from the printed statement will not work, because it's not in the program itself. Instead, the statement must contain a GOTO that will jump back into the loop. For a demonstration, delete line 30 and remove the REM in line 35. Lines 200-240, along with line 35, will let you set up a calculated GOSUB.

VIC owners may be puzzled by the POKEs in line 10. They

are used to get screen colors when the program is run on the 64; on the VIC, however, they are meaningless and will be ignored by the computer. They can be removed with no ill effect.

Calculated GOTO for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 PRINT"{CLR}":POKE53281,1:POKE53280,3     :rem 88
20 FORX=0TO4                                :rem 229
25 Z=100+(X*10)                             :rem 238
30 PRINT"{WHT}GOTO";Z                        :rem 7
35 REM PRINT"{WHT}GOSUB";Z+100;":GOTO50"    :rem 74
40 POKE631,145                              :rem 243
41 POKE632,145                              :rem 245
42 POKE633,145                              :rem 247
44 POKE634,13                               :rem 196
45 POKE198,4:END                            :rem 171
50 FORY=1TO1000:NEXTY                        :rem 73
60 NEXTX:END                                 :rem 14
100 PRINT"{BLK}100":GOTO50                   :rem 91
110 PRINT"{BLK}110":GOTO50                   :rem 93
120 PRINT"{BLK}120":GOTO50                   :rem 95
130 PRINT"{BLK}130":GOTO50                   :rem 97
140 PRINT"{BLK}140":GOTO50                   :rem 99
200 PRINT"{BLK}200":RETURN                  :rem 159
210 PRINT"{BLK}210":RETURN                  :rem 161
220 PRINT"{BLK}220":RETURN                  :rem 163
230 PRINT"{BLK}230":RETURN                  :rem 165
240 PRINT"{BLK}240":RETURN                  :rem 167
```

PRINT AT for Commodore Computers

David Johnson

Many versions of BASIC have a statement called PRINT AT or PRINT @. With this routine you can simulate that command on the VIC or 64.

It is often convenient to print a message, such as SETTING UP MAP, at a particular location on the screen. For instance, you might want to begin that message on row 20, column 3. Using Commodore BASIC, the statement would look like this:

```
10 PRINT "{HOME}{20 DOWN}{3 RIGHT}SETTING UP MAP"
```

It works, but typing 20 cursor downs and 3 cursor rights is a lot of work. With several such statements, even the best program could become very bulky and hard to handle.

However, by using PLOT (one of the Kernal routines) and a very short machine language program, you can simulate the much shorter PRINT @ statement. Using PRINT AT, the above command would be entered like this:

```
10 PRINT @ 20,3;"SETTING UP MAP"
```

As you can see, PRINT @ is much easier to use.

Programs 1 and 2 provide you with a PRINT @ command for your computer. The programs reside in an unused area of RAM, so you can save or load from tape without disturbing them.

Here is how the programs work:

Line	
10	Read machine language from DATA statements and put into memory.
20-180	Provide a demonstration of the program.
1000	The two POKE statements set up the row (PR) and column (PC) positions, SYS679 executes the machine language program, and RETURN sends control back to the main body of the program.
1010	This is the machine language program in DATA statements.

Here is what the machine language program does:

```
LDX #0      ;set up the row coordinate
LDY #0      ;set up the column coordinate
CLC         ;clear carry flag, cause PLOT to
            ;position cursor
JSR $FFF0   ;jump to Kernal routine PLOT to
            ;position the cursor
RTS         ;return to BASIC
```

To use the routine in your own programs, add lines 10, 1000, and 1010 (renumber them if you wish). After reading in the machine language data, you can position the cursor at any time by setting the variable PR equal to the row to which you wish the cursor to move and setting PC equal to the desired column.

When using this subroutine, remember that rows and columns are numbered starting at the upper left corner of the screen, beginning with zero. For the VIC, the row limits (PR) are 0 and 22 and the column limits (PC) are 0 and 21. For the 64, row limits are 0 and 24 and column limits are 0 and 39.

On both computers, a PRINT statement that ends on the last column of a row will cause a carriage return and linefeed. To prevent this, place a semicolon at the end of the PRINT statement (as in line 110). However, this will not help if a PRINT statement ends in the lower right corner of the screen. In that case, a carriage return and linefeed will occur no matter what. To avoid the problem, simply do not print in the lower right corner of the screen.

Note that if the carry flag is set instead of cleared, the X and Y registers will contain the present position of the cursor. This may be useful in games or at other times when you want to find out exactly where the cursor is positioned. To do this, you can add this program:

```
SEC         ;set carry flag
JSR $FFF0   ;jump to PLOT and find cursor position
STX $02B0   ;store row in 688
STY $02B1   ;store column in 689
RTS         ;return to BASIC
```

To use this cursor locator and the PRINT @ program together, type in Program 3 which works on both the VIC-20 and 64. Line 3000 contains the DATA for the PRINT AT routine and line 3010 contains the cursor locator routine. Enter row and column numbers, following the prompts, and an asterisk will appear. When

the program prints out the current cursor position (line 110), the column number will be one more than what you typed in line 20. This is because the cursor has advanced one position after printing the asterisk.

Now that you have a PRINT @ statement to add to your BASIC, try experimenting with it. The sample program is very simple and does not begin to explore its capabilities.

Program 1. PRINT AT, VIC Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 FORM=679TO687:READA:POKEM,A:NEXTM          :rem 64
20 PRINTCHR$(147)                            :rem 221
30 FORT=1TO500:NEXT                          :rem 189
40 PR=10:PC=8:GOSUB1000                       :rem 97
50 PRINT"MIDDLE"                             :rem 229
60 FORT=1TO800:NEXT                          :rem 195
70 PR=0:PC=0:GOSUB1000                        :rem 43
80 PRINT"UPPER LEFT                          :rem 240
90 FORT=1TO800:NEXT                          :rem 198
100 PR=22:PC=10:GOSUB1000                    :rem 186
110 PRINT"LOWER RIGHT{LEFT}";                 :rem 66
120 FORT=1TO800:NEXT                         :rem 240
130 PR=0:PC=11:GOSUB1000                     :rem 138
140 PRINT"UPPER RIGHT";                      :rem 171
150 FORT=1TO800:NEXT                         :rem 243
160 PR=22:PC=0:GOSUB1000                     :rem 143
170 PRINT"LOWER LEFT";                        :rem 88
180 FORT=1TO800:NEXT                         :rem 246
190 GOTO20                                    :rem 53
1000 POKE680,PR:POKE682,PC:SYS679:RETURN      :rem 29
1010 DATA 162,0,160,0,24,32,240,255,96        :rem 56
```

Program 2. PRINT AT, 64 Version

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 FORM=679TO687:READA:POKEM,A:NEXTM          :rem 64
20 PRINTCHR$(147)                            :rem 221
30 FORT=1TO500:NEXT                          :rem 189
40 PR=12:PC=17:GOSUB1000                     :rem 147
50 PRINT"MIDDLE"                             :rem 229
60 FORT=1TO500:NEXT                          :rem 192
70 PR=0:PC=0:GOSUB1000                        :rem 43
80 PRINT"UPPER LEFT"                         :rem 240
90 FORT=1TO500:NEXT                          :rem 195
100 PR=24:PC=28:GOSUB1000                    :rem 197
110 PRINT"LOWER RIGHT";                      :rem 165
120 FORT=1TO500:NEXT                         :rem 237
```

```
130 PR=0:PC=29:GOSUB1000                          :rem 147
140 PRINT"UPPER RIGHT";                           :rem 171
150 FORT=1TO500:NEXT                              :rem 240
160 PR=24:PC=0:GOSUB1000                          :rem 145
170 PRINT"LOWER LEFT";                             :rem 88
180 FORT=1TO500:NEXT                              :rem 243
190 GOTO20                                         :rem 53
1000 POKE680,PR:POKE682,PC:SYS679:RETURN           :rem 29
1010 DATA 162,0,160,0,24,32,240,255,96             :rem 56
```

Program 3. Demonstration

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 FOR M=679 TO 700:READ A:POKE M,A:NEXT M :rem 50
20 INPUT"{CLR}ROW";PR:INPUT"COLUMN";PC      :rem 72
30 GOSUB 1000                              :rem 164
40 PRINT "*";                              :rem 154
50 GET A$:IF A$="" THEN 50                 :rem 237
100 GOSUB 2000                             :rem 211
110 PRINT"ROW,COLUMN=";PR;",";PC;           :rem 35
120 GET A$:IF A$="" THEN 120                :rem 73
130 GOTO 20                                 :rem 47
1000 POKE 680,PR:POKE682,PC:SYS 679:RETURN :rem 29
2000 SYS 690:PR=PEEK(688):PC=PEEK(689):RETURN
                                           :rem 214
3000 DATA 162,0,160,0,24,32,240,255,96      :rem 57
3010 DATA 0,0,56,32,240,255,142,176,2,140,177,2,96
                                           :rem 140
```

Fast Sort

Bill Pfeifer

It is often convenient to be able to sort lists of string data. With this program your VIC or 64 can make short work of long lists.

The machine language bubble-sort algorithm written for the Atari by Ronald and Lynn Marcuse (*COMPUTE!*, March 1982) is an unusually versatile and flexible utility. It can be run, with minor changes on Commodore computers. A version of this utility was presented for PET/CBM computers by Richard Mansfield in *COMPUTE!*, May 1982.

These routines will run on the Commodore 64 or any VIC-20. They incorporate information from the above articles and include additional routines to insure proper data entry and string manipulation.

The two critical prerequisites demanded by this utility are satisfied by these routines, namely: (1) equal-length records and (2) continuous files with no extraneous strings interrupting the individual records.

Using Fast Sort

The first few lines of the program are specific to disk-based or tape-based systems. Enter the appropriate lines. Line 5, for disk-based systems, stores the machine language (ML) part of the program in the cassette buffer. Skip over lines 10 and 15 and continue entering the program at line 20. If your system uses tape for data storage, begin entering the program at line 10. Lines 10 and 15 section off a portion of high memory for the sort, and readjust the pointers accordingly.

One word of caution for tape users: Lines 10 and 15 assume that there are no cartridges or programs present which use ML "wedges" (such as the VIC Super Expander or a Toolkit or Programmer's Aid). These wedges usually change the pointers at memory locations 51 and 55 from zero to some other value. If a wedge is present, lines 10 and 15 will probably have to be rewritten.

There is another option for 64 owners using tape: use line 5 instead of lines 10 and 15, but give ML the value 49152 (this is free RAM above BASIC ROM). Such a version will work on the 64, but will *not* run on the VIC.

After entering the lines which apply to your system, enter the program beginning with line 20, the BASIC loader. Line 25 defines the formulas used to POKE the string addresses into memory for use by the sort routine. Lines 30 to 60 comprise the ML program in the form of DATA statements.

Lines 65 and 70 initialize the record counter, input the file name and the number of letters per record, and set up a string of spaces equal to the length of the record.

Line 74 (or line 76) determines the maximum number of records that can be held in memory, based on the size of the records, CE. Line 74 is for the VIC-20 only; line 76 is for the Commodore 64. Use only the line that applies to your computer. Line 80 DIMensions A$ to the maximum number of records. You don't have to use every element, but the array is DIMensioned to handle them just in case.

The next section (lines 85 to 120) is a monitor which sees to it that all records entered into the array are of equal length. If a record is entered with less than the specified number of characters per entry (CE), the monitor adds the necessary number of spaces. If it is too long, the monitor tells you how many characters to delete and prints out the record so you can make adjustments using the screen editor. To get out of the data entry section, just hit RETURN following the next entry prompt. At that point, you would probably want to go to a menu (to select a CORRECT, REVIEW, or DELETE routine), but for this example, the program goes directly to the sort setup routine, lines 125 to 160.

The first function of the setup routine is to position the records into one continuous block, with no foreign bytes to foul things up. This should be done every time the sort routine is called, so that a changed array (with corrected, added, or deleted elements) is written into a fresh block of memory. Immediately following the array positioning, the array pointers are read and stored for use by the sort.

The routine's second function is to determine the sort OPTIONS, or the user's preferences as to how the array is arranged. First select an ascending (A-Z) or descending (Z-A) direction, then define the sort key. The sort key is that section of the record which will be considered in the actual sorting process. It can be the whole record or any portion of it. The default values are the first and last characters. Just hit RETURN to keep these values, or enter different numbers if you so desire. These parameters are then passed to the sort routine and the utility is called.

The next section, lines 165 and 170, prints out the sorted elements of the array, along with their element numbers. Again, you may prefer to go to a menu before printing the records.

Lines 175 to 190 comprise a minimenu to allow you to resort the records (using a different sort key, a different sort direction, or both) or to end the program.

The last section returns the memory pointers to their original values, if they were adjusted at the beginning of the program. This is not absolutely necessary, but it does return full RAM to BASIC when you are through with the program.

I suggest that you type in, save, and run this program before you incorporate it into a file program of your own. Enter a list of names and phone numbers, for example, and then sort the records using different values for the sort key. This will give you a feel for the flexibility and potential application of the sort utility and will help you in writing programs which use it to the best advantage.

With proper handling of string data before it is sorted, this machine language utility makes short work of long lists and is vastly superior to its BASIC counterpart. The slight extra attention required for its use is a small price to pay for the tremendous gain in speed and efficiency.

Fast Sort for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
5 M=PEEK(56):ML=828:REM THIS LINE FOR DISK SYSTEMS
  -RESUME CODE ENTRY AT LINE 20          :rem 228
10 M=PEEK(56):POKE51,140:POKE52,M-1:POKE55,140:POK
   E56,M-1                               :rem 66
15 POKE2,M:CLR:M=PEEK(2):ML=(M-1)*256+142:REM LINE
   S 10 & 15 FOR TAPE SYSTEMS            :rem 43
20 FORI=MLTOML+112:READZ:POKEI,Z:NEXT    :rem 196
25 DEFFNLM(X)=X-(INT(X/256)*256):DEFFNHM(X)=INT(X/
   256)                                  :rem 190
30 DATA169,0,133,80,133,81,162,1,165,249,133,251,1
   65,250,133,252,24,165                 :rem 209
35 DATA251,133,247,101,82,133,251,165,252,133,248,
   105,0,133,252,164,78,165              :rem 107
40 DATA2,240,10,177,251,209,247,144,44,240,12,176,
   19,177,251,209,247,144                :rem 12
45 DATA13,240,2,176,30,200,196,79,240,227,176,23,1
   44,223,169,1,133,80                   :rem 114
50 DATA164,82,136,177,251,72,177,247,145,251,104,1
   45,247,192,0,208,241,232              :rem 120
55 DATA224,0,208,2,230,81,228,253,208,172,165,254,
   197,81,208,166,165,80                 :rem 232
```

```
60 DATA201,0,208,144,96                          :rem 149
65 R$=CHR$(13):Z=-1:INPUT"{CLR}FILE NAME";F$:INPUT
   "{DOWN}# OF LETTERS/ENTRY";CE                  :rem 8
70 S$="":FORI=1TOCE:S$=S$+" ":NEXT               :rem 175
74 Q=INT(FRE(X)/(CE+7)): REM THIS LINE FOR COMMODO
   RE VIC-20 ONLY                                :rem 226
76 Q=INT((FRE(X)+65536)/(CE+7)): REM THIS LINE FOR
    COMMODORE 64 ONLY                            :rem 98
80 DIMA$(Q)                                      :rem 73
85 GOSUB90:ONXGOTO85,125                         :rem 239
90 X=1:PRINT"{DOWN}#"Z+2"OF"Q:X$="":INPUTX$:IFX$="
   "THENX=2:RETURN                               :rem 162
95 IFLEN(X$)<>CETHENGOSUB105                     :rem 240
100 Z=Z+1:A$(Z)=X$:RETURN                        :rem 251
105 CH=LEN(X$):IFCH<CETHENX$=X$+LEFT$(S$,(CE-CH)):
    RETURN                                       :rem 159
110 PRINT"{DOWN}{RVS}DELETE"CH-CE"LETTER(S){OFF}"
                                                 :rem 195
115 PRINT"{DOWN}#"Z+2:PRINT"{2 SPACES}"X$;:FORK=1T
    OCH+2:PRINT"{LEFT}";:NEXTK                   :rem 33
120 INPUTX$:CH=LEN(X$):ON-(CH<>CE)GOTO105:RETURN
                                                 :rem 24
125 PRINT"{DOWN}POSITIONING RECORDS..":FORI=0TOZ:A
    $(I)=A$(I):NEXT                              :rem 253
130 S1=PEEK(51):S2=PEEK(52):POKE247,S1:POKE248,S2
                                                 :rem 120
135 HD=S2*256+S1+CE:POKE82,CE:POKE249,S1:POKE250,S
    2                                             :rem 0
140 L=FNLM(HD):H=FNHM(HD):POKE251,L:POKE252,H:HD=Z
    +1                                           :rem 149
145 LO=FNLM(HD):HI=FNHM(HD):POKE253,LO:POKE254,HI:
    F=1:LL=CE                                    :rem 154
150 PRINT"{DOWN}{RVS}SORT OPTIONS:{OFF}":INPUT"
    {DOWN}{RVS}A{OFF}SCENDG/{RVS}D{OFF}ESCENDG";X$
    :U=1:IFX$="D"THENU=0                         :rem 65
155 INPUT"{DOWN}START KEY (CHAR#)";F:F=F-1:INPUT"
    {DOWN}END KEY (CHAR#)";LL:LL=LL-1            :rem 180
160 ON-(LL>CE)GOTO155:POKE2,U:POKE78,F:POKE79,LL:P
    RINT"{DOWN}SORTING..":SYSML                  :rem 87
165 FORI=0TOZ:PRINTI+1:PRINTA$(I)"{DOWN}"        :rem 189
170 FORT=1TO500:NEXT:NEXT                        :rem 107
175 PRINT"{DOWN}{RVS}1{OFF}-RE SORT;{RVS}2{OFF}-EN
    D"                                           :rem 117
180 GETX$:ON-(X$="")GOTO180                       :rem 25
185 IFX$="1"GOTO125                               :rem 44
190 IFX$<>"2"GOTO180                             :rem 103
195 IFM=PEEK(56)THENEND                          :rem 159
200 POKE51,0:POKE55,0:POKE52,M:POKE56,M:CLR:rem 99
```

Commodore Data Handling Workshop

John Fisher

Part 1. Super Shell Sort for the VIC and 64

Your Commodore computer is an efficient data handler. This article describes a sophisticated sorting system for the VIC (with at least 8K expansion) or the 64.

All sorting routines written in BASIC suffer the same limitation. At some point in the sorting process a substitution routine is invoked, and the value of one element is swapped with the value of another. This is illustrated in the following three lines:

```
100 WORK$ = A$(HI)
110 A$(HI) = A$(LO)
120 A$(LO) = WORK$
```

To do the swap, you have to move the contents of three strings. This movement, combined with the garbage collection, inevitably results in very slow sorts.

The machine language sorting routine presented here speeds up the substitution process by swapping the pointers to the strings instead of the strings themselves. Combined with the efficiency of the shell sort algorithm, that produces a very useful tool. This is particularly true since these programs will tailor the machine language to any usable memory location on the VIC-20 or the Commodore 64.

Program 1 is the driver program for the BASIC loader (Program 2). It first determines where you want the machine language routine located. You have three choices: at the top, at the bottom, or external to BASIC memory. On the VIC-20 place the code at the top of memory; however, if you have 3K expansion in addition to 8K or more of memory expansion, then you may want to locate it at the beginning of the 3K area (address 1024), external to BASIC memory. If you have a Commodore 64, consider locating the routine external to BASIC memory, starting at address 49152 (the 4K of RAM just beyond the BASIC ROM).

As written, the driver will load the second program from disk (device 8). To load from tape, change the assignment of DA in line 105 from 8 to 1 (1 is the device number for tape). For the autoload

feature to work properly with tape, you must save Program 2 immediately following Program 1 on the same tape, and you must leave the PLAY button depressed after Program 1 is loaded. VIC owners using tape will need to add two additional cursor-down characters in front of the RUN in line 260; Commodore 64 tape users should add one additional cursor down. For either disk or tape, be sure that the program name (PN$) in line 110 of Program 1 matches the name under which you saved Program 2.

Program 2 is the BASIC loader for the machine language sorting routine and will take a few moments to run. Using the address in locations 252 and 251 (placed there by Program 1), this program tailors the machine language (ML) to the location you requested. Before actually loading the ML into memory, the program performs a checksum against each line from 500 through 635. That should help to isolate any typing errors to a specific line.

As soon as the checksumming is completed, the ML is loaded into memory. To repeat this task each time you wanted to use this sort routine would be very wasteful. It would be much better to load only the ML, now that it has been tailored to your needs, and that is the function of Program 3. This program allows the ML to be saved to disk or tape. Then the code may be loaded at any time without the need for a BASIC loader. Note that to save the ML to tape, the first four numbers on line 1050 should be changed from 169,1,162,8 to 169,1,162,1. For Program 3 to work properly, you must *not* turn off or reset the computer between running Programs 1 and 2 and running Program 3.

Program 4 should show you how to put this ML to work. Note that it is necessary to modify Program 4 to reflect the location of the ML.

Immediately after running Program 3, type:

```
PRINT PEEK(251),PEEK(252)
```

and record the values. For example, if you have a VIC with 8K expansion and you located the ML at the top of memory, the values you should get are 64 and 62. For Program 4 to work properly, add the following line:

```
105 POKE 251,64:POKE 252,62
```

If you have a Commodore 64 and you located the ML in the free RAM above BASIC ROM, the values you should get when you PEEK locations 251 and 252 are 0 and 192, so you should add the following line to Program 4:

```
105 POKE 251,0:POKE 252,192
```

The values will vary for other configurations, but in any case, you should change line 105 to reflect the values for your ML. If you located the ML at the bottom of memory or external to BASIC memory, you must also delete line 120. This line should be used only if your ML is located at the top of memory.

Make sure that the program name in line 135 of Program 4 matches the name you specified when you saved the machine language with Program 3. Also, if you are loading the ML from tape instead of disk, you'll need to change the ,8,1 in line 135 to ,1,1. The value of N in line 150 may have to be changed, too, depending on the amount of memory available. On a 16K VIC-20, an array size of 1000 takes a few minutes to build and sort, so be patient.

Once the sort has completed, the program will wait until the f1 key is pressed. Then the sorted contents of the array will be displayed one screen at a time. You might have noticed that element zero of the array is not referenced. That is because the sort will not touch it, so it is available for your own use.

Lines 100-135 prepare the system and load the ML. Once the LOAD instruction is completed, the program restarts at line 100. To keep the program out of an infinite loop, a flag indicates if the ML has already been loaded. If so, then the program continues at line 145. Lines 145-190 build the array to be sorted. The subroutine at line 295 (called from 200) determines the following addresses:

S0 The address to place the ASCII value of the first character of the array name.

S1 The address to place the ASCII value of the second character of the array name, or a value of 128 if the array name is only one character in length.

S2 The address to locate the low byte of the number of elements to be sorted.

S3 The address to locate the high byte of the number of elements to be sorted.

S4 The address into which the completion code will be returned by the sort routine.

SRT The starting address of the sort routine.

Lines 200-220 handle preparation for the machine language sort, which is called in line 225. Lines 235 and 240 check the error code returned by the sorting routine. The possible error codes that might be returned are as follows:

0 No errors occurred.

1 The array could not be found (check the values placed in locations S0 and S1).

2 An attempt was made to sort a multidimensional array (for example, A$(x,y)).

Finally, lines 255-280 display the contents of the array.

To use this sorting routine to order your own data, you would substitute for N in line 150 the number of items of data you wish to enter, then delete lines 165-190 in Program 4 and add your own input routine. For example, you could use the following lines:

```
165 FOR I=1 TO N
170 PRINT "ITEM #";N;:INPUT A$
175 A$(I)=A$
180 NEXT
```

Program 1. Driver

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 REM SHELL SORT DRIVER PROGRAM          :rem 25
105 SL=448:DA=8                            :rem 70
110 PN$="SORT 2"                           :rem 79
115 REM MAIN MENU                         :rem 213
120 PRINT"{CLR}{2 SPACES}SHELL SORT SETUP":PRINT
                                           :rem 15
125 PRINT"RESERVE MEMORY FOR":PRINT"SORT ROUTINE A
    T:":PRINT                              :rem 84
130 PRINT"{2 SPACES}1-TOP"                :rem 182
135 PRINT"{2 SPACES}2-BOTTOM"             :rem 158
140 PRINT"{2 SPACES}3-EXTERNAL"            :rem 41
145 PRINT"{2 SPACES}4-EXIT"                 :rem 6
150 A=4:PRINT:PRINT                       :rem 214
155 INPUT" OPTION";A                      :rem 196
160 ON A GOTO 185,200,215,175             :rem 151
165 GOTO 155                              :rem 112
170 REM EXIT                              :rem 182
175 PRINT"{CLR}":END                       :rem 18
180 REM TOP OF MEMORY                     :rem 222
185 N=PEEK(56)*256+PEEK(55)-SL            :rem 168
190 M=N:GOTO 235                          :rem 127
195 REM BOTTOM OF MEMORY                  :rem 198
200 M=PEEK(44)*256+PEEK(43)-1              :rem 39
205 N=M+SL+1:GOTO 235                     :rem 162
210 REM OUTSIDE OF MEMORY                   :rem 2
215 M=0:PRINT:PRINT                       :rem 224
220 INPUT" ADDRESS";M                     :rem 246
225 IF M=0 THEN 120                       :rem 164
```

```
230 REM COMMON ROUTINE                          :rem 104
235 H1=INT(N/256):L1=N-H1*256                    :rem 85
240 H2=INT(M/256):L2=M-H2*256                    :rem 82
245 POKE 251,L2:POKE 252,H2                     :rem 180
250 REM LOAD COMMAND                            :rem 154
255 PRINT"{CLR}{3 DOWN}LOAD ";CHR$(34);PN$;CHR$(34
    );",";DA                                   :rem 163
260 PRINT"{4 DOWN}RUN{HOME}";                   :rem 240
265 FOR I=631 TO 633:POKE I,13:NEXT:POKE 198,3
                                                 :rem 8
270 REM FIX POINTERS                            :rem 216
275 IF A=1 THEN 290                             :rem 166
280 IF A=2 THEN 295                             :rem 168
285 NEW                                         :rem 137
290 POKE 56,H1:POKE 55,L1:NEW                   :rem 122
295 POKE 44,H1:POKE 43,L1:POKE N,0:NEW          :rem 140
```

Program 2. BASIC Loader

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 REM SHELL SORT BASIC LOADER                  :rem 78
105 DIM C3%(11),C2%(16),CK(27)                  :rem 176
110 REM READ DATA                               :rem 172
115 FOR I=0 TO 11:READ C3%(I):NEXT               :rem 61
120 FOR I=0 TO 16:READ C2%(I):NEXT               :rem 61
125 FOR I=0 TO 27:READ CK(I):NEXT                :rem 56
130 REM CHECK FOR ERRORS IN DATA                 :rem 75
135 FOR I=0 TO 27:CK=0                          :rem 119
140 FOR J=1 TO 16:READ A:CK=CK+A:NEXT            :rem 77
145 C1=CK(I)-CK:IF C1=0 THEN 160                :rem 158
150 PRINT"ERROR IN LINE";500+I*5                 :rem 83
155 C2=C2+1                                      :rem 30
160 NEXT                                        :rem 214
165 IF C2 THEN STOP                              :rem 21
170 RESTORE:FOR I=1 TO 57:READ A:NEXT           :rem 179
175 REM BASIC LOADER                            :rem 154
180 ML=PEEK(252)*256+PEEK(251):CL=ML             :rem 29
185 PRINT"{CLR}{DOWN}LOADING MACHINE LANGUAGE":PRI
    NT:PRINT                                    :rem 216
190 READ A                                      :rem 247
195 REM 3 BYTE OPCODE                           :rem 164
200 I=0                                          :rem 72
205 IF I>11 THEN 245                            :rem 217
210 IF A=C3%(I) THEN 220                        :rem 152
215 I=I+1:GOTO 205                              :rem 205
220 READ LO:READ HI:ADDR=HI*256+LO+ML           :rem 166
225 HI=INT(ADDR/256):LO=ADDR-HI*256              :rem 60
230 POKE CL+0,A:POKE CL+1,LO:POKE CL+2,HI        :rem 72
235 CL=CL+3:GOTO 290                             :rem 97
```

```
240 REM 2 BYTE OPCODE                                  :rem 154
245 I=0                                                 :rem 81
250 IF I>16 THEN 280                                   :rem 221
255 IF A=C2%(I) THEN 265                               :rem 169
260 I=I+1:GOTO 250                                     :rem 205
265 READ B:POKE CL+0,A:POKE CL+1,B                     :rem 125
270 CL=CL+2:GOTO 290                                    :rem 95
275 REM 1 BYTE OPCODE                                  :rem 161
280 POKE CL,A:CL=CL+1                                  :rem 182
285 REM CHECK LENGTH                                   :rem 163
290 LN=CL-ML:IF LN<448 THEN PRINT"{UP}";LN:GOTO 19
    0                                                  :rem 121
295 PRINT:PRINT"DONE ..."                              :rem 232
300 END                                                :rem 106
400 REM 3 BYTE OPCODES                                 :rem 236
410 DATA 32,76,78,109,110,140,141,172,173,205,237,
    238                                                 :rem 48
420 REM 2 BYTE OPCODES                                 :rem 237
430 DATA 16,41,48,105,133,144,145,160,162,165,169,
    176,177,197,201,208,240                             :rem 22
440 REM CHECKSUMS                                       :rem 34
450 DATA 0,793,1290,1723,1382,1337,1721,1078,1260,
    1206,1043                                          :rem 104
460 DATA 923,969,1259,1478,1534,1553,1306,1592,156
    7,1242                                             :rem 253
470 DATA 1652,1439,1451,1573,1700,1487,1829
                                                       :rem 103
490 REM MACHINE LANGUAGE DATA                          :rem 212
500 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0              :rem 67
505 DATA 0,0,0,173,0,0,41,127,141,0,0,173,1,0,9,12
    8                                                  :rem 152
510 DATA 141,1,0,169,0,141,4,0,165,47,133,71,165,4
    8,133,72                                            :rem 19
515 DATA 160,0,177,71,205,0,0,208,8,200,177,71,205
    ,1,0,240                                             :rem 7
520 DATA 42,160,2,177,71,141,5,0,200,177,71,141,6,
    0,24,165                                           :rem 13
525 DATA 71,109,5,0,133,71,165,72,109,6,0,133,72,1
    97,50,144                                          :rem 84
530 DATA 207,240,205,169,1,141,4,0,76,64,1,160,4,1
    77,71,201                                          :rem 68
535 DATA 1,240,8,169,2,141,4,0,76,64,1,24,165,71,1
    05,7                                               :rem 85
540 DATA 133,71,165,72,105,0,133,72,173,2,0,141,17
    ,0,173,3                                           :rem 10
545 DATA 0,141,18,0,173,18,0,208,12,173,17,0,240,4
    ,201,1                                            :rem 157
550 DATA 208,3,76,64,1,78,18,0,110,17,0,56,173,2,0
    ,237                                               :rem 83
```

```
555 DATA 17,0,141,15,0,173,3,0,237,18,0,141,16,0,1
    62,0                                          :rem 58
560 DATA 138,141,8,0,141,9,0,173,17,0,141,10,0,173
    ,18,0                                         :rem 112
565 DATA 141,11,0,238,8,0,208,3,238,9,0,173,9,0,20
    5,16                                          :rem 80
570 DATA 0,240,4,176,85,144,10,173,8,0,205,15,0,24
    0,2,176                                       :rem 223
575 DATA 73,238,10,0,208,3,238,11,0,160,3,165,71,1
    33,88,133                                     :rem 79
580 DATA 90,165,72,133,89,133,91,24,165,88,109,8,0
    ,133,88,165                                   :rem 217
585 DATA 89,109,9,0,133,89,24,165,90,109,10,0,133,
    90,165,91                                     :rem 104
590 DATA 109,11,0,133,91,136,208,223,32,65,1,173,7
    ,0,240,163                                    :rem 119
595 DATA 48,161,32,176,1,162,1,76,211,0,138,208,12
    9,76,148,0                                    :rem 144
600 DATA 96,160,0,140,7,0,177,88,141,12,0,177,90,1
    41,13,0                                       :rem 224
605 DATA 200,152,205,12,0,240,2,176,15,205,13,0,24
    0,25,144,23                                   :rem 143
610 DATA 169,1,141,7,0,76,175,1,205,13,0,240,2,176
    ,64,169                                       :rem 233
615 DATA 255,141,7,0,76,175,1,140,5,0,160,1,177,88
    ,133,92                                       :rem 241
620 DATA 200,177,88,133,93,172,5,0,136,177,92,141,
    14,0,140,5                                    :rem 130
625 DATA 0,160,1,177,90,133,92,200,177,90,133,93,1
    72,5,0,177                                    :rem 136
630 DATA 92,200,205,14,0,208,3,76,80,1,144,180,76,
    111,1,96                                      :rem 21
635 DATA 160,2,177,88,72,177,90,145,88,104,145,90,
    136,16,243,96                                 :rem 64
```

Program 3. ML Saver

Refer to "The Automatic Proofreader" (Appendix I) before typing this program in.

```
1000 REM MACHINE LANGUAGE SAVE                    :rem 13
1005 E0=PEEK(252)*256+PEEK(251)+448               :rem 82
1010 E1=INT(E0/256):E2=E0-E1*256                  :rem 191
1015 POKE 254,E1:POKE 253,E2                      :rem 217
1020 ML=680:NN=719                                :rem 235
1025 FOR J=0 TO 31:READ T:POKE ML+J,T:NEXT:rem 135
1030 PRINT"FILENAME TO USE":INPUT N$              :rem 162
1035 L=LEN(N$):POKE 2,L                            :rem 7
1040 FOR J=1 TO L:POKE NN+J,ASC(MID$(N$,J,1)):NEXT
                                                  :rem 46
1045 SYS ML:END                                   :rem 115
```

```
1050 DATA 169,1,162,8,160,1,32,186          :rem 130
1055 DATA 255,165,2,162,208,160,2,32        :rem 228
1060 DATA 189,255,169,128,133,157,169,251   :rem 250
1065 DATA 166,253,164,254,32,216,255,96     :rem 144
```

Program 4. Demonstration

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 REM SORT TEST                                   :rem 253
105 POKE 251,64:POKE 252,62                         :rem 137
110 ON FLAG GOTO 145                                 :rem 28
115 REM SET THE TOP OF MEMORY                       :rem 169
120 POKE 56,PEEK(252):POKE 55,PEEK(251):CLR
                                                    :rem 144
125 FLAG=1                                           :rem 32
130 REM LOAD SORTING SUBROUTINE                     :rem 206
135 LOAD "SORT/ML",8,1                              :rem 206
140 REM BUILD{2 SPACES}THE TEST ARRAY               :rem 137
145 CLR                                             :rem 123
150 N=100:DIM A$(N+1)                                :rem 38
155 TI$="000000"                                    :rem 253
160 PRINT"{CLR}BUILD":PRINT"ARRAY> ";TI$             :rem 47
165 FOR I=1 TO N                                     :rem 43
170 N1=INT(RND(0)*5+5):A$=""                        :rem 212
175 FOR J=1 TO N1                                    :rem 94
180 R1=RND(0)*26+65                                 :rem 230
185 A$=A$+CHR$(R1)                                  :rem 165
190 NEXT J:A$(I)=A$:NEXT I                          :rem 192
195 PRINT:PRINT "SORT"                              :rem 127
200 GOSUB 295                                       :rem 178
205 POKE S0,65:POKE S1,128                          :rem 148
210 N2=INT(N/256):N1=N-N2*256                        :rem 94
215 POKE S3,N2:POKE S2,N1                           :rem 146
220 PRINT:PRINT "BEGIN> ";TI$                       :rem 203
225 SYS SRT                                         :rem 145
230 PRINT:PRINT "DONE > ";TI$                       :rem 141
235 EC=PEEK(S4)                                      :rem 92
240 GOSUB 325                                       :rem 176
245 PRINT:PRINT "ERROR> ";E$                        :rem 159
250 GOSUB 310                                       :rem 171
255 FOR I=1 TO N STEP 40                            :rem 203
260 PRINT"{CLR}";                                    :rem 55
265 J=I+39:S=2                                        :rem 0
270 FOR K=I TO J STEP S:IF K>N THEN 290              :rem 55
275 PRINT A$(K),A$(K+1)                             :rem 181
280 NEXT K:GOSUB 310:NEXT I                          :rem 52
285 PRINT "{CLR}";                                   :rem 62
290 END                                             :rem 114
295 S0=PEEK(251)+PEEK(252)*256                      :rem 111
```

```
300 S1=SØ+1:S2=SØ+2:S3=SØ+3:S4=SØ+4          :rem 205
305 SRT=SØ+19:RETURN                          :rem Ø
310 PRINT"{HOME}{21 DOWN}F1 KEY>"            :rem 123
315 GET A$:IF A$<>CHR$(133) THEN 315          :rem 55
320 RETURN                                   :rem 117
325 ON EC+1 GOTO 335,340,345                 :rem 122
330 E$="UNKNOWN":RETURN                      :rem 202
335 E$="NONE":RETURN                         :rem 207
340 E$="NO ARRAY":RETURN                     :rem 183
345 E$="MULTI-DIMEN.":RETURN                 :rem 243
```

Part 2. Relative Files

This article examines how the relative file capability of the VIC 1540 or 1541 disk drive can be used to expand the potential of your VIC-20 or Commodore 64.

The 1540 owner's manual suggests that although the disk drive could use a relative file format, the VIC-20 could not. Actually, that is only partially correct. The owner's manual for the 1541 admitted that with a little more programming, relative files could be accessed. While the manual did provide some simple examples, it did not fully illustrate the strengths and weaknesses of relative files.

This might be a good time to describe some of the features and vocabulary of Commodore disk drives. One very common feature of Commodore drives is that they are intelligent devices. That is, they contain microprocessors with their own operating systems. Such operating systems are called the Disk Operating System (DOS). What makes the Commodore arrangement unusual, however, is that DOS resides in memory located within the disk drive unit itself. Most other manufacturers place DOS within the controlling computer. On computers with limited memory, that can be quite a disadvantage, but Commodore's design permits a more flexible approach.

All disk operations, such as file OPENs and CLOSEs, require that some commands be passed to DOS. Usually these commands are transparent to the programmer. With the relative file format, it is necessary to communicate directly with DOS from the program level.

Before getting into the programming of relative files, it may be helpful to describe some of their characteristics. A relative file is a

collection of records which have the following characteristics:

- Have a fixed maximum record length for the entire file, which is defined when the file is created.
- Have an absolute maximum record length of 254 bytes, including the mandatory carriage return as end-of-file marker. This value may not be exceeded, even at file-create time.
- Can be randomly accessed by informing DOS (via the command channel) which record number is to be accessed (for either input or output).
- Can be sequentially accessed, since the DOS automatically positions to the next record after a carriage return has been received or sent. Additionally, when the file is first opened, it is positioned at the first record in the file.

You may be thinking that these attributes remind you of an array on the disk. If so, you are correct. However, with the disk drive you can access far more data than could be maintained at one time in memory — and that is the basic strength and weakness of relative files. The strength is the amount of data that can be accessed, while the weakness is that accessing data requires communication with DOS from your BASIC program.

With that in mind, let's look at the next problem. On my disk drive, I learned through hard experience that the process of scratching (deleting) a disk file tends to leave DOS in chaos. If a relative file is then created or extended, the chances are that the relative file will walk all over the contents of the disk. All that is required to avoid this problem is that prior to creating or extending any relative file, you issue the UI or U9 command. This will reset DOS to the power-up condition. It is a quick and safe way to be certain that everything is in order within DOS. Prior to issuing the command, just be certain that all open files are closed, with the natural exception of the command channel.

Program 5 illustrates the steps necessary to create a relative file. The U9 command in line 30 insures that DOS is in its power-up condition. The ,L added to the filename in line 50 tells DOS that this is to be a relative file. The variable LN is used to assign the record length. Line 60 shows how the CHR$ equivalent of the length must be added to the filename under which the file is opened. Once the file has been opened with this record length, the length cannot be changed.

Lines 110-130 create 100 records for the file, consisting simply of the word RECORD and a number from 1 to 100. Notice that

although the record length was specified to be 25 characters, it is not necessary to use all 25. Trying to use more than 25, however, would cause a DOS Error 51, Overflow In Record.

To store your own data in a relative file, you would replace these lines with your own input routine. You might read the data from the keyboard via INPUT statements, read it from an array, or perhaps even read it from DATA statements.

The subroutine at line 200 is used to check for error messages from DOS. Error number 0 indicates that no problems were encountered. Error 73, DOS Mismatch Error, is also ignored because it can sometimes appear after the power-up vector (U9). Note that the DOS messages are read on file number 1, which was opened with a secondary address of 15, which is the DOS command channel.

Once the record size has been specified and the relative file opened, DOS is able to build the necessary structures to randomly access any of the records in the file. To select a particular record, it is necessary to send the P (Position) command to DOS. The P command has the following format:

PRINT#*file*,"P";CHR$(*sa*+96);CHR$(*lo*);CHR$(*hi*);
CHR$(*off*)

where:

- *sa* is the secondary address on which the relative file has been opened. DOS expects that 96 be added to this value.
- *lo* is the low byte of the record number.
- *hi* is the high byte of the record number.
- *off* is the optional offset within the record at which the next I/O request is to start.

Program 6 demonstrates this by reading through the file in reverse sequence. Lines 10-90 are the same as in Program 5. Line 110 sets up the decreasing loop, and line 120 converts the record number to high-byte/low-byte format. Line 140 illustrates the Position command to select the proper record, and line 160 shows how the selected record is read. Note that in line 140 we are sending the command to file number 1, because that is the file which is opened with a secondary address of 15, the DOS command channel. The 2+96 in the P command is because the relative file was opened in line 80 with a secondary address of 2. In line 160, we are reading the data from file number 2, which is the channel open to the relative file itself.

Extending a relative file is as simple as loading the file. The

only difference is that the program must first position to the record number to be output. If this record does not already exist, then a DOS Error 50 (Record Not Present) will be generated. That is quite acceptable, since the file is being extended.

Once the file is positioned, the output sequence may continue as normal. Program 7 should help to clarify this process.

To allow for quick access to any record within the file, DOS maintains sector blocks. Figure 1 presents the layout of a single directory entry. Figure 2 is a layout for a single sector block. Note that each sector block contains the track and sector for all of the six possible sector blocks, which allows rapid access to the necessary data. Essentially, the data within these sector blocks are the contents of the first two bytes of each sector within the file. That is the link information to the next sector. Thus, DOS can rapidly access the proper position in the disk without a sequential read-through of the file.

This accounts for the speed of relative files. Naturally, from the sector level, DOS is capable of calculating the offset to the proper record.

There is one additional consideration when using relative files. Due to the overhead in DOS, if a relative file is open, then only one sequential file may be open at the same time. The file may be opened for Read or Write processing. This is probably the greatest limitation of the 1540 and 1541 relative file format. There are ways around it, but they tend to be cumbersome.

Speed and random access do not come without some cost. In particular, these limitations are:

- Fixed length records, which tends to waste your disk space.
- A maximum of 720 records per file.
- A maximum of 254 bytes per record, including the carriage return which must be present as an end of file marker. All values greater will result in DOS Error 51 (Overflow In Record).
- One additional sector is required for every 120 blocks within the relative file. These extra sectors are used as the sector blocks for the file.
- Only one other file may be open on the disk, as long as the relative file remains open.

In spite of these limits, relative files are well justified, due to their speed and potential.

Figure 1. A Single Directory Entry

Byte	Description
00	File Type: $80 — Deleted $81 — Sequential $82 — Program $83 — User $84 — Relative
1-2	Track and sector of the first data block within the file.
3-18	Filename
19-20	Relative file *only*: Track and sector of the first sector block for the file.
21	Relative file *only*: Record size including the carriage return. Maximum 254 bytes.
22-27	Unused for relative files.
28-29	Number of blocks (sectors) in the file. This is stored in low-byte/high-byte format.

Figure 2. A Single Sector Block

Byte	Description
0-1	Track and sector to the next sector block for the file.
2	Current sector block number. All values are within 0 and 5 .
3	Record size (maximum 254 bytes).
4-15	Track and sector (2 bytes each) of all of the sector blocks for the relative file.
16-255	Track and sector information (2 bytes each) for 120 sectors within this file. Note the first sector block contains the first 120 pointers, the second the next 120 sector pointers, etc.

Program 5. Relative Files

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 REM RELATIVE FILE WRITER                    :rem 158
20 REM{2 SPACES}OPEN COMMAND CHANNEL           :rem 112
30 OPEN 1,8,15,"U9":GOSUB 200                    :rem 6
40 REM BUILD FILENAME                          :rem 249
50 FI$="RELFILE,L,":LN=25                      :rem 184
60 FI$=FI$+CHR$(LN)                             :rem 32
70 REM OPEN{2 SPACES}RELATIVE FILE             :rem 249
80 OPEN 2,8,2,FI$                              :rem 109
90 GOSUB 200                                   :rem 123
100 REM WRITE THE FILE                           :rem 1
110 FORI=1 TO 100                              :rem 100
120 PRINT#2,"RECORD";STR$(I);CHR$(13);         :rem 194
130 NEXT I                                      :rem 28
140 REM CLOSE FILES                             :rem 98
```

```
150 CLOSE 2:CLOSE 1:END                          :rem 48
200 REM CHECK FOR DISK ERRORS                    :rem 195
210 INPUT#1,EN,ET$,ET,ES                         :rem 168
220 IF EN=0 OR EN=73 THEN 240                    :rem 195
230 PRINT EN;ET$;ET;ES:STOP                      :rem 212
240 RETURN                                       :rem 118
```

Program 6. Reading in Reverse Sequence

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 REM RELATIVE FILE READER                      :rem 116
20 REM{2 SPACES}OPEN COMMAND CHANNEL             :rem 112
30 OPEN 1,8,15,"U9":GOSUB 200                     :rem 6
40 REM BUILD FILENAME                            :rem 249
50 FI$="RELFILE,L,":LN=25                        :rem 184
60 FI$=FI$+CHR$(LN)                               :rem 32
70 REM OPEN{2 SPACES}RELATIVE FILE               :rem 249
80 OPEN 2,8,2,FI$                                :rem 109
90 GOSUB 200                                     :rem 123
100 REM READ THE FILE BACKWARDS                   :rem 36
110 FORI=100TO1STEP-1                            :rem 254
120 HI=INT(I/256):LO=I-HI*256                    :rem 146
130 REM POSITION TO THE DESIRED RECORD            :rem 48
140 PRINT#1,"P";CHR$(2+96);CHR$(LO);CHR$(HI);
                                                  :rem 16
150 GOSUB 200                                    :rem 168
160 INPUT#2,A$                                    :rem 13
170 PRINT " ";I;A$:NEXT I                         :rem 79
180 REM CLOSE FILES                              :rem 102
190 CLOSE 2:CLOSE 1:END                           :rem 52
200 REM CHECK FOR DISK ERRORS                    :rem 195
210 INPUT#1,EN,ET$,ET,ES                         :rem 168
220 IF EN=0 OR EN=73 THEN 240                    :rem 195
230 PRINT EN;ET$;ET;ES:STOP                      :rem 212
240 RETURN                                       :rem 118
```

Program 7. Extending a Relative File

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 REM EXTENDING RELATIVE FILES                  :rem 186
20 REM OPEN COMMAND CHANNEL                      :rem 112
30 OPEN 1,8,15,"U9":GOSUB 300                     :rem 7
40 REM BUILD FILENAME                            :rem 249
50 FI$="RELFILE,L,":LN=25                        :rem 184
60 FI$=FI$+CHR$(LN)                               :rem 32
70 REM OPEN RELATIVE FILE                        :rem 249
80 OPEN 2,8,2,FI$                                :rem 109
90 GOSUB 300                                     :rem 124
100 REM READ THE LAST RECORD                     :rem 101
```

```
110 PRINT#1,"P";CHR$(2+96);CHR$(100);CHR$(0);
                                              :rem 162
120 GOSUB 300                                 :rem 166
130 PRINT#2,"UPDATED";CHR$(13);                :rem 25
140 REM EXTEND BEYOND RECORD #100             :rem 117
150 PRINT#1,"P";CHR$(2+96);CHR$(200);CHR$(0);
                                              :rem 167
160 PRINT#2,"EXTENDED";CHR$(13);              :rem 102
170 PRINT#1,"P";CHR$(2+96);CHR$(1);CHR$(0);:rem 72
180 GOSUB 300                                 :rem 172
190 REM READ BACK EXTENDED FILE                :rem 28
200 FORI=1 TO 200                             :rem 101
210 INPUT#2,A$                                  :rem 9
220 PRINT " ";I;A$                            :rem 137
230 NEXT I                                     :rem 29
240 REM CLOSE FILES                            :rem 99
250 CLOSE 2:CLOSE 1:END                        :rem 49
300 REM CHECK FOR DISK ERRORS                 :rem 196
310 INPUT#1,EN,ET$,ET,ES                      :rem 169
320 IF EN=0 OR EN=50 OR EN=73 THEN 340        :rem 155
330 PRINT EN;ET$;ET;ES:STOP                   :rem 213
340 RETURN                                    :rem 119
```

Part 3. Searching for Data

Question: How do you retrieve data that you have entered into your computer? That may sound silly — after all, if you put it in you should be able to get it out — but experience has shown that data retrieval can be one of the biggest problems for programmers.

It is possible to lose information, but it is also possible to make your computer work hard to find it.

One of the easiest approaches, of course, is to read information sequentially until you find some sort of match (for example, looking at the last names in a name and address file). This is an excellent approach for a limited amount of information because the programming involved is quite simple. Unfortunately, when the number of records you have to read increases, so does the time it takes to do the necessary search operations and comparisons.

The next logical step in this process might be to divide the file into smaller units as the amount of data increases. Again, this is

actually a viable approach as long as the time it takes to find any given record does not exceed tolerable limits.

This division forces some structure onto the data that you are storing, thus adding to the complexity of the programming. But the fact that you are adding structure does not necessarily imply that the program should greatly increase in size. In fact, it is usually quite simple to reduce the structure to a very unassuming algorithm. For example, assume that you want to place all records beginning with the letters A through D into one file, those starting with E through H into a second file, and so on. Start by matching letters with numbers:

A – 0	E – 4	I 8	M – 12
B – 1	F – 5	J – 9	N – 13
C – 2	G – 6	K – 10	O – 14
D – 3	H – 7	L – 11	P – 15

Note that all you have done is arrange the letters in the same sequence that you'll store them in a file. Note, too, that if you divide any of those numbers by four and ignore the remainder, the result will be a value between zero and six. To express it in BASIC:

```
110 FI = INT(X/4)
```

The next problem is to convert each letter into one of the 26 values. Looking up the ASCII values of the letters A through Z, you'll discover that they have respective values of 65 through 90. To place those values within our required range, simply subtract 65 from the ASCII value. If the record resides in LN$, then your BASIC code might read:

```
100 X = ASC(LN$) - 65
```

That works great until someone uses a strange character, such as a space, at the beginning of the string. But there is a way out of that problem too. All you have to do is make a slight adjustment to the program. The results might be as follows:

```
100 X = ASC(LN$) - 65
110 FI = INT(X/4) + 1
120 ON FI GOTO 150,200,250,300,350,400,450
130 PRINT "INVALID LAST NAME..."
140 STOP
```

As you can see, by adding one to the value obtained in line 110, you are able to use some simple BASIC constructs to allow for human error. Note that this approach permits most tape files to expand to surprisingly large proportions.

Yet even this approach rapidly reaches the time limit barrier. Splitting the file into smaller and smaller segments soon requires more programming overhead than the technique returns to the user. However, if you are willing to force a little more structure onto your data, you can increase the speed even more. In particular, if the data are sorted, it becomes possible to apply the binary search algorithm to the data. This is true regardless of the storage media; the only difference is that the relative file format of the Commodore disk lends itself quite well to a binary search.

Why is it called a binary search? Because the algorithm works by attempting to split the file into two portions — one that might contain the record, and one that can *not* contain it. This continues until the record is located or no match is found. The middle element of the file is checked first; if a match is found, the search completes. If not, that record is checked to see if it is higher or lower in value than the target. If it is higher, the upper half of the file is eliminated from the search. Next, a new middle point (within the lower portion of the file) is chosen. Again, the check is made to determine if the record has been found. If not, the splitting continues until the record is located.

This might seem like a great deal of work, but in the worst case a file containing 512 records would require only *nine* comparisons to locate the record or determine that it does not exist. The way this can be calculated is

```
100 CC = LOG(RC) / LOG(2)
```

where RC is the record count and CC is the comparison count. This is opposed to reading the file sequentially, which would require, on the average, that half of the total records be compared. Using the previous example, that would be an average of 256 comparisons to find a single record. Thus the binary search would only require approximately 4 percent of the comparisons that a sequential search requires.

Obviously, by applying just a little bit of structure to your file, it is possible to rapidly and accurately access any of the records within the file.

Program 8 shows how to apply these techniques to an array in memory. It generates an array of 512 elements (lines 170-220), then randomly chooses a possible element (lines 230-260) and searches the array for a match.

Program 9 illustrates how the binary search may be applied to relative files. The example program builds a relative file, selects a

random element, and searches for it. The program as presented creates a new file each time it is run. To do multiple searches on the same file, delete line 220 after the first time the program is run.

As an example of how to modify Program 9 to work on your own data files, you can use it to search the file you created with Programs 6 and 7 from Part 2. Change the filename in line 180 from BINFILE to RELFILE, and change the value of the record length (LN) in line 120 from 5 to 25. You'll need to set the highest record number (HR) in line 120 to 100 for the original file created with Program 6, or to 200 if you extended the file with Program 7. Next, delete lines 210-260 and add these lines:

```
210 REM SELECT RECORD TO SEARCH FOR
220 INPUT "RECORD TO SEARCH FOR";N
230 TR$="RECORD" + STR$(N)
```

You'll discover that there are some records that the binary search won't be able to find. The reason is that this file is in numerical, rather than alphabetical, order. For example, if you try a binary search for RECORD 6, the algorithm will try to find that record between RECORD 59 and RECORD 60, which is where RECORD 6 should appear alphabetically. This illustrates the importance of correctly sorting any files on which you wish to perform a binary search.

Perhaps you can now see how this three-part article ties together. To effectively manage a large assortment of data such as a mailing list, a recipe file, or an inventory record, you would enter the data into memory, sort it into order (as shown in Part 1), and store it in a relative file (as shown in Part 2). You could then retrieve any item of the data whenever required simply by using the binary search technique described here.

Program 8. Binary Search

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 REM BINARY SEARCH                          :rem 240
110 REM DEFINE VARS.                           :rem 139
120 LR=1:HR=512                                 :rem 72
130 RC=HR-LR                                   :rem 203
140 MR=INT(RC/2)+LR                            :rem 108
150 TC=LOG(HR)/LOG(2)                          :rem 203
160 TC=INT(TC+.999)                             :rem 66
170 REM BUILD ARRAY                            :rem 107
180 PRINT"{CLR}BUILDING ARRAY":PRINT           :rem 145
190 DIM A$(HR)                                 :rem 196
200 FOR I=1 TO HR                              :rem 109
210 A$(I)=RIGHT$(("{2 SPACES}"+STR$(I*2)),4)
                                               :rem 245
```

```
220 NEXT I                                          :rem 28
230 REM RANDOM NUMBER GENERATION                   :rem 239
240 TR$="{2 SPACES}"+STR$(INT(RND(0)*HR*2)):rem 59
250 TR$=RIGHT$(TR$,4)                              :rem 187
260 PRINT"TARGET=";TR$:PRINT                        :rem 57
270 REM BEGIN BINARY SEARCH                         :rem 93
280 MC=MC+1                                         :rem 83
290 PRINT "LO=";A$(LR),                            :rem 255
300 PRINT "HI=";A$(HR),                            :rem 233
310 PRINT "MR=";A$(MR)                             :rem 209
320 IF A$(MR)=TR$ THEN 460                          :rem 73
330 IF A$(MR)>TR$ THEN 370                          :rem 75
340 REM MOVE UP LR                                 :rem 245
350 LR=MR:GOTO 390                                  :rem 33
360 REM MOVE DOWN HR                               :rem 134
370 HR=MR                                           :rem 16
380 REM SET UP NEXT SEARCH                           :rem 5
390 RC=HR-LR:IF RC=1 THEN 430                      :rem 101
400 MR=INT(RC/2)+LR                                :rem 107
410 GOTO 280                                       :rem 104
420 REM RECORD DOES NOT EXIST                      :rem 226
430 PRINT:PRINT TR$;" NOT FOUND"                   :rem 161
440 GOTO 470                                       :rem 108
450 REM RECORD FOUND                               :rem 184
460 PRINT:PRINT TR$;" FOUND"                       :rem 179
470 PRINT"AFTER";MC;"ATTEMPTS":PRINT                :rem 97
480 PRINT"MAX. OF";TC;"ATTEMPTS"                   :rem 217
490 END                                            :rem 116
```

Program 9. Searching Relative Files

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 REM BINARY SEARCH FOR RELATIVE FILES  :rem 166
110 REM DEFINE VARS.                      :rem 139
120 LR=1:HR=512:LN=5                      :rem 142
130 RC=HR-LR                              :rem 203
140 MR=INT(RC/2)+LR                       :rem 108
150 TC=LOG(HR)/LOG(2)                     :rem 203
160 TC=INT(TC+.999)                        :rem 66
170 REM OPEN RELATIVE FILE                 :rem 42
180 FI$="BINFILE,L,"+CHR$(LN)             :rem 129
190 OPEN 1,8,15,"U9":GOSUB 700             :rem 66
200 OPEN 2,8,2,FI$:GOSUB 700              :rem 232
210 REM CREATE FILE                        :rem 75
220 GOSUB 600                             :rem 170
230 REM RANDOM NUMBER GENERATION          :rem 239
240 TR$="{2 SPACES}"+STR$(INT(RND(0)*HR*2)):rem 59
250 TR$=RIGHT$(TR$,LN-1)                  :rem 127
260 PRINT"TARGET=";TR$:PRINT               :rem 57
```

```
270 REM BEGIN BINARY SEARCH                    :rem 93
280 MC=MC+1                                    :rem 83
285 RN=LR:GOSUB 500                           :rem 105
290 PRINT "LO=";A$,                            :rem 16
295 RN=HR:GOSUB 500                           :rem 102
300 PRINT "HI=";A$,                           :rem 254
305 RN=MR:GOSUB 500                            :rem 99
310 PRINT "MR=";A$                            :rem 225
320 IF A$=TR$ THEN 460                         :rem 89
330 IF A$>TR$ THEN 370                         :rem 91
340 REM MOVE UP LR                            :rem 245
350 LR=MR:GOTO 390                             :rem 33
360 REM MOVE DOWN HR                          :rem 134
370 HR=MR                                      :rem 16
380 REM SET UP NEXT SEARCH                      :rem 5
390 RC=HR-LR:IF RC=1 THEN 430                 :rem 101
400 MR=INT(RC/2)+LR                           :rem 107
410 GOTO 280                                  :rem 104
420 REM RECORD DOES NOT EXIST                 :rem 226
430 PRINT:PRINT TR$;" NOT FOUND "             :rem 161
440 GOTO 470                                  :rem 108
450 REM RECORD FOUND                          :rem 184
460 PRINT:PRINT TR$;" FOUND "                 :rem 179
470 PRINT"AFTER";MC;"ATTEMPTS":PRINT           :rem 97
480 PRINT"MAX. OF";TC;"ATTEMPTS"              :rem 217
490 CLOSE 2:CLOSE 1:END                        :rem 55
500 REM READ A RECORD                         :rem 149
510 HI=INT(RN/256):LO=RN-HI*256                :rem 67
520 PRINT#1,"P";CHR$(2+96);CHR$(LO);CHR$(HI);
                                               :rem 18
530 GOSUB 700                                 :rem 175
540 A$=""                                     :rem 127
550 GET#2,B$:IF ST THEN 570                   :rem 156
560 A$=A$+B$:GOTO 550                          :rem 64
570 GET#2,B$                                   :rem 99
580 RETURN                                    :rem 125
600 REM BUILD ARRAY                           :rem 105
610 PRINT"{CLR}CREATING FILE":PRINT            :rem 47
620 FOR I=1 TO HR                             :rem 115
630 T$="{2 SPACES}"+STR$(I*2)                 :rem 208
640 T$=RIGHT$(T$,LN-1)+CHR$(13)               :rem 191
650 PRINT#2,T$;                                :rem 92
660 NEXT I                                     :rem 36
670 RETURN                                    :rem 125
700 REM CHECK FOR DISK ERRORS                 :rem 200
710 INPUT#1,EN,ET$,ET,ES                      :rem 173
720 IF EN=0 OR EN=73 THEN 740                 :rem 205
730 PRINT EN;ET$;ET;ES:STOP                   :rem 217
740 RETURN                                    :rem 123
```

Chapter 5

Graphics and Sound

Creating Multicolor Graphics on the VIC

Daryl Biberdorf

Multicolor characters add excitement to your VIC programs, and this article will help you understand how they work. A multicolor character editor is included too. These programs will run on any VIC.

Multicolor characters can be a puzzling, if not frustrating, part of the VIC-20's graphics features. But once you understand the basics, multicolor graphics are much easier to handle. This article introduces you to the subject and presents a character editor to aid in designing multicolor characters.

A multicolor character is, quite simply, a character which can be made up of as many as four colors at the same time. Such characters can add a great deal of excitement to graphics displays. There is a price to be paid, however: In a multicolor character, each pixel doubles in size. That halves horizontal resolution, but it provides the extra bit needed to define the character's colors.

The pattern of bits within the pair determines the color displayed by the pair. If both bits are on, that pixel's color will be the auxiliary color, set in the auxiliary color register (36878). If the first bit is on and the second is off, the color will be that of the screen border (location 36879). If the first bit is off and the second is on, the color will be determined by the contents of the color memory location for the character's screen position (38400 to 38905). If both bits are off, the color will be that of the background, also set in location 36879.

The auxiliary color is determined in location 36878, the same one used as a volume control for the VIC's sound. The upper four bits (7-4) control the auxiliary color, while the lower four bits (3-0) determine volume. The following commands will maintain a constant volume and let you set the auxiliary color to whatever you choose:

```
POKE 36878, N*16 OR (PEEK(36878) AND 15)
```

N is a number chosen from Table 1 for the color you want.

Table 1. VIC Auxiliary Color Codes

Black	0	Orange	8
White	1	Light Orange	9
Red	2	Pink	10
Cyan	3	Light Cyan	11
Purple	4	Light Purple	12
Green	5	Light Green	13
Blue	6	Light Blue	14
Yellow	7	Light Yellow	15

The character color of a multicolor character is determined by the character's specific location on the screen. This color is controlled by locations 38400 to 38905, the area for character color memory. Table 2 lists eight numbers with a corresponding color for each. By POKEing one of these numbers into the appropriate location in color memory, the character color is set for multicolor mode.

Table 2. VIC Screen Location Color Codes For Multicolor Mode

Black	8	Purple	12
White	9	Green	13
Red	10	Blue	14
Cyan	11	Yellow	15

The character's screen (transparent) and border colors are controlled jointly by location 36879. This register is the easiest one to set, mainly because all of the work has already been done for you. In Appendix F, all of the possible screen and border colors are listed. Just POKE your chosen color combination into location 36879.

It is possible to design multicolor characters by hand, but it's easier to let the computer do the tedious work for you. The multicolor character editor described here was designed to do just that, and it is especially helpful because it lets you visualize your characters as you create them.

When typing in the character editor program, you should use the abbreviated PRINT command (?) in all lines that contain PRINT statements. Otherwise, those lines may exceed the maximum allowable line length of 88 characters.

When the character editor is run, a menu will appear in the upper left of the screen, and a grid representing an enlarged version of the character you are editing will form in the upper right. The character being edited will appear normal size in the lower

left. In the lower right you'll get a list of the four different pixel patterns and their associated colors.

The I, M, J, and K keys move the yellow cursor over the grid. To turn on a pixel, press the + key. To turn off a pixel, press the – key. Press the back-arrow key to clear the grid and clear out any characters in memory. To select a character to edit, press the S key and use the + and – keys to page through the 64 characters available to you. If you forget which character you are editing, press the W key, which acts as a toggle between the two character sets. Press W again to return to the edit mode.

The D option lets you save the character being edited as a data file on cassette. When you choose this option, the prompt SURE? will appear in the upper left corner of the screen. Answer Y or N. The L option lets you load that data file back into memory for further editing. Again, the SURE? prompt will appear and should be answered Y or N. Press Q to abort the program.

To incorporate a custom character into your own programs, first SAVE the character file with the D command. Then LOAD Program 2, RUN it, and answer all prompts. It loads the previously created data file and converts it into a program that can be saved in the usual manner. The program fragment thus created can be read into your BASIC program with the following line:

```
xxxx READN:IFNTHENFORN=NTON+7:READX:POKEN,
X:NEXT:GOTOxxxx
```

The four x's represent a line number. The x's at the end of the line must be the same as those at the beginning.

Program 1. Multicolor Characters for the VIC

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
30 POKE52,28:POKE56,28:CLR                    :rem 20
40 FORI=7168TO7679:POKEI,PEEK(I+25600):NEXT:GOSUB
   {SPACE}29000:POKE650,128                   :rem 229
70 GETK$:IFK$="I"THENUD=-1:GOSUB28000         :rem 207
80 IFK$="M"THENUD=1:GOSUB28000                 :rem 30
90 IFK$="J"THENLR=-1:GOSUB28100                :rem 79
100 IFK$="K"THENLR=1:GOSUB28100                :rem 75
110 IFK$="+"THENGOSUB28200                    :rem 231
120 IFK$="-"THENGOSUB28200                    :rem 234
130 IFK$="D"THENGOSUB26000                    :rem 254
140 IFK$="S"THENGOSUB27000:GOSUB24000         :rem 191
150 IFK$="←"THEN155                            :rem 62
152 GOTO 160                                  :rem 104
155 FORI=CMTOCM+7:POKEI,0:NEXT:FORI=1TO8:FORJ=1TO8
    :CD%(I,J)=0:NEXTJ,I:GOSUB31000            :rem 100
```

```
160 IFK$="Q"THENPRINT"{CLR}":POKE36869,240:PRINT"
    {BLU}";:POKE650,0:END                          :rem 11
180 IFK$="W"THENOO=-OO:GOSUB23000                  :rem 243
190 IFK$="L"THENGOSUB22000                          :rem 8
200 GOTO70                                         :rem 50
22000 INPUT"{HOME}{RVS}{BLU}SURE";A$:IFA$="N"THENP
      RINT"{HOME}{RVS}{10 SPACES}";:RETURN         :rem 98
22010 POKE36869,240:INPUT"{CLR}{2 DOWN}FILENAME";F
      $:PRINT"{DOWN}{RED}INSERT TAPE AND":PRINT"PR
      ESS {RVS}SPACE"                              :rem 191
22020 WAIT197,32,0:OPEN1,1,0,F$:INPUT#1,CM,CH:FORI
      =1TO8:LK=0:FORJ=1TO8:                        :rem 21
22030 INPUT#1,X:CD%(I,J)=X:LK=LK+X:NEXTJ:POKECM-1+
      I,LK:NEXTI:CLOSE1                             :rem 6
22040 GOSUB30000:RETURN                            :rem 133
23000 IFOO=-1THENPOKE36869,240:POKECC,6             :rem 62
23010 IFOO=1THENPOKE36869,255:POKECC,15             :rem 72
23020 RETURN                                       :rem 215
24000 FORI=1TO8:FORJ=1TO8:CD%(I,J)=0:NEXTJ,I:GOSUB
      31000:W=7722:FORJ=0TO7:LK=0:FORI=0TO7
                                                   :rem 173
24010 IF(PEEK(CM+J)AND(2↑I))THENPOKEW-I,209:CD%(J+
      1,8-I)=2↑I                                    :rem 14
24020 NEXTI:W=W+22:NEXTJ:RETURN                     :rem 17
26000 INPUT"{HOME}{BLU}{RVS}SURE";A$:IFA$="N"THENP
      RINT"{HOME}{RVS}{10 SPACES}";:RETURN:rem 102
26007 POKE36869,240:INPUT"{CLR}{BLU}{DOWN} FILENAM
      E";F$                                        :rem 183
26008 PRINT"{DOWN}{RED} INSERT TAPE AND":PRINT" PR
      ESS {RVS}SPACE":WAIT197,32,0:OPEN1,1,1,F$:
                                                    :rem 10
26010 PRINT#1,CM:PRINT#1,CH:FORI=1TO8::FORJ=1TO8:P
      RINT#1,CD%(I,J):NEXTJ,I                       :rem 68
26020 CLOSE1:POKE 36869,255:GOSUB30000:RETURN
                                                   :rem 169
27000 POKE36879,26:GETK$:IFK$="+"THENCH=CH+1:IFCH>
      63THENCH=0                                    :rem 58
27010 IFK$="-"THENCH=CH-1:IFCH<0THENCH=63          :rem 166
27020 IFK$=CHR$(13)THEN27040                        :rem 24
27030 POKECL,CH+128:POKECC,6:GOTO27000             :rem 244
27040 POKECL,CH:CM=7168+8*CH:POKE36879,SC:RETURN
                                                   :rem 189
28000 VP=VP+UD:PRINT"{HOME}";:FORI=1TOVP-UD:PRINT"
      {DOWN}";:NEXT                                 :rem 87
28005 IFCD%(VP-UD,HP)THENPRINTTAB(12+HP)"{RVS}
      {BLK}Q":GOTO28020                            :rem 235
28010 PRINTTAB(12+HP)"{BLK}{RVS}."                 :rem 234
28020 IFVP<1THENVP=8                               :rem 232
28025 IFVP>8THENVP=1                               :rem 239
```

```
28030 PRINT"{HOME}";:FORI=1TOVP:PRINT"{DOWN}";:NEX
      T                                          :rem 13
28035 IFCD%(VP,HP)THENPRINTTAB(12+HP)"{RVS}{YEL}Q"
      :GOTO28050                                 :rem 57
28040 PRINTTAB(12+HP)"{RVS}{YEL}."               :rem 251
28050 RETURN                                     :rem 223
28100 HP=HP+LR:PRINT"{HOME}";:FORI=1TOVP:PRINT"
      {DOWN}";:NEXT                              :rem 123
28120 IFCD%(VP,HP-LR)THENPRINTTAB(12+HP-LR)"{RVS}
      {BLK}Q":GOTO28140                          :rem 188
28130 PRINTTAB(12+HP-LR)"{RVS}{BLK}."            :rem 184
28140 IFHP<1THENHP=8                             :rem 207
28143 IFHP>8THENHP=1                             :rem 212
28145 PRINT"{HOME}";:FORI=1TOVP:PRINT"{DOWN}";:NEX
      T                                          :rem 20
28147 IFCD%(VP,HP)THENPRINTTAB(12+HP)"{RVS}{YEL}Q"
      :GOTO28160                                 :rem 63
28150 PRINTTAB(12+HP)"{RVS}{YEL}."               :rem 253
28160 RETURN                                     :rem 225
28200 IFK$="+"THEN28205                          :rem 214
28203 GOTO 28210                                  :rem 53
28205 PRINT"{HOME}";:FORI=1TOVP:PRINT"{DOWN}";:NEX
      T:PRINTTAB(12+HP)"{RVS}{YEL}Q":CD%(VP,HP)=2↑
      (8-HP)                                     :rem 167
28210 IFK$="-"THENPRINT"{HOME}";:FORT=1TOVP:PRINT"
      {DOWN}";:NEXT:PRINTTAB(12+HP)"{RVS}{YEL}.":C
      D%(VP,HP)=0                                :rem 56
28220 LK=0:FORI=1TO8:LK=LK+CD%(VP,I):NEXT:POKECM-1
      +VP,LK:POKECL,CH:POKECC,15:RETURN          :rem 75
29000 HP=1:VP=1:PN=209:PF=174:FORI=1TO8:FORJ=1TO8:
      DC%(I,J)=0:NEXTJ,I:CL=8057:CC=38777        :rem 145
29010 CH=0:CM=7168:SC=27:OO=1                    :rem 141
30000 PRINT"{CLR}":POKE36869,255:GOSUB31000:rem 70
30010 PRINT"{HOME}{16 DOWN}{2 SPACES}{RVS}{GRN}[A]
      *[S]":PRINT"{2 SPACES}{RVS}- -":PRINT"
      {2 SPACES}{RVS}[Z]*[X]"                    :rem 84
30020 PRINT"{HOME}{DOWN}{RVS}{3 SPACES}{CYN}MENU":
      PRINT"{RVS} {RED}EEEEEEEE"                 :rem 49
30030 PRINT"{RVS}{BLK} I=UP":PRINT"{RVS}{CYN} M=DO
      WN":PRINT"{RVS}{RED} J=LEFT":PRINT"{RVS}
      {PUR} K=RIGHT"                             :rem 188
30040 PRINT"{RVS}{BLU} +=DOT ON":PRINT"{RVS}{PUR}
      {SPACE}-=DOT OFF"                          :rem 202
30045 PRINT"{RVS}{RED} W=WHICH"                   :rem 2
30050 PRINT"{RVS}{BLK} S=SELECT":PRINT"{RVS}{YEL}
      {SPACE}D=DATA":PRINT"{RVS}{GRN} ←=WIPE"
                                                 :rem 29
30055 PRINT"{RVS}{BLU} L=LOAD":PRINT"{RVS}{CYN} Q=
      QUIT"                                      :rem 53
```

```
30060 PRINT"{DOWN}"SPC(9)"{RVS}{BLK}.Q{RED}={BLK}C
      HARACTER":PRINTSPC(9)"{RVS}Q.{RED}={BLK}BORD
      ER"                                    :rem 217
30070 PRINTSPC(9)"{RVS}..{RED}={BLK}SCREEN":PRINTS
      PC(9)"{RVS}QQ{RED}={BLK}AUXILIARY":POKECL,CH
      :POKECC,15                             :rem 82
30080 GOSUB24000:RETURN                      :rem 139
31000 PRINT"{HOME}":FORI=0TO7:PRINTTAB(13)"{RVS}
      {BLK}........{BLU}":NEXT:RETURN        :rem 228
```

Program 2. Listing Generator

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
40 PRINT"{CLR}{DOWN}{RIGHT}{RED}INSERT TAPE AND":P
   RINT" PRESS {RVS}SPACE":WAIT197,32,0    :rem 247
50 LR$="9000 READN:IFNTHENFORN=NTON+7:READX:POKEN,
   X:NEXT:GOTO9000":PP$="9010 DATA"        :rem 53
60 OPEN1,1,0:INPUT#1,CM:PP$=PP$+STR$(CM):INPUT#1,C
   H:FORI=1TO8:N=0:FORJ=1TO8:INPUT#1,X:    :rem 145
70 N=N+X:NEXT:PP$=PP$+","+STR$(N):NEXT:CLOSE1:PRIN
   T"{CLR}{5 DOWN}"LR$:PP$=PP$+",0"        :rem 150
80 PRINTPP$:FORI=1TO10:READA:POKE630+I,A:NEXT:POKE
   198,10:END                              :rem 49
90 DATA19,78,69,87,13,13,13,13,13,13       :rem 30
```

Super Expander Graphics

Dave Needham

The Super Expander cartridge adds exciting graphics commands to the VIC-20's repertoire. These entertaining routines demonstrate some of those commands and can be customized to produce a variety of effects.

Here are two short programs that take advantage of some of the capabilities of Commodore's Super Expander for the VIC-20.

The first program draws a three-dimensional box and rotates it through 90 degrees, using BASIC's trigonometric functions. Line 20 sets the trig parameters, in radians, and lines 30-95 define the corners of the front face of the cube. The front face rotates around the coordinates 350,550; the rest of the cube tags along behind.

The second program demonstrates the Super Expander's drawing abilities, generating a spiral by continually expanding small segments of a circle. You can change the size and density of the spiral by varying the numbers in line 10.

Program 1. Rotating Box

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 GRAPHIC2:COLOR0,3,1,1                          :rem 147
20 FORA=.8TO2.34STEP.07                           :rem 104
30 X1=350+ COS(A)*212                              :rem 34
40 X2=350- COS(A)*212                              :rem 38
50 Y1=550+ SIN(A)*212                              :rem 44
60 Y2=550- SIN(A)*212                              :rem 48
70 X3=350+ SIN(A)*212                              :rem 45
80 Y3=550+ COS(A)*212                              :rem 44
90 X4=350- SIN(A)*212                              :rem 50
95 Y4=550- COS(A)*212                              :rem 53
97 SCNCLR                                          :rem 53
100 DRAW1,X1,Y2TOX4,Y4TOX2,Y1TOX3,Y3TOX1,Y2
                                                   :rem 240
110 DRAW1,X1,Y2TO X1+100,Y2-100TO X4+100 ,Y4-100TO
    X4,Y4                                           :rem 4
120 DRAW1,X3,Y3TO X3+100, Y3-100 TO X2+100, Y1-100
    TOX2,Y1                                         :rem 1
130 DRAW1,X4+100, Y4-100 TO X2+100, Y1-100 :rem 59
140 DRAW1,X1+100, Y2-100 TO X3+100, Y3-100 :rem 58
900 FORM=1TO100:NEXT                               :rem 232
910 NEXTA                                           :rem 26
```

Program 2. Expanding Spiral

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
2 GRAPHIC2:REGIONØ                          :rem 144
5 M=Ø                                       :rem 239
1Ø FOR I=3Ø TO 7ØØ STEP 5                   :rem 22Ø
3Ø CIRCLE1,5ØØ,5ØØ,I,I,M,M+5                  :rem 4
4Ø M=M+5:IFM=1ØØTHENM=Ø                     :rem 1Ø4
5Ø NEXTI                                    :rem 237
```

Multicolor Sprites for the Commodore 64

Gary Robinson

With this graphics utility, it's easy to create impressive multicolor sprites on your Commodore 64.

While it is entirely possible to construct multicolored sprites using pencil and paper, why anyone would want to is a mystery to me. It's far easier to let the computer do it for you.

The accompanying program is a sprite definition utility for the Commodore 64. It lets you create a multicolored sprite on the screen and it also generates the numbers needed to define your sprite block for later use in BASIC programs.

Multicolored sprites on the Commodore 64 work in much the same manner as single-colored sprites. However, there are a couple of differences. Obviously, each multicolored sprite can contain up to four colors. In addition, the horizontal resolution of multicolored sprites is half that of a single-colored sprite. However, once constructed, both multicolored and single-colored sprites are activated and moved in the same way.

Multicolored sprites have a lower horizontal resolution because each sprite element is defined by a pair of bits rather than by a single bit. Each bit-pair can take on one of four values (00, 01, 10, or 11), and each pair defines a given element color. Bit-pair 00 is the screen background, or transparent, color. Pair 01 is defined by the contents of sprite multicolor register 0 (53285, $D025). Pair 10 is defined by the contents of the appropriate sprite color register (53287–53294, $D027–$D02E), and pair 11 is defined by the contents of sprite multicolor register 1 (53286, $D026).

To use the multicolor sprite editor to create your own sprites, type in and run the program. The first thing you will see on your screen is a blue rectangle. The area within this rectangle is the sprite definition area where you will define the shape and color of the sprite. The area to the right will be used to display both the sprite you create and the numbers defining the sprite block.

Below the sprite definition area, three numbered blocks appear. Initially, the number 2 is displayed with a reverse field. This indicates that any element defined will be white. A different element color may be selected by pressing f7.

Once the cursor appears in the sprite definition area (it takes

a few seconds), the cursor control keys may be used to position the cursor. The cursor will exhibit full wraparound characteristics. Pressing the space bar will cause the element under the cursor to take on the selected color. The cursor will then advance one element in the direction of last motion. If this motion seems awkward, change line 530 to a REM statement to suppress any cursor movement after a SPACE. The cursor control keys and SPACE keys will repeat if held down. To erase an element, use SHIFT and SPACE simultaneously.

As mentioned before, f7 is used to select the current color from the three available. The colors actually used are controlled by the 1, 2, and 3 keys. Successive use of either of these keys will cause the corresponding color block to rotate through the 16 available colors. This operation also alters the contents of the sprite color registers, and thus alters the color of the displayed sprite. By combining f7 and the 1, 2, and 3 keys, you have full control over which color goes where in your sprite.

Once you have defined the sprite, press f1. The cursor will disappear, and a series of digits will appear to the right of the sprite definition area. These 63 numbers, taken rowwise, can be used in a BASIC DATA statement to set up a sprite block. Once they have been generated, the cursor will return and further operations may continue. Function f3 is used to display the sprite you have created. Function f5 will alternately expand or shrink the displayed sprite.

While the sprite is being displayed, its color composition may be altered with the 1, 2, and 3 keys. While this action keeps the sprite color consistent with the current color selection, the colors within the sprite definition area will not be altered. To update the color distribution within the sprite definition area, use f8. When f8 is invoked, the cursor will disappear and each element within the sprite definition area will be updated to reflect the current color set. When you're through, press E to exit the program.

In summary, the procedure used to create a multicolored sprite with this utility is as follows:

1. Select the colors to be used with the 1, 2, and 3 keys.
2. Define the shape of the sprite within the sprite definition area. Use f7 to select colors.
3. Use f1 to generate the data for the sprite block.
4. Then use f3 to display the sprite you have created.
5. Modify the colors of the displayed sprite as desired with the 1, 2, and 3 keys.

6. Update the colors within the sprite definition area by using f8.
7. Return to step 2 if further modification is desired.

Below is a list of control functions:

SPACE/ SHIFT SPACE	Sets or resets an element within the sprite definition area
f1	Generates sprite block data
f3	Displays the generated sprite
f5	Expands/shrinks displayed sprite
f7	Selects current color
f8	Updates colors in sprite definition area
1	Rotates color block #1 — Sprite multicolor register 0 (53285, $D025)
2	Rotates color block #2 — sprite color register (53287-53294, $D027-$D02E)
3	Rotates color block #3 — Sprite multicolor register 1 (53286, $D026)
E	Ends the editor program

While this program does not save sprite blocks on tape or allow multiple sprite definitions, it does let you quickly and easily get a sprite up on the screen.

Multicolor Sprite Editor for the 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
5 DIM V%(63):POKE 53280,6:L=1106                    :rem 156
10 PRINT "{CLR}":CO(1)=0:CO(2)=1:CO(3)=3:CC=2
                                                    :rem 44
20 FOR I=1066 TO 1089:POKE I,111:POKE I+880,119
                                                    :rem 198
21 NEXT                                             :rem 162
25 FOR I=55338 TO 55362:POKE I,14:POKE I+880,14:NE
   XT                                               :rem 78
30 FOR I=1105 TO 1905 STEP 40:POKE I,106:POKE I+25
   ,116:NEXT                                        :rem 159
40 FOR I=55377 TO 56177 STEP 40:POKE I,14:POKE I+2
   5,14:NEXT                                        :rem 186
41 POKE 56260,1:POKE 56262,0:POKE 56263,0:POKE 562
   67,1:POKE 56269,1                                :rem 43
42 POKE 56270,1:POKE 56274,1:POKE 56276,3:POKE 562
   77,3                                             :rem 105
43 POKE 1988,49:POKE 1990,160:POKE 1991,160:POKE 1
   995,178:POKE 1997,160                            :rem 27
44 POKE 1998,160:POKE 2002,51:POKE 2004,160:POKE 2
   005,160                                          :rem 217
50 POKE 2040,192:POKE 2041,193                      :rem 27
55 FOR I=31106 TO 31929:POKE I,0:NEXT               :rem 4
```

```
60 FOR I=12288 TO 12414:POKE I,0:NEXT      :rem 254
70 FOR I=12288 TO 12311 STEP 3:POKE I,1:POKE I+1,1
   28:NEXT                                  :rem 64
75 POKE 53264,2:POKE 53250,17:POKE 53251,138
                                            :rem 42
76 POKE 53271,0:POKE 53277,0                :rem 199
80 POKE 12297,254:POKE 12300,254:POKE 12298,127:PO
   KE 12301,127                             :rem 228
90 POKE 53287,14:POKE 53288,1:POKE 53285,0:POKE 53
   286,3                                    :rem 158
100 X=40:Y=66:POKE 53248,X:POKE 53249,Y     :rem 174
110 POKE 53269,3:F=1                         :rem 23
120 GET K$:IF K$="" THEN 120                 :rem 93
121 IF K$="E" THEN PRINT "{UP}":POKE 53269,0:END
                                            :rem 200
122 IF K$="{F3}" THEN 2000                  :rem 139
125 IF K$="{F1}" THEN 1000                  :rem 140
126 IF K$="{F5}" THEN 3000                  :rem 145
127 IF K$="{F7}"THEN 4000                   :rem 148
128 IF K$="1" THEN 5000                      :rem 63
129 IF K$="2" THEN 6000                      :rem 66
130 IF K$="3" THEN 7000                      :rem 60
131 IF K$="{F8}" THEN 9000                  :rem 152
138 IF K$<>"{RIGHT}" THEN 200                :rem 54
140 X=X+16:IF X>224 THEN X=40:Y=Y+8:IF Y> 226 THEN
     Y=66                                   :rem 234
150 POKE 53248,X:POKE 53249,Y:F=1            :rem 50
160 GOTO 120{14 SPACES}                      :rem 99
200 IF K$<>"{LEFT}" THEN 300                :rem 173
210 X=X-16:IF X<40 THEN X=216:Y=Y-8:IF Y< 66 THEN
    {SPACE}Y=226                            :rem 233
220 POKE 53248,X:POKE 53249,Y:F=2            :rem 49
230 GOTO 120                                 :rem 97
300 IF K$<>"{DOWN}" THEN 400                 :rem 35
310 Y=Y+8:IF Y>226 THEN Y=66:X=X+16:IF X> 224 THEN
     X=40                                   :rem 233
320 POKE 53248,X:POKE 53249,Y:F=3            :rem 51
330 GOTO 120                                 :rem 98
400 IF K$<>"{UP}" THEN 500                  :rem 165
410 Y=Y-8:IF Y<66 THEN Y=226:X=X-16:IF X< 40 THEN
    {SPACE}X=216                            :rem 235
420 POKE 53248,X:POKE 53249,Y:F=4            :rem 53
430 GOTO 120                                 :rem 99
500 L=1106+5*(Y-66)+(X-40)/8                :rem 127
510 IF K$<>" " THEN 520                      :rem 24
515 POKE L+54272,CO(CC):POKE L+54273,CO(CC):POKE 3
    0000+L,CC                               :rem 217
516 POKE L,160:POKE L+1,160:GOTO 530        :rem 185
520 IF ASC(K$)<>160 THEN 530                :rem 149
```

```
525 POKE L,32:POKE L+1,32:POKE L+30000,0  :rem 121
530 ON F GOTO 140,210,310,410             :rem 137
540 GOTO 120                              :rem 101
1000 N=1                                  :rem 125
1010 POKE 53269,0                          :rem 86
1020 PRINT "{HOME}{DOWN}"                 :rem 184
1050 FOR I=1 TO 63:V%(I)=0:NEXT           :rem 165
1100 FOR I=1106 TO 1906 STEP 40            :rem 10
1200 FOR J=0 TO 2                          :rem 54
1300 FOR K=0 TO 3                          :rem 57
1400 V5=PEEK(30000+I+8*J+2*K):V%(N+J)=V%(N+J)+V5*2
     ↑(6-2*K)                             :rem 115
1500 NEXT                                   :rem 5
1600 NEXT                                   :rem 6
1605 FOR M=0 TO 2                          :rem 66
1610 A$(M)="{2 SPACES}":IF V%(N+M)> 9 THEN A$(M)="
      "                                   :rem 209
1620 IF V%(N+M)>99 THEN A$(M)=""            :rem 77
1630 NEXT{17 SPACES}                        :rem 9
1640 PRINT TAB(27)A$(0)+STR$(V%(N))TAB(29)A$(1)+ST
     R$(V%(N+1));                         :rem 70
1650 PRINT TAB(33)A$(2)+STR$(V%(N+2))     :rem 223
1700 N=N+3                                :rem 255
1800 NEXT                                   :rem 8
1850 POKE 53269,1                          :rem 99
1900 GOTO 120                             :rem 150
2000 POKE 53269,0:POKE 53276,2             :rem 36
2100 FOR I=1132 TO 1142                    :rem 98
2200 FOR J=0 TO 800 STEP 40                :rem 61
2300 POKE I+J,32                           :rem 67
2400 NEXT                                   :rem 5
2500 NEXT                                   :rem 6
2600 FOR I=1 TO 63:POKE 12351+I,V%(I):NEXT  :rem 5
2700 POKE 53269,2                          :rem 95
2800 GOTO 110                             :rem 149
3000 POKE 53271,2+2*(PEEK(53271)=2)        :rem 17
3100 POKE 53277,2+2*(PEEK(53277)=2)        :rem 30
3200 GOTO 120                             :rem 145
4000 CC=CC+1:IF CC>3 THEN CC=1             :rem 76
4005 IF CC=1 THEN POKE 1988,177:POKE 2002,51:GOTO
     {SPACE}120                           :rem 20
4006 IF CC=2 THEN POKE 1988,49:POKE 1995,178:GOTO
     {SPACE}120                           :rem 50
4007 POKE 1995,50:POKE 2002,179:GOTO 120   :rem 99
4010 GOTO 120                             :rem 145
5000 CO(1)=CO(1)+1:IF CO(1)>15 THEN CO(1)=0
                                          :rem 183
5100 POKE 56262,CO(1):POKE 56263,CO(1)    :rem 233
5105 POKE 53285,CO(1)                      :rem 65
```

```
5200 GOTO 120                                          :rem 147
6000 CO(2)=CO(2)+1:IF CO(2)>15 THEN CO(2)=0
                                                       :rem 188
6100 POKE 56269,CO(2):POKE 56270,CO(2)                 :rem 241
6105 POKE 53288,CO(2)                                   :rem 70
6200 GOTO 120                                          :rem 148
7000 CO(3)=CO(3)+1:IF CO(3)>15 THEN CO(3)=0
                                                       :rem 193
7100 POKE 56276,CO(3):POKE 56277,CO(3)                 :rem 249
7105 POKE 53286,CO(3)                                   :rem 70
7200 GOTO 120                                          :rem 149
9000 POKE 53269,2                                       :rem 95
9001 FOR Q1=1 TO 3:FOR Q=1106 TO 1988 STEP 2
                                                       :rem 217
9200 IF PEEK(Q+30000)=Q1 THEN POKE 54272+Q,CO(Q1):
     POKE 54273+Q,CO(Q1)                               :rem 232
9205 NEXT Q                                             :rem 96
9206 NEXT Q1                                           :rem 146
9207 POKE 53269,3                                      :rem 105
9300 GOTO 120                                          :rem 152
```

Character Editor for the Commodore 64

Larry Chiger

Have you ever needed italicized letters or wished for a few characters of Old English script? This character editor lets you create any custom character you need — in as many as four colors per character. A disk drive is required to load and store character definitions from this program.

This program lets you easily create custom characters on the Commodore 64. Any of the 256 characters in either character set can be customized.

To begin, type in the program, save it, then run it. The default character set is copied into memory starting at decimal address 12288, setting the top of memory at that point to allow for instant display of modified characters.

Select f1 to choose the character you want to modify and follow the onscreen prompts. To obtain a character in reverse video, press CTRL-9 (RVS ON), as you would normally, at the prompt for character choice. To turn off reverse video, press CTRL-0 (RVS OFF). Press RETURN to continue.

The next prompt asks if you would like the character displayed in the grid. Displaying is useful if you're only slightly modifying a character. Selecting a new character automatically erases the display grid. This is accomplished by line 1003 of the program. If this line is removed, the display grid will not be erased after choosing f1. This can be useful if you would like to assign the same character image to more than one key. Just draw the image and store it, select a new character with f1 (but do not display it in the grid), and then store that character.

While working on a character in the display grid, press the Commodore and SHIFT keys to draw, but press the Commodore key alone to erase.

Type f2 to display the completed character in actual size. This can be used while making small adjustments on the working character to obtain the best possible results.

This program uses the back-arrow key (←) memory location to store the byte representations of a character while it is being worked on. If you want the back arrow to be itself or some other

character, it must be designed last before you save the entire set to disk.

Type f3 to store the character, as it appears in the display grid, to its proper location in memory. This also records the character for later storage to disk. If the character is represented onscreen, all of those characters onscreen will change to reflect the character's new form.

Type f4 to erase the display grid.

Type f5 to select the multicolor mode, bringing a new menu to the screen. This lets the user select a new background color and choose colors for the character. The colors will be displayed only in the character located under ACTUAL SIZE, and color information is stored to disk when the character set is saved (see program lines 6020-6021 and 6080-6095). Turning multicolor OFF returns the displayed character to its previous form.

Type f6 to save the completed or partially completed character set to disk. The user is prompted to name this new character set, thus enabling many different character sets to be stored on the same disk. If a file by the same name already exists, the user is so informed and given the choice to write over it or return to menu.

Type f7 to load a character set from the disk. If no file exists by the name the user has supplied, the user is given a choice of supplying another name or returning to the menu.

A note about loading a character set. Only those characters that were modified and stored are read into memory. If other characters had been modified previously, they would not be affected. For example, suppose that the characters A, B, and C were modified and saved to disk under the name MODABC. At a later session, the characters C, D, and E were modified, and the character set MODABC was again loaded into memory. The result: Characters A, B, and C would appear as they did in MODABC, while the characters D and E would appear in their newly modified form. All other characters would remain the same.

One advantage of this character generator is that once an old character set is loaded from the disk, it can be modified and re-saved. Characters that were previously modified can be re-modified, and new characters can be added. However, if new characters are created and a character set is then loaded from the disk and saved, only the characters in the most recent loaded set are saved. The newly created characters are lost.

This problem can be avoided in one of two ways. You can first load the character set that you wish to modify, and then modify new characters and resave the set. Alternatively, if you have already modified characters and loaded an old character set from the disk, you can follow these steps for each newly modified character:

1. Select f1.
2. Select a newly modified character and display it in the grid.
3. Select f3.

The f7 key has another use. Once you have developed a new character set, you need a way to load it into memory for other use. Typing f7 runs the character generator program and loads the character set. After completing this process, f8 will end the program with the character set in place.

Sometimes, the placement of the character set as 12288 can be an inconvenience. You might want to modify lines 800-840 (setting up a new character base) and lines 7000-7090 (loading a character set from the disk) for use in your own programs.

Character Editor for the 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
4 PRINTCHR$(142):PRINTCHR$(8):POKE649,1    :rem 100
10 PRINT"{CLR}":PRINT"PLEASE WAIT 1 MINUTE"
                                           :rem 194
20 GOTO800                                  :rem 51
29 PRINT"{CLR}":POKE214,0:PRINT            :rem 242
30 PRINT"{UP}{RVS}CHIGER'S CHARACTER GENERATOR"
                                            :rem 55
40 PRINTTAB(30);"{8 O}"                    :rem 195
50 PRINTTAB(29);"{L}";SPC(8);"{J}"         :rem 152
60 PRINT"{RVS}F1.{OFF} PICK NEW CHAR";TAB(29);"
   {L}";SPC(8);"{J}"                       :rem 144
65 PRINT"{RVS}F2.{OFF} DISPLAY CHAR";TAB(29);"
   {L}";SPC(8);"{J}"                       :rem 155
70 PRINT"{RVS}F3.{OFF} STORE NEW CHAR";TAB(29);"
   {L}";SPC(8);"{J}"                       :rem 249
75 PRINT"{RVS}F4.{OFF} NEW SCREEN";TAB(29);"{L}"
   ;SPC(8);"{J}"                            :rem 20
80 PRINT"{RVS}F5.{OFF} MULTICOLOR ON/OFF";TAB(29);
   "{L}";SPC(8);"{J}"                       :rem 24
85 PRINT"{RVS}F6.{OFF} SAVE CHAR SET TO DISK";TAB(
   29);"{L}";SPC(8);"{J}"                  :rem 116
90 PRINT"{RVS}F7.{OFF} LOAD CHAR SET FROM DISK";TA
   B(29);"{L}";SPC(8);"{J}"                :rem 243
95 PRINT"{RVS}F8.{OFF} QUIT";TAB(30);"{8 Y}"
                                           :rem 207
```

```
110 PRINTTAB(31)"{DOWN}ACTUAL"                    :rem 186
120 PRINTTAB(32)"SIZE"                             :rem 44
130 PRINTTAB(31)"{6 Y}"                            :rem 59
300 REM DRAWING MOVEMENT & FUNCTIONS              :rem 205
305 CP=1134                                        :rem 49
306 IFPEEK(CP)<>160THENT=PEEK(CP)                  :rem 11
307 POKECP,160                                     :rem 31
310 K=PEEK(197):C=PEEK(653)                         :rem 1
320 IFK=7ANDC=0THENPOKECP,T:CP=CP+40:IFCP>1453THEN
    CP=CP-320                                     :rem 190
330 IFK=7ANDC=1THENPOKECP,T:CP=CP-40:IFCP<1102THEN
    CP=CP+320                                     :rem 181
340 IFK=2ANDC=1THENPOKECP,T:CP=CP-1:IFPEEK(CP)=118
    THENCP=CP+8                                   :rem 110
350 IFK=2ANDC=0THENPOKECP,T:CP=CP+1:IFPEEK(CP)=117
    THENCP=CP-8                                   :rem 109

370 IFC=3THENK=2:C=0:T=35:GOTO350                 :rem 237
380 IFC=2ANDK=64THENK=2:C=0:T=32:GOTO350          :rem 175
400 IFK=4ANDC<>1THENPOKECP,T:GOSUB1000            :rem 140
410 IFK=4ANDC=1THENPOKECP,T:GOSUB2000              :rem 81
420 IFK=5ANDC<>1THENPOKECP,T:GOSUB3000            :rem 145
430 IFK=5ANDC=1THENPOKECP,T:GOSUB4000              :rem 86
440 IFK=6ANDC<>1THENPOKECP,T:GOSUB5000            :rem 150
450 IFK=6ANDC=1THENPOKECP,T:GOTO6000               :rem 20
460 IFK=3ANDC<>1THENPOKECP,T:GOSUB7000            :rem 151
465 IFK=3ANDC=1THENPOKECP,T:GOTO8000               :rem 25
480 GOTO306                                       :rem 110
800 REM COPY CHARACTERS TO 12288 ROUTINE          :rem 101
805 POKE52,48:POKE56,48:CLR:REM SET TOP OF MEM TO
    {SPACE}12288                                  :rem 107
810 AD=12288:REM AD IS ADDRESS OF NEW CHAR RAM
                                                   :rem 34
815 POKE56334,PEEK(56334)AND254:REM TURN OFF INTER
    RUPTS                                          :rem 73

820 POKE1,PEEK(1)AND251:REM SWITCH OUT I/O SWITCH
    {SPACE}IN CHAR ROM                             :rem 92
825 FORI=0TO4095:POKEI+AD,PEEK(I+53248):NEXT
                                                  :rem 168
830 POKE1,PEEK(1)OR4:REM SWITCH IN I/O            :rem 241
835 POKE56334,PEEK(56334)OR1:REM TURN ON INTERRUPT
    S                                             :rem 113
840 POKE53272,(PEEK(53272)AND240)OR12:REM CHANGE L
    OC OF CHAR MEMORY                              :rem 95
850 DIMCO$(15):FORX=0TO15:READA$:CO$(X)=A$:NEXT
                                                   :rem 66
860 DATA"BLACK{3 SPACES}","WHITE{3 SPACES}","RED
    {5 SPACES}","CYAN{4 SPACES}","PURPLE{2 SPACES}
    "                                             :rem 120
```

```
863 DATA"GREEN{3 SPACES}","BLUE{4 SPACES}","YELLOW
    {2 SPACES}","ORANGE{2 SPACES}","BROWN
    {3 SPACES}","LT RED{2 SPACES}"          :rem 99
865 DATA"GRAY 1{2 SPACES}","GRAY 2{2 SPACES}","LT
    {SPACE}GREEN","LT BLUE ","GRAY 3{3 SPACES}"
                                            :rem 201
870 DIMSC(511):N%=0:Z$=","                  :rem 118
900 GOTO29                                   :rem 61
1000 REM PICK NEW CHAR                      :rem 212
1003 A=32:GOSUB1230                          :rem 39
1005 POKE214,16:PRINT                       :rem 230
1010 PRINT"IF CHAR DESIRED IS IN LOWER CASE SET,"
                                            :rem 48
1015 PRINT"THEN PRESS LOGO & SHIFT."         :rem 87
1016 PRINT"OTHERWISE PRESS SPACE BAR"        :rem 33
1020 K=PEEK(197):C=PEEK(653)                 :rem 48
1030 IFC=3THENBA=2048:PRINTCHR$(14):GOTO1060
                                            :rem 123
1040 IFK=60THENBA=0:PRINTCHR$(142):GOTO1060:rem 75
1050 GOTO1020                               :rem 194
1060 POKE214,16:PRINT                       :rem 231
1070 PRINT"{38 SPACES}"                     :rem 153
1071 PRINT"{38 SPACES}"                     :rem 154
1072 PRINT"{38 SPACES}"                     :rem 155
1073 POKE214,16:PRINT:PRINT"CHARACTER ?"    :rem 194
1076 GETCH$:POKE1717,160:FORI=0TO150:NEXT:POKE1717
     ,32:IFCH$=""THEN1076                   :rem 162
1078 IFCH$=CHR$(13)THEN1090                 :rem 250
1079 IFCH$="{OFF}"THENRV=0                  :rem 102
1080 IFCH$="{RVS}"THENRV=1                  :rem 223
1085 POKE214,16:PRINT                       :rem 238
1086 IFRV=1THENPRINTTAB(12);"{RVS}"CH$"{OFF}":GOTO
     1088                                    :rem 21
1087 PRINTTAB(12);CH$                       :rem 210
1088 CH=PEEK(1716):GOTO1076                  :rem 31
1090 PRINTCHR$(146):PRINTCHR$(142)          :rem 244
1095 RV=0:POKE214,16:PRINT:PRINT"{23 SPACES}"
                                            :rem 73
1100 FORI=0TO7:POKE12536+I,PEEK(CH*8+12288+BA+I):N
     EXT                                    :rem 113
1110 POKE214,16:PRINT                       :rem 227
1115 INPUT"DISPLAY CHARACTER IN GRID";A$    :rem 156
1120 POKE214,16:PRINT                       :rem 228
1122 PRINT"{37 SPACES}"                     :rem 151
1130 A=CH:IFBA<>0THENA=256+CH               :rem 196
1135 BA=CH*8+BA                              :rem 37
1140 IFLEFT$(A$,1)="N"THENRETURN            :rem 149
1230 B=12288+8*A:POKE214,1:PRINT             :rem 60
1270 FORJ=0TO7:X=PEEK(B+J)/128:PRINTTAB(30);
                                            :rem 149
```

```
1300 FORK=1TO8:X%=X:X=(X-X%)*2:PRINTCHR$(32+X%*3);
                                              :rem 199
1310 NEXT:PRINT:NEXT                          :rem 68
1320 RETURN                                   :rem 166
2000 REM DISPLAY CHARACTER WORKING ON          :rem 7
2010 FORI=0TO7:TE=0:FORJ=0TO-7STEP-1          :rem 243
2015 IFPEEK(1141+I*40+J)<>35THEN2025          :rem 229
2020 TE=TE+2↑ABS(J)                           :rem 95
2025 NEXT:POKE12536+I,TE:NEXT                 :rem 36
2040 POKE55970,1:POKE1698,31:RETURN           :rem 71
3000 REM STORE NEW CHAR                       :rem 60
3010 N%=N%+1                                  :rem 67
3020 FORI=0TO7:TE=0:FORJ=0TO-7STEP-1          :rem 245
3025 IFPEEK(1141+I*40+J)<>35THEN3035          :rem 233
3030 TE=TE+2↑ABS(J)                           :rem 97
3035 NEXT:POKE12288+BA+I,TE:NEXT              :rem 216
3110 FORI=0TON%-1:IFSC(I)=BATHENN%=N%-1:RETURN
                                              :rem 89
3120 NEXT:SC(N%)=BA:RETURN                    :rem 115
4000 REM CLEAR GRID                           :rem 53
4010 A=32:GOSUB1230                           :rem 40
4020 FORI=0TO7:POKE12536+I,0:NEXT             :rem 240
4030 RETURN                                   :rem 167
5000 REM MULTICOLOR MODE ON/OFF               :rem 127
5010 DP=55970                                 :rem 161
5020 POKE214,17:PRINT                         :rem 232
5030 INPUT"MULTICOLOR {RVS}ON{OFF} OR {RVS}OFF
     {OFF}";A$                                :rem 167
5035 IFA$<>"OFF"GOTO5040                      :rem 92
5036 POKE53270,PEEK(53270)AND239              :rem 18
5037 POKEDP,((PEEK(DP)AND240)ORPEEK(55296)AND7)
                                              :rem 255
5038 POKE214,17:PRINT:PRINT"{25 SPACES}":RETURN
                                              :rem 22
5040 POKEDP,PEEK(DP)OR8                       :rem 155
5042 POKE53270,PEEK(53270)OR16                :rem 166
5050 POKE214,17:PRINT                         :rem 235
5055 PRINT"USE{2 SPACES}{RVS}F1{OFF} BACKGROUND CO
     LOR 00"                                  :rem 103
5060 PRINT"USE{2 SPACES}{RVS}F3{OFF} BACKGROUND CO
     LOR 01"                                  :rem 102
5065 PRINT"USE{2 SPACES}{RVS}F5{OFF} BACKGROUND CO
     LOR 10"                                  :rem 109
5070 PRINT"USE{2 SPACES}{RVS}F7{OFF} BACKGROUND CO
     LOR 11"                                  :rem 108
5075 PRINT"USE{2 SPACES}{RVS}F2{OFF} EXIT WITH CUR
     RENT COLORS"                             :rem 22
5095 F1=PEEK(53281)AND15:F3=PEEK(53282)AND15
                                              :rem 220
```

```
5096 F5=PEEK(53283)AND15:F7=PEEK(DP)AND7    :rem 72
5100 POKE214,17:PRINT                       :rem 231
5101 PRINTTAB(28);CO$(F1):POKE53281,F1      :rem 174
5102 PRINTTAB(28);CO$(F3):POKE53282,F3      :rem 180
5103 PRINTTAB(28);CO$(F5):POKE53283,F5      :rem 186
5104 PRINTTAB(28);CO$(F7):POKEDP,F7OR8:GOTO5110
                                            :rem 97
5110 GETA$:IFA$=""GOTO5110                  :rem 187
5120 IFA$="{F1}"THENF1=F1+1:IFF1>15THENF1=0:GOTO51
     00                                     :rem 168
5130 IFA$="{F3}"THENF3=F3+1:IFF3>15THENF3=0:GOTO51
     00                                     :rem 178
5140 IFA$="{F5}"THENF5=F5+1:IFF5>15THENF5=0:GOTO51
     00                                     :rem 188
5150 IFA$="{F7}"THENF7=F7+1:IFF7>7THENF7=0:GOTO510
     0                                      :rem 151
5160 IFA$<>"{F2}"THEN5100                   :rem 252
5162 POKE214,16:PRINT:FORI=0TO5             :rem 157
5163 PRINT"{37 SPACES}":NEXT:RETURN          :rem 51
6000 REM SAVE CHAR SET TO DISK              :rem 177
6010 PRINT"{CLR}":INPUT"ARE YOU SURE";A$    :rem 237
6015 IFLEFT$(A$,1)="N"THEN29                 :rem 38
6020 B0%=PEEK(53281):B1%=PEEK(53282):B2%=PEEK(5328
     3)                                     :rem 41
6021 CC%=PEEK(DP)AND15:MC%=PEEK(53270)      :rem 151
6030 OPEN15,8,15                            :rem 87
6035 INPUT"CHARACTER SET NAME";CN$          :rem 44
6040 OPEN2,8,2,CN$+",S,W"                   :rem 66
6043 GOSUB6500                              :rem 24
6045 IFAA<>63ANDAA<>0THENGOTO6600           :rem 246
6050 IFAA=0GOTO6080                         :rem 80
6055 PRINT"CHAR SET ALREADY EXISTS ON DISK,"
                                            :rem 129
6060 INPUT"WRITE OVER IT";A$                :rem 164
6065 IFLEFT$(A$,1)="N"THENCLOSE2:CLOSE15:GOTO29
                                            :rem 92
6070 CLOSE2:OPEN2,8,2,"@0:"+CN$+",S,W"      :rem 64
6080 PRINT#2,N%;Z$;B0%;Z$;B1%;Z$;B2%;Z$;CC%;Z$;MC%
                                            :rem 59
6085 FORI=0TON%                             :rem 134
6087 D=SC(I)+12288                          :rem 182
6090 PRINT#2,SC(I);Z$PEEK(D)Z$PEEK(D+1)Z$PEEK(D+2)
     Z$PEEK(D+3)Z$PEEK(D+4)                 :rem 214
6091 PRINT#2,PEEK(D+5)Z$PEEK(D+6)Z$PEEK(D+7)
                                            :rem 43
6095 NEXT                                    :rem 19
6100 CLOSE2:CLOSE15                         :rem 133
6110 GOTO29                                 :rem 108
6500 INPUT#15,AA,BB$,CC,DD                  :rem 204
```

```
6510 RETURN                                          :rem 172
6600 PRINTAA;BB$;CC;DD                               :rem 66
6610 CLOSE2:CLOSE15:STOP                             :rem 11
7000 REM READ CHAR SET FROM DISK                     :rem 48
7005 OPEN15,8,15                                     :rem 90
7010 PRINT"{CLR}":INPUT"CHARACTER SET NAME";CN$
                                                     :rem 196
7020 OPEN2,8,2,"@0:"+CN$+",S,R"                      :rem 85
7025 GOSUB6500                                       :rem 25
7030 IFAA<>62ANDAA<>0THENGOTO6600                    :rem 240
7035 IFAA=0GOTO7060                                  :rem 83
7040 PRINT"CHAR SET NOT FOUND":INPUT"ANOTHER";A$
                                                     :rem 210
7045 IFLEFT$(A$,1)="N"THENCLOSE2:CLOSE15:GOTO29
                                                     :rem 91
7050 CLOSE2:GOTO7010                                 :rem 175
7060 INPUT#2,N%,B0%,B1%,B2%,CC%,MC%                  :rem 85
7065 POKE53281,B0%:POKE53282,B1%:POKE53283,B3%:POK
     E55970,CC%                                      :rem 118
7066 POKE53270,MC%                                   :rem 228
7070 FORI=0TON%                                      :rem 129
7075 INPUT#2,SC(I),C0,C1,C2,C3,C4                    :rem 57
7076 INPUT#2,C5,C6,C7                                :rem 168
7077 D=SC(I)+12288                                   :rem 182
7080 POKED,C0:POKED+1,C1:POKED+2,C2:POKED+3,C3:POK
     ED+4,C4                                         :rem 145
7081 POKED+5,C5:POKED+6,C6:POKED+7,C7                :rem 175
7085 NEXT                                            :rem 19
7090 CLOSE2:CLOSE15                                  :rem 142
7095 GOTO29                                          :rem 121
8000 REM QUIT                                        :rem 239
8010 PRINTCHR$(9)                                    :rem 225
8015 POKE649,10                                      :rem 45
8020 END                                             :rem 161
```

Commodore 64 Sound Editor

Daniel L. Riegel

Thanks to its Sound Interface Device (SID), your Commodore 64 can produce many sounds not possible on other home computers. But it takes patience to coax the right sounds from SID. This sound editor will make it easier to explore your 64's audio capabilities.

While most home computers use only frequency and duration settings to produce sound or music, the Commodore 64 adds other parameters to shape, modulate, and filter the resulting output. This sound editor gives you control over those parameters, making it much easier to become familiar with them or to experiment with various combinations.

Though the SID chip has three voices, which can act independently or in combination, a single register (54296) controls the volume of all three voices. It must be set from 1 to 15 for sounds to be heard; the sound editor described here uses the maximum volume setting (15).

Four parameters determine the nature of any sound that you hear: attack, decay, sustain, and release (ADSR). The attack rate specifies the time allowed to reach maximum volume, as determined by the value at 54296. A value of 0 produces a very short attack time, while 15 yields the longest attack time. Explosions or percussion instruments, for instance, would have low attack time values.

The decay rate determines the time allowed for the volume to fall from a maximum value to some lower value that will be sustained. As with attack, a value of 0 is very short, while a value of 15 is long. Similarly, the sustain parameter determines the volume that is maintained until the release phase begins.

The release rate parameter determines how fast the volume falls to 0 from the sustain volume level. A value of 0 produces a very quick release, while a value of 15 produces a release that is very slow.

Each voice has a gate that is used to initiate the attack phase (when the gate is set to 1) and initiate the release phase (when it is set to 0). Duration is the amount of time between setting that gate to 1 and resetting it to 0. This sound editor measures duration in intervals of 1/60 second. A duration of 60 produces a sound lasting one second, a duration of 6 produces a sound lasting 0.1 second,

and so on. The duration must be long enough to allow attack and decay to complete before the voice is reset; otherwise, distortion may occur. For that reason, very short sounds usually require ADSR values of 0, 0, 15, and 0, combined with short durations.

The SID chip can produce eight octaves, designated 0 through 7. The "Sound Editor" dynamically generates and stores tone settings for octave 7, using its highest note (B) as a base. The octave is divided into 12 tones, and each tone's frequency is 2 ↑ (1/12) lower than the next higher tone. For instance, C = C#/2 ↑ (1/12).

The frequency of any tone is half that of the same note in the next higher octave (OCTAVE 6 = OCTAVE 7/2). Therefore, the Sound Editor can generate the scale for any octave N (where N is 0-7) using the formula OCTAVE N = OCTAVE 7/2 ↑ (7 – N). This saves memory by eliminating the need for an array of 96 frequency settings to define eight octaves of 12 tones each.

A sound's waveform determines its harmonic content, or color. SID provides triangle (17), sawtooth (33), pulse (65), and noise (129) waveforms. These waveforms can yield sounds of many different qualities, and the best way to learn about them is to experiment with the Sound Editor.

When the pulse waveform is selected, you will need to specify a value for pulse width (0-4095). For instance, a value of 2048 produces a square wave, which results in a clear, hollow sound.

Other values produce sounds with different feels. For example, a waveform value of 19 synchronizes the frequencies of voices 1 and 3 to produce complex harmonic structures. Value 21 modulates voice 1 with voice 3 to produce ringing sounds (bells, gongs). In such cases the Sound Editor uses note C for voice 3's frequency, picking a frequency that is one octave lower than that specified for voice 1.

The Sound Editor is easy to use. First, enter the various parameters that define the desired sound. Then, referring to the menu, choose one of the four options, using the appropriate function key. The BASIC option lists BASIC code that produces the sound for note C of the octave selected. The CHANGE option allows the parameters to be modified to produce a different sound. The SCALE option plays the 12 tones of the octave specified. The QUIT option terminates the program.

After you have typed in the Sound Editor program, save it to disk or tape and try the combinations given in Table 1.

Table 1. Sample Parameters for 64 Sound Editor

Sound	Attack	Decay	Sustain	Release	Waveform	Pulse Width
Trumpet	6	0	8	0	33	N/A
Violin	10	8	10	9	33	N/A
Xylophone	0	9	0	0	17	N/A
Piano	0	9	0	9	65	1000
Flute	9	10	0	0	17	N/A
Harpsichord	0	9	0	0	33	N/A
Organ	0	0	15	0	17	N/A
Clarinet	8	4	8	0	17	N/A
Chimes	0	11	0	9	19	N/A

Commodore 64 Sound Editor

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
100 REM SOUND EDITOR                                :rem 197
110 PRINT"{CLR}","SOUND EDITOR{3 DOWN}"             :rem 233
115 DIMF(11):F(11)=64814:FORF=10TO0STEP-1:F(F)=INT
    (1/2+F(F+1)/2↑(1/12)):NEXT                      :rem 37
120 SD=54272:V=SD+24:FORI=SDTOV:POKEI,0:NEXT:POKEV
    ,15                                             :rem 111
130 DIMN$(11):N$(0)="C ":N$(1)="C#":N$(2)="D ":N$(
    3)="D#":N$(4)="E ":N$(5)="F "                   :rem 149
140 N$(6)="F#":N$(7)="G ":N$(8)="G#":N$(9)="A ":N$
    (10)="A#":N$(11)="B ":GOTO200                   :rem 246
150 PRINT"{HOME}{2 DOWN}ENTER OPTION [F1] BASIC
    {2 SPACES}[F3] CHANGE"                          :rem 91
152 PRINT"{13 RIGHT}[F5] SCALE{2 SPACES}[F7] QUIT"
                                                    :rem 245
153 GETOP$:IFOP$=""THEN153                          :rem 17
155 IFOP$="{F7}"THENPRINT"{CLR}";:POKEV,0:END
                                                    :rem 240
160 IFOP$="{F3}"THEN200                             :rem 177
165 IFOP$="{F1}"THEN500                             :rem 184
168 IFOP$="{F5}"THEN400                             :rem 188
170 GOTO150                                         :rem 103
200 INPUT"{DOWN}ENTER{2 SPACES}ATTACK VALUE (0-15)
    ";A                                             :rem 186
205 IFA<0ORA>15THENPRINT"{3 UP}":GOTO200            :rem 17
210 INPUT"ENTER{3 SPACES}DECAY VALUE (0-15)";D
                                                    :rem 91
215 IFD<0ORD>15THENPRINT"{2 UP}":GOTO210            :rem 136
220 POKESD+5,A*16+D                                 :rem 39
230 INPUT"ENTER SUSTAIN VALUE (0-15)";S             :rem 45
235 IFS<0ORS>15THENPRINT"{2 UP}":GOTO230            :rem 170
```

```
240 INPUT"ENTER RELEASE VALUE (0-15)";R      :rem 7
245 IFR<0ORR>15THENPRINT"{2 UP}":GOTO240  :rem 170
250 POKESD+6,S*16+R                        :rem 75
260 INPUT"ENTER{2 SPACES}OCTAVE VALUE{2 SPACES}(0-
    7)";OC                                 :rem 219
261 IFOC<0OROC>7THENPRINT"{2 UP}":GOTO260 :rem 251
280 INPUT"ENTER DURATION LOOP{2 SPACES}VALUE";DU
                                           :rem 221
285 IFDU<1THENPRINT"{2 UP}":GOTO280         :rem 99
290 INPUT"ENTER WAVEFORM 17 19 21 33 65 129";W
                                           :rem 136
294 RS=0:H3=0:L3=0:IFW=19ORW=21THENRS=1    :rem 154
295 IFRS=1THENSC=INT(F(0)/2↑(8-OC)):H3=INT(SC/256)
    :L3=SC-H3*256                          :rem 217
296 POKESD+15,H3:POKESD+14,L3              :rem 218
300 IFW=65THEN310                          :rem 228
303 PRINT"{38 SPACES}":GOTO150             :rem 112
310 INPUT"ENTER PULSE WIDTH VALUE (0-4095)";PW
                                           :rem 206
315 IFPW<0ORPW>4095THENPRINT"{2 UP}":GOTO310
                                           :rem 188
320 PH=INT(PW/256):PL=PW-PH*256             :rem 95
330 POKESD+2,PL:POKESD+3,PH:GOTO150        :rem 172
400 FORF=0TO11:SC=INT(F(F)/2↑(7-OC)):X=INT(SC/256)
    :POKESD+1,X:POKESD,SC-256*X            :rem 186
410 TD=TI+DU:POKE53280,F:PRINT"{HOME}{23 DOWN}
    {3 RIGHT}";N$(F):POKESD+4,W            :rem 202
420 IFTI<TDTHEN420                          :rem 91
430 POKESD+4,W-1:NEXT:POKESD+4,0:POKE53280,14:PRIN
    T"{UP}{5 SPACES}":GOTO150              :rem 114
500 PRINT"{HOME}{14 DOWN}10 SD=54272:V=SD+24"
                                           :rem 149
502 PRINT"15 FORI=SDTOV:POKEI,0:NEXT:POKEV,15"
                                           :rem 163
504 SC=INT(F(0)/2↑(7-OC))                  :rem 117
505 H=INT(SC/256):L=SC-256*H                :rem 82
510 PRINT"20 POKESD,";MID$(STR$(L),2);":POKESD+1,"
    ;MID$(STR$(H),2);"{4 SPACES}"          :rem 129
520 PRINT"30 POKESD+5,";MID$(STR$(16*A+D),2);
                                           :rem 239
525 PRINT":POKESD+6,";MID$(STR$(16*S+R),2);"
    {6 SPACES}"                             :rem 48
530 IFW=65THENGOSUB630                     :rem 110
535 IFRS=1THENGOSUB650                     :rem 137
540 PRINT"40 TD=TI+";MID$(STR$(DU),2);":POKESD+4,"
    ;MID$(STR$(W),2);"{9 SPACES}"          :rem 144
545 PRINT"50 IFTI<TDTHEN50{17 SPACES}"     :rem 104
550 PRINT"60 POKESD+4,0{20 SPACES}"         :rem 82
560 PRINT"{26 SPACES}"                     :rem 108
```

```
600 GOTO150                                          :rem 101
630 PRINT"35 POKESD+2,";MID$(STR$(PL),2);  :rem 78
640 PRINT":POKESD+3,";MID$(STR$(PH),2);"
    {10 SPACES}":RETURN                              :rem 124
650 PRINT"35 POKESD+15,";MID$(STR$(H3),2); :rem 99
660 PRINT":POKESD+14,";MID$(STR$(L3),2);"
    {7 SPACES}":RETURN                               :rem 151
```

12-Tone Matrix Generator

Gregg Peele

Despite its relatively limited memory, the unexpanded VIC can handle some rather sophisticated assignments. Here is one from modern music theory: a tone row generator for 12-tone music.

If you are a musician, you may be familiar with 12-tone music. Simply put, it is a decidedly modern form of music based on a nonrepeating sequence of 12 musical tones.

The basic sequence of 12 tones is called a *tone row.* The 12-tone composer takes that sequence, inverts and transposes it, mixes those variations with the original tone row, and ends up with a 12-tone composition.

It sounds complex, and it is. But with this program the VIC can take care of a lot of the bookkeeping, generating the inversions and transpositions of any tone row that you enter. The result is a matrix of tones that can be used as a quick reference for the teacher or composer of 12-tone music. As an added bonus, the program will also sound the tone row to give you an idea of how the piece might sound.

Because this program was written for use on a black-and-white television, no color has been added. In addition, to take best advantage of the VIC's 22-column screen, only sharps and natural tones have been represented. Sharps are represented by reverse letters.

The original tone row can be generated randomly by the program, or you can create and enter your own tone row. In any case, the computer will produce all possible transpositions of the tones and all possible inversions of those transpositions. They will be displayed in matrix form.

This program is of great interest to the serious musician; it may also be useful in a college-level music theory class. But whether musician or not, you're certain to be fascinated by this journey into the realm of modern music — courtesy of your unexpanded but unintimidated VIC.

Twelve-Tone Matrix Generator

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
1 REM"MATRIX"                                    :rem 46
2 PRINT"{CLR}"                                   :rem 150
3 CLR                                            :rem 20
```

```
5 DIMX(14)                                          :rem 29
6 GOTO2000                                          :rem 49
7 PRINT"ENTER THE NOTES FOR YOUR ROW. DO NOT REPEA
  T NOTES."                                         :rem 72
8 PRINT"CHOOSE FROM C,C#,D,D#,E,F,F#,G,G#,A,A# AND
   B.{2 SPACES}ENTER NOTE AND HIT RETURN"           :rem 227
9 FORY=1TO12                                        :rem 237
10 INPUTA$:IFA$="C"THENX(Y)=1                       :rem 231
15 IFA$="C#"THENX(Y)=2                              :rem 225
20 IFA$="D"THENX(Y)=3                               :rem 188
25 IFA$="D#"THENX(Y)=4                              :rem 229
30 IFA$="E"THENX(Y)=5                               :rem 192
35 IFA$="F"THENX(Y)=6                               :rem 199
40 IFA$="F#"THENX(Y)=7                              :rem 231
45 IFA$="G"THENX(Y)=8                               :rem 203
50 IFA$="G#"THENX(Y)=9                              :rem 235
55 IFA$="A"THENX(Y)=10                              :rem 239
60 IFA$="A#"THENX(Y)=11                             :rem 15
65 IFA$="B"THENX(Y)=12                              :rem 243
66 IFX(Y)=0THENPRINT"INVALID ENTRY START OVER":GOT
   O6000                                            :rem 6
80 FORBC=1TO12                                      :rem 72
85 IFBC<YANDX(BC)=X(Y)THENPRINT"INVALID ENTRY STAR
   T OVER":GOTO6000                                 :rem 242
88 NEXTBC                                           :rem 52
99 NEXTY                                            :rem 10
100 PRINT"{CLR}"                                    :rem 245
200 FORG=1TO12                                      :rem 52
1000 IFX(G)=1THENPRINT"C";                          :rem 44
1010 IFX(G)=2THENPRINT"{RVS}C{OFF}";                :rem 210
1020 IFX(G)=3THENPRINT"D";                          :rem 49
1030 IFX(G)=4THENPRINT"{RVS}D{OFF}";                :rem 215
1040 IFX(G)=5THENPRINT"E";                          :rem 54
1050 IFX(G)=6THENPRINT"F";                          :rem 57
1060 IFX(G)=7THENPRINT"{RVS}F{OFF}";                :rem 223
1070 IFX(G)=8THENPRINT"G";                          :rem 62
1080 IFX(G)=9THENPRINT"{RVS}G{OFF}";                :rem 228
1090 IFX(G)=10THENPRINT"A";                         :rem 99
1100 IFX(G)=11THENPRINT"{RVS}A{OFF}";               :rem 0
1110 IFX(G)=12THENPRINT"B";                         :rem 95
1118 NEXTG                                          :rem 81
1119 PRINT" "                                       :rem 157
1120 REM DIFFERENCE ROW                             :rem 107
1150 DIMZ(13)                                       :rem 176
1200 Z(10)=X(11)-X(1)                               :rem 30
1201 Z(1)=-1*(X(2)-X(1))                            :rem 152
1210 Z(2)=-1*(X(3)-X(1))                            :rem 154
1220 Z(3)=-1*(X(4)-X(1))                            :rem 157
1240 Z(4)=-1*(X(5)-X(1))                            :rem 161
```

```
1250 Z(5)=-1*(X(6)-X(1))                    :rem 164
1260 Z(6)=-1*(X(7)-X(1))                    :rem 167
1270 Z(7)=-1*(X(8)-X(1))                    :rem 170
1280 Z(8)=-1*(X(9)-X(1))                    :rem 173
1290 Z(9)=-1*(X(10)-X(1))                   :rem 215
1292 Z(10)=-1*(X(11)-X(1))                    :rem 2
1294 Z(11)=-1*(X(12)-X(1))                    :rem 6
1296 Z(12)=-1*(X(1)-X(12))                    :rem 9
1298 Z(13)=-1*(X(2)-X(1))                   :rem 219
1300 REM INVERTED ROW                         :rem 1
1305 DIMI(13)                               :rem 161
1310 FORJ=1TO13                             :rem 107
1320 LETI(J)=X(1)+Z(J)                      :rem 198
1330 REM IFI(J)>12THENI(J)=I(J)-12          :rem 131
1340 REM IFI(J)<1THENI(J)=I(J)+12            :rem 78
1350 NEXTJ                                   :rem 82
1400 REM POINT DIFFERENCE ROW               :rem 246
1410 DIMD(13)                               :rem 153
1420 FORE=1TO11                             :rem 102
1430 D(1)=X(2)-X(1)                         :rem 173
1440 D(2)=X(3)-X(1)                         :rem 176
1445 D(3)=X(4)-X(1)                         :rem 183
1450 D(4)=X(5)-X(1)                         :rem 181
1460 D(5)=X(6)-X(1)                         :rem 184
1470 D(6)=X(7)-X(1)                         :rem 187
1480 D(7)=X(8)-X(1)                         :rem 190
1490 D(8)=X(9)-X(1)                         :rem 193
1510 D(9)=X(10)-X(1)                        :rem 227
1520 D(10)=X(11)-X(1)                        :rem 13
1530 D(11)=X(12)-X(1)                        :rem 16
1540 NEXTE                                   :rem 78
1600 REM FINAL ROW                           :rem 13
1603 DIMK(12)                               :rem 163
1605 FORF=1TO11                             :rem 108
1610 FOR H=1TO12                            :rem 107
1620 IFH=1THENK(1)=I(F)                      :rem 39
1630 IFH>1THENK(H)=K(1)+D(H-1)              :rem 147
1640 IFK(H)>12THENK(H)=K(H)-12              :rem 163
1650 IFK(H)<1 THENK(H)=K(H)+12              :rem 110
1660 IFK(H)=1THENPRINT"C";                   :rem 44
1661 IFK(H)=2THENPRINT"{RVS}C{OFF}";        :rem 210
1662 IFK(H)=3THENPRINT"D";                   :rem 49
1663 IFK(H)=4THENPRINT"{RVS}D{OFF}";        :rem 215
1664 IFK(H)=5THENPRINT"E";                   :rem 54
1665 IFK(H)=6THENPRINT"F";                   :rem 57
1666 IFK(H)=7THENPRINT"{RVS}F{OFF}";        :rem 223
1667 IFK(H)=8THENPRINT"G";                   :rem 62
1668 IFK(H)=9THENPRINT"{RVS}G{OFF}";        :rem 228
1669 IFK(H)=10THENPRINT"A";                  :rem 99
```

```
1670 IFK(H)=11THENPRINT"{RVS}A{OFF}";          :rem 0
1671 IFK(H)=12THENPRINT"B";                    :rem 95
1675 NEXTH                                     :rem 90
1676 PRINT" "                                  :rem 165
1680 NEXTF                                     :rem 84
1685 PRINT                                     :rem 97
1690 PRINT"ONLY NATURALS AND{5 SPACES}SHARPS ARE R
     EPRESENTED"                               :rem 10
1695 PRINT"SHARPS ARE                          :rem 45
1696 PRINT" {RVS}REVERSE{OFF} FIELD NOTES."
                                               :rem 130
1700 GOTO4990                                  :rem 215
2000 PRINT"{CLR}WOULD YOU LIKE THE{4 SPACES}COMPUT
     ER TO PRODUCE{3 SPACES}A ROW? ENTER Y OR N
     {3 SPACES}";                              :rem 81
2001 PRINT"THEN PRESS RETURN"                  :rem 48
2010 INPUTQ$                                   :rem 200
2020 IFQ$="N"THENPRINT"{CLR}":GOTO7            :rem 212
2025 FORY=1TO12                                :rem 125
2027 IFQ$<>"Y"THEN2000                         :rem 215
2030 X(Y)=INT(12*RND(0))+1                     :rem 142
2040 FORBC=1TO12                               :rem 166
2050 IFBC<YANDX(BC)=X(Y)THENGOTO2030           :rem 221
2060 NEXTBC                                    :rem 140
2070 NEXTY                                     :rem 97
2080 GOTO100                                   :rem 148
4990 PRINT"LISTEN TO THE ORIGINAL ROW"         :rem 71
4999 FORL=1TO13                                :rem 135
5000 V=36878                                   :rem 104
5010 W=36875                                   :rem 103
5020 C=195:CS=199:D=201:DS=203:E=207:F=209:FS=212:
     G=215:GS=217                              :rem 218
5021 A=219:AS=221:B=223:FORY=1TO12:IFY=1THENPOKEV,
     15                                        :rem 150
5030 IFX(L)=1THENPOKEW,C                       :rem 222
5031 IFX(L)=2THENPOKEW,CS                      :rem 51
5032 IFX(L)=3THENPOKEW,D                       :rem 227
5033 IFX(L)=4THENPOKEW,DS                      :rem 56
5034 IFX(L)=5THENPOKEW,E                       :rem 232
5035 IFX(L)=6THENPOKEW,F                       :rem 235
5036 IFX(L)=7THENPOKEW,FS                      :rem 64
5037 IFX(L)=8THENPOKEW,G                       :rem 240
5038 IFX(L)=9THENPOKEW,GS                      :rem 69
5039 IFX(L)=10THENPOKEW,A                      :rem 21
5040 IFX(L)=11THENPOKEW,AS                     :rem 97
5041 IFX(L)=12THENPOKEW,B                      :rem 17
5050 FOREF=1TO900:NEXTEF                       :rem 234
5060 IFL=13THENPOKE36878,0                     :rem 17
5070 NEXTL                                     :rem 87
```

```
5080 PRINT"{CLR}DO YOU WANT ANOTHER{4 SPACES}ROW?
     {SPACE}ENTER Y OR N "                    :rem 9
5082 PRINT"THEN PRESS RETURN"                 :rem 60
5085 INPUTJ$:IFJ$="Y"THENGOTO1                :rem 122
5086 IFJ$<>"N"THENGOTO5080                    :rem 17
5090 END                                      :rem 165
6000 FORN=1TO2000:NEXTN:GOTO1                 :rem 57
```

Chapter 6

Utilities

Quick Delete

W.M. Shockley

This helpful utility lets you automatically delete a block of lines from your programs. It will run equally well on the VIC or 64.

If you have ever had to delete a large number of lines from a program, you know how tedious it can be to remove them one at a time. "Quick Delete" is a six-line program that will let you delete them quickly and easily. Type in and SAVE the program. Then LIST it and LOAD your own program. Move the cursor up to the first line of the Quick Delete routine, and press RETURN to append each line to your program.

Type RUN 60000 and you will be prompted to enter the first and last line numbers of the block you want deleted. Type in those numbers, and the routine will delete them and all lines in between.

Quick Delete uses the dynamic keyboard technique. Lines 60000 and 60010 ask you for the starting and ending lines. Line 60020 prints the number of the line currently being deleted, and line 60030 increments the line number and checks to see if the ending line number has been reached. Line 60040 prints a line that will be executed in direct mode after the program ends. Just before it ends, though, line 60050 simulates pressing the RETURN key twice by POKEing two 13's (the ASCII code for RETURN) into the keyboard buffer and POKEing location 198 with the number of keys pressed. These RETURNs will be executed immediately after the program ends.

The line number to be deleted is printed on the screen on the same row that the cursor will be on when the program ends. When the first RETURN is executed, that line will be deleted just as if you typed the line number and hit RETURN yourself. The second RETURN then enters the other line that was printed. Since all variables are cleared when a line is deleted, this line first restores the variables to the values they had when they were printed on the screen. Then the GOTO 60020 command continues the program to delete the next line.

For short programs, this utility will not be very useful. If you have only a few lines to delete, it is easier to handle them manually. On longer programs, however, it will definitely be of use.

Quick Delete for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
60000 INPUT "{CLR}STARTING LINE";SL          :rem 203
60010 INPUT "{CLR}ENDING LINE";EL             :rem 7
60020 PRINT"{CLR}{3 DOWN}"SL                  :rem 46
60030 SL=SL+1:IF EL=SL -2 THEN END           :rem 107
60040 PRINT"SL="SL":EL="EL":GOTO 60020{HOME}"
                                             :rem 229
60050 POKE631,13:POKE632,13:POKE198,2         :rem 81
```

Formatting Numbers

Larry D. Moody

Have you ever tried to print columns of numbers only to find that they line up under the leftmost character — or that they don't line up at all? You can use the two formatting routines described here to round off numbers, organize them into columns, and make them easier to read. Both will run on the VIC or 64.

People are creatures of habit. They're used to seeing columns of numbers right justified, and they expect to see dollars-and-cents figures with aligned decimal points and two digits to the right of the decimal.

Unfortunately, neither the Commodore 64 nor the VIC-20 has the commands necessary to format columns of numbers in the traditional way. Both computers ignore trailing zeros and would print $500.00 as $500, and neither will automatically break large numbers into groups of three digits separated by commas (#,###,### or #,###,###.00).

You can use these routines to give your 64 or VIC these capabilities. Both routines will accurately handle positive and negative numbers of up to nine significant digits.

The routines use line numbers that do not overlap and that allow both to be used in a single program. In addition, the line numbers are high enough to be added to all but the longest programs without requiring any renumbering.

The following variables are used:

Numeric	Alphanumeric
F1	F$
F2	F0$
F3	F1$
F4	F2$
F5	
F7	

You should reserve these variables for use by the formatting routines. Otherwise, serious errors may result.

How the Routines Work

Integer Format. This routine (Program 1) accepts as input any number in decimal form. It can be either a real number or an integer, and it can be positive or negative.

The program rounds the number to the closest integer — 5's are rounded up; 4's are rounded down. The resulting number is then converted into a string of alphanumeric characters and broken down into groups of three digits separated by commas. The formatted string F$ and its length F3 are available when you return to the main program.

Dollars and Cents. This routine (Program 2) is fundamentally similar to the previous one, except that it returns a string with two digits to the right of the decimal point and a dollar sign ($) as the leftmost character. The formatted string F$ and its length F3 are available when you return to the main program.

Using the Routines

Both listings include extra lines (lines 10-95) to demonstrate how the routines work. Type in and RUN the complete listings to get a feel for what the routines can do, but delete lines 10-95 before including them in your programs. Note that actual screen formatting is done outside the number formatting routines. You can use either or both in your programs, and the result will be number columns that are easier to read and understand.

Program 1. Integer Formatter for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 REM ---LINES 10-95 ILLUSTRATE THE          :rem 45
20 REM{4 SPACES}USE OF THIS SUBROUTINE IN :rem 167
30 REM{4 SPACES}A PROGRAM.                    :rem 206
35 REM{4 SPACES}OUTPUT IS TO MONITOR OR TV.:rem 29
40 FOR I=1 TO 20                               :rem 7
50 PRINT"UNFORMATTED FORMATTED"               :rem 37
55 PRINT"=========== =========="              :rem 255
60 FOR I=1 TO 20                               :rem 9
65 N=(RND(1)-.5)*2*10↑(INT(RND(1)*5))        :rem 212
70 REM--------------------------------        :rem 24
75 F1=N :GOSUB 51410 :N$=F$ :T1=F3            :rem 235
80 PRINT N;TAB(20-T1)N$                        :rem 44
85 REM--------------------------------        :rem 30
90 NEXT I                                     :rem 241
95 END                                         :rem 69
51400 REM ----"INTEGER # FORMAT"----LDM       :rem 97
51401 REM{3 SPACES}FORMATS NUMBERS TO PRINT:rem 71
51402 REM{3 SPACES}RIGHT JUSTIFIED.{2 SPACES}THIS
                                              :rem 107
51404 REM{3 SPACES}SUB-RTN WILL HANDLE NUMBERS
                                              :rem 237
```

```
51406 REM{4 SPACES}<={2 SPACES}999,999,999.9
      {2 SPACES}AND                          :rem 240
51408 REM{4 SPACES}>= -999,999,999.9          :rem 78
51409 REM                                     :rem 231
51410 F1=INT(F1+.5) :REM{3 SPACES}ROUNDOFF #
                                              :rem 148
51415 REM{2 SPACES}DELETE BLANK AT LEFT OF STR$
                                              :rem 113
51420 F0$=STR$(F1)                            :rem 184
51430 F4=LEN(F0$)-1                           :rem 220
51435 F0$=RIGHT$(F0$,F4)                       :rem 12
51440 F2$=""                                   :rem 27
51445 IF F4<4 THEN F2$=F0$ :GOTO 51510        :rem 141
51450 REM{3 SPACES}SEPARATE INTO GROUPS OF:rem 231
51455 REM{3 SPACES}THREE DIGITS, USING COMMAS
                                              :rem 150
51460 F5=INT((F4+2)/3)-1 :REM{2 SPACES}# OF ,'S
                                               :rem 88
51470 FOR F3=1 TO F5                          :rem 237
51480 F2$=","+MID$(F0$,F4+1-3*F3,3)+F2$       :rem 138
51485 NEXT F3                                 :rem 191
51490 F3=F3-1                                 :rem 144
51500 F2$=LEFT$(F0$,F4-3*F3)+F2$              :rem 126
51510 F1$=""                                   :rem 24
51515 REM{3 SPACES}CHECK FOR NEGATIVE VALUE
                                              :rem 250
51520 IF F1<0 THEN F1$="-"                    :rem 231
51525 REM{3 SPACES}ASSEMBLE FINAL STR$        :rem 185
51530 F$=F1$+F2$                               :rem 7
51535 REM{3 SPACES}LENGTH OF COMPLETED STR$
                                              :rem 248
51540 F3=LEN(F$)                               :rem 79
51550 RETURN                                  :rem 224
```

Program 2. Dollars and Cents Formatter for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 REM ---LINES 10-95 ILLUSTRATE THE          :rem 45
20 REM{4 SPACES}USE OF THIS SUBROUTINE IN :rem 167
30 REM{4 SPACES}A PROGRAM.                    :rem 206
35 REM{4 SPACES}OUTPUT IS TO MONITOR OR TV.:rem 29
40 FOR I=1 TO 20                               :rem 7
50 PRINT"UNFORMATTED{2 SPACES}FORMATTED"      :rem 37
55 PRINT"==========={2 SPACES}========="      :rem 255
60 FOR I=1 TO 20                               :rem 9
65 N=(RND(1))*2*10↑(INT(RND(1)*4))            :rem 67
70 REM                                         :rem 75
75 F1=N :GOSUB 51000 :N$=F$ :T1=F3            :rem 230
```

```
80 PRINT N;TAB(21-T1)N$                                  :rem 45
85 REM                                                   :rem 81
90 NEXT I                                               :rem 241
95 END                                                   :rem 69
51000 REM                                               :rem 218
51001 REM{3 SPACES}REFORMATS 'F1' TO RIGHT:rem 116
51002 REM{3 SPACES}JUSTIFIED DOLLARS & CENTS.
                                                        :rem 101
51004 REM{3 SPACES}SUB-RTN WILL HANDLE NUMBERS
                                                        :rem 233
51005 REM{5 SPACES}<={2 SPACES}$9,999,999.99
      {3 SPACES}AND                                     :rem 214
51006 REM{5 SPACES}>= $-9,999,999.99                     :rem 51
51007 REM                                               :rem 225
51008 REM{3 SPACES}ROUND THE AMOUNT TO PENNY.
                                                        :rem 122
51010 F1=INT(F1*100+.5)/100                             :rem 103
51020 F0$="" :F$=""                                      :rem 56
51040 IF F1=0 GOTO 51060                                :rem 162
51050 IF F1<1 AND F1>-1 THEN F0$="0"                    :rem 206
51060 F1$=STR$(F1)                                      :rem 185
51070 REM{2 SPACES}DELETE BLANK AT LEFT OF STR$
                                                        :rem 110
51080 F2=LEN(F1$)-1                                     :rem 220
51090 F1$=MID$(F1$,2,F2)                                :rem 195
51100 REM{3 SPACES}FIND LOC OF DECIMAL POINT.
                                                         :rem 22
51105 FOR F7=1 TO F2                                    :rem 233
51110 F2$=MID$(F1$,F7,1) :F3=F7                          :rem 46
51125 IF F2$="." GOTO 51150                              :rem 13
51130 NEXT F7                                           :rem 182
51132 REM{3 SPACES}IF NO DIGITS RIGHT OF '.'
                                                         :rem 95
51133 REM{3 SPACES}THEN PUT '00' THERE.                  :rem 93
51140 F$=".00" :GOTO 51155                              :rem 232
51145 REM{3 SPACES}IF NO PENNIES DIGIT, THEN
                                                        :rem 238
51146 REM{3 SPACES}PUT A ZERO THERE.                      :rem 5
51150 IF F3=F2-1 THEN F$="0"                             :rem 97
51152 REM{3 SPACES}ASSEMBLE INTERMEDIATE STRING
                                                        :rem 128
51155 F$=F0$+F1$+F$ :F4=LEN(F$)                          :rem 40
51160 F0$=LEFT$(F$,F4-3)                                :rem 228
51165 F1$=RIGHT$(F$,3)                                  :rem 150
51170 F3$="," :F2$=""                                   :rem 159
51175 F4=LEN(F0$)                                       :rem 132
51180 IF F4<4 THEN F2$=F0$ :GOTO 51250                  :rem 138
51190 F5=INT((F4+2)/3)-1 :REM # OF ','S                 :rem 127
51195 REM{3 SPACES}SEPARATE INTO GROUPS OF:rem 237
```

```
51196 REM{3 SPACES}THREE DIGITS WITH COMMAS.
                                                :rem 80
51200 FOR F7=1 TO F5                           :rem 232
51210 F2$=F3$+MID$(F0$,F4+1-3*F7,3)+F2$        :rem 178
51220 NEXT F7                                  :rem 182
51230 F7=F7-1                                  :rem 144
51235 REM{3 SPACES}ADD ON LEFTMOST GROUP OF
                                               :rem 218
51236 REM{3 SPACES}DIGITS.                     :rem 215
51240 F2$=LEFT$(F0$,F4-3*F7)+F2$               :rem 131
51245 REM{3 SPACES}CHECK FOR NEGATIVE VALUE
                                               :rem 250
51250 F3$="$" :IF F1<0 THEN F3$="$-"           :rem 137
51255 REM{3 SPACES}ASSEMBE FINAL STR$          :rem 109
51260 F$=F3$+F2$+F1$                           :rem 207
51270 REM{3 SPACES}LENGTH OF COMPLETED STR$
                                               :rem 244
51280 F3=LEN(F$)                                :rem 80
51290 RETURN                                   :rem 225
```

Numeric Keypad

Ronnie Isbel

A numeric keypad would be a welcome accessory for many Commodore users. Use this short program to redefine keys on your VIC or 64 and customize a keypad of your own.

Here is a simple program that you can use to give your Commodore computer a numeric keypad. Plug-in keypads are available. But experience with data entry operators has shown that separate keypads waste a lot of hand and head motion, particularly when working with alphanumeric data, so I designed a numeric keypad routine that would use existing keys on the Commodore keyboard.

This program uses the keys M, J, K, L, U, I, O, 7, 8, and 9 for the digits 0 through 9. However, by redefining other keys, it is easy to customize your keypad in any way that you desire. You could define 1, 2, and 3 across the top, for example, or you could even define a keypad on the left side of the keyboard. Left-handers, rejoice!

To make labels for your keys, use a dime as a template to draw ten circles on a white adhesive address label. Write the new values for each key inside one of the circles; then cut them out and stick them on the keys. You can remove or rearrange the labels at any time.

This is written as a general-purpose routine to be added to the end of your own programs. When you want to use the keypad, type in GOSUB 50000. Following the GOSUB, equate the subroutine's variable name (VA) to your own variable name. Finally, do not use $, the comma, or the period, or you'll get the message EXTRA IGNORED.

Numeric Keypad for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10 GOSUB50000                                    :rem 214
20 NN=VA                                         :rem 210
30 PRINTNN                                       :rem 140
40 GOTO10                                        :rem 254
50000 REM{2 SPACES}INTEGER NUMERIC KEYPAD        :rem 184
50005 IF VV > 1 GOTO 50020                       :rem 212
50006 VV = 1                                      :rem 21
50010 DIM VO$(16)                                 :rem 81
```

```
50020 FOR VV=0TO16:VO$(VV)=" ":NEXTVV      :rem 167
50025 INPUT VA$: VL = LEN(VA$)              :rem 75
50030 FOR V = 1 TO VL                       :rem 232
50035 VX$ = MID$(VA$,V,1)                   :rem 245
50040 IFVX$="7"ORVX$="'"THEN VO$(V)="7"     :rem 132
50050 IFVX$="8"ORVX$="("THEN VO$(V)="8"     :rem 136
50060 IFVX$="9"ORVX$=")"THEN VO$(V)="9"     :rem 140
50070 IFVX$="U"ORVX$="U"THEN VO$(V)="4"     :rem 80
50080 IFVX$="I"ORVX$="I"THEN VO$(V)="5"     :rem 58
50090 IFVX$="O"ORVX$="O"THEN VO$(V)="6"     :rem 72
50100 IFVX$="J"ORVX$="J"THEN VO$(V)="1"     :rem 49
50110 IFVX$="K"ORVX$="K"THEN VO$(V)="2"     :rem 53
50120 IFVX$="L"ORVX$="L"THEN VO$(V)="3"     :rem 57
50130 IFVX$="M"ORVX$="M"THEN VO$(V)="0"     :rem 57
50140 IFVX$=","ORVX$="<"THEN VO$(V)="."     :rem 134
50150 NEXT V                                :rem 144
50160 FOR V = 0 TO VL                       :rem 235
50170 VO$(16) = VO$(16)+VO$(V)              :rem 215
50180 NEXT V                                :rem 147
50190 VA = VAL(VO$(16))                     :rem 136
50200 REM YOUR INPUT IS COMING BACK TO YOU IN VA.
      {SPACE}USE = TO YOUR NAME.            :rem 139
50210 RETURN                                :rem 216
```

Auto Save/Scratch

Robert Jones

Experienced VIC and 64 programmers do frequent SAVEs when writing or debugging a program. Use this routine to simplify the process and keep track of what you've done.

"Auto Save/Scratch," written for the VIC-20 or Commodore 64, will save you a lot of time. Like many utilities, it was developed to get rid of a bug. I always save every half-hour or so, first to disk 1 and then to disk 2, but after swapping several times I found that I had lost my end-of-program marker. The program was still on both copies of the disk, but I got it back only at the cost of time that I could not afford to lose. The solution turned out to be to use different version numbers for each SAVE, and that was the birth of Auto Save/Scratch.

This routine will let you save any program with a simple RUN 10000. As it saves the program, it also saves a unique version number in front of the name, and since each name starts with a different version number, you can load any version using short wild card commands. For instance, to load the fifth version, you would only have to type in LOAD "05*",8.

If you use this utility, you'll find that you have many versions of your program on the same disk at one time. Thus, if you develop a bug in one version, you can return to an earlier version to get yourself out of trouble.

Auto Save/Scratch remembers what your last version was named. To automatically scratch earlier versions, simply select option 2 from the menu. You will be asked for the first and last versions that you want to scratch (you have to enter only the version numbers, not the whole name), and you don't even have to look to see what versions are on that particular disk. If you tell the computer to scratch a version that is not there, it will ignore the mistake and automatically continue until it has deleted all the versions you told it to scratch. So you can keep track of what is going on, it will also display the version number and name every time it scratches or saves.

The listing includes a number of REM statements, but you can leave them out when you type in the program. There are no jumps to REM lines.

Using this routine is straightforward. When you get ready to

write a program, LOAD Auto Save/Scratch, type the name you want to use in line 10000 (don't use more than 13 characters and spaces), and go ahead with your programming. When you are ready to save a copy of what you've done, just type RUN 10000 and press RETURN. The program is menu-driven, so all you have to do is follow the directions on the screen to save your program as often as you wish.

Auto Save/Scratch for VIC and 64

Refer to "The Automatic Proofreader" (Appendix J) before typing this program in.

```
10000 XN$="PROGRAM{2 SPACES}NAME"                    :rem 117
10010 REM{2 SPACES}NOT OVER 13 CHARACTERS OR SPACE
      S IN NAME                                       :rem 95
10020 PRINT CHR$(147)                                :rem 110
10030 REM{2 SPACES}CHR$(147) CLEARS THE SCREEN
                                                      :rem 33
10040 FOR X=1 TO 10:PRINT:NEXT                       :rem 230
10050 REM{2 SPACES}MOVE DOWN 10 LINES                 :rem 37
10060 PRINT"1{2 SPACES}SAVE THIS PROGRAM":PRINT
                                                      :rem 63
10070 PRINT"2{2 SPACES}SCRATCH OLD VERSION":PRINT
                                                     :rem 207
10080 PRINT"3{2 SPACES}EXIT":PRINT                   :rem 254
10090 FOR X=1 TO 10:GET XX$:NEXT                      :rem 18
10100 REM{2 SPACES}EMPTY BUFFER                       :rem 31
10110 X$="":GET X$:IF X$="" GOTO 10110               :rem 120
10120 X=VAL(X$):IF X<1 OR X>3 GOTO 10090             :rem 100
10130 ON X GOTO 10160,10480,10140                    :rem 102
10140 PRINT CHR$(147):END                            :rem 130
10150 REM **********{2 SPACES}SAVE ROUTINE
      {2 SPACES}**********                           :rem 120
10160 OPEN15,8,15,"I0"                               :rem 111
10170 REM{2 SPACES}INITIALIZE THE DISK               :rem 219
10180 XZ$="0:"+XN$+" #,S,R":OPEN2,8,2,XZ$            :rem 161
10190 REM{2 SPACES}XZ$ IS THE NAME OF A FILE THAT
      {SPACE}HOLDS THE LAST VERSION NUMBER:rem 165
10200 INPUT#2,X:CLOSE2:CLOSE15                        :rem 84
10210 REM{2 SPACES}GET THE LAST VERSION NUMBER AND
       CLOSE THE FILE                                  :rem 6
10220 XZ$="@0:"+XN$+" #,S,W":OPEN2,8,2,XZ$:rem 225
10230 REM{2 SPACES}OPENS A CHANNEL TO SAVE THE NEW
       VERSION NUMBER                                 :rem 37
10240 PRINT#2,X+1:CLOSE2:X$=STR$(X):IF X<10 THEN X
      $="0"+RIGHT$(X$,1)                             :rem 101
10250 REM{2 SPACES}SAVE THE NEW VERSION NUMBER AND
       SET X$ TO VERSION NUMBER TO BE SAVED:rem 47
10260 X$=RIGHT$(X$,2):X$="@0:"+X$+" "+XN$ :rem 107
```

```
10270 REM{2 SPACES}X$ IS THE NAME OF THE PROGRAM V
      ERSION YOU ARE ABOUT TO SAVE          :rem 206
10280 PRINT"SAVING":PRINT RIGHT$(X$,(LEN(X$)-3))
                                            :rem 83
10290 SAVE X$,8                             :rem 11
10300 REM{2 SPACES}SAVE THE PROGRAM VERSION:rem 38
10310 VERIFY X$,8                           :rem 170
10320 REM{2 SPACES}IT IS NOT NECESSARY TO VERIFY T
      HE SAVE BUT IT MAKES ME FEEL BETTER :rem 166
10330 CLOSE 15                              :rem 211
10340 OPEN15,8,15,"V0"                      :rem 124
10350 REM{2 SPACES}***********{2 SPACES}CAUTION
      {2 SPACES}***********                 :rem 224
10360 REM{2 SPACES}LINE # 10340 VALIDATES THE DISK
                                            :rem 202
10370 REM THE VIC-1541 DISK DRIVE USER'S MANUAL ST
      ATES THAT YOU                         :rem 184
10380 REM{2 SPACES}SHOULD NEVER VALIDATE A DISK TH
      AT HOLDS RANDOM FILES                 :rem 196
10390 REM{2 SPACES}IF THE DISK HOLDS RANDOM FILES
      {SPACE}DELETE LINE # 10340 - 10430    :rem 69
10430 CLOSE15                               :rem 212
10440 PRINT:PRINT"DONE -- PRESS ANY KEY"    :rem 111
10450 X$="":GET X$:IF X$="" GOTO 10450      :rem 134
10460 GOTO 10000                            :rem 37
10470 REM{3 SPACES}****** SCRATCH OLD VERSIONS
                                            :rem 60
10480 PRINT CHR$(147)                       :rem 120
10490 REM{2 SPACES}CHR$(147) CLEARS THE SCREEN
                                            :rem 43
10500 FOR X=1 TO 10:PRINT:NEXT              :rem 231
10510 REM{2 SPACES}MOVE DOWN 10 LINES       :rem 38
10520 PRINT"ENTER VERSION NUMBER":PRINT     :rem 253
10530 PRINT"FIRST ONE TO SCRATCH":INPUT XF:PRINT
                                            :rem 14
10540 PRINT"LAST ONE TO SCRATCH":INPUT XL:PRINT
                                            :rem 193
10550 OPEN15,8,15,"I0"                      :rem 114
10560 REM{2 SPACES}INITIALIZE THE DISK      :rem 222
10570 PRINT"SCRATCHED"                      :rem 95
10580 FOR X=XF TO XL:X$=STR$(X):IF X<10 THEN X$="0
      "+RIGHT$(X$,1)                        :rem 41
10590 REM{2 SPACES}SET UP LOOP TO SCRATCH THEM AND
       SET X$ TO THE VERSION NUMBER         :rem 53
10600 X$=RIGHT$(X$,2):X$="S0:"+X$+" "+XN$   :rem 124
10610 REM{2 SPACES}X$ IS THE NAME OF THE PROGRAM V
      ERSION YOU ARE ABOUT TO SCRATCH       :rem 165
10620 CLOSE15:OPEN15,8,15,X$                :rem 69
10630 REM{2 SPACES}SCRATCH IT               :rem 131
```

```
10640 CLOSE15                                          :rem 215
10650 PRINT RIGHT$(X$,(LEN(X$)-3))                     :rem 129
10660 REM{2 SPACES}PRINT NAME OF ONE SCRATCHED
                                                       :rem 151
10670 NEXT                                              :rem 61
10680 GOTO 10330:END                                    :rem 64
```

Appendices

A Beginner's Guide to Typing In Programs

What Is a Program?

A computer cannot perform any task by itself. Like a car without gas, a computer has *potential,* but without a program, it isn't going anywhere. Most of the programs published in COMPUTE! Books are written in a computer language called BASIC. BASIC is easy to learn and is built into all VIC-20s and Commodore 64s.

BASIC Programs

This book includes programs for both the VIC and 64. To start out, type in only programs written for your machine, e.g., "VIC Version" if you have a VIC-20.

Computers can be picky. Unlike the English language, which is full of ambiguities, BASIC usually has only one "right way" of stating something. Every letter, character, or number is significant. A common mistake is substituting a letter such as O for the numeral 0, a lowercase l for the numeral 1, or an uppercase B for the numeral 8. Also, you must enter all punctuation such as colons and commas just as they appear in the listing. Spacing can be important. To be safe, type in the listings *exactly* as they appear.

Braces and Special Characters

The exception to this typing rule is when you see the braces, such as {DOWN}. Anything within a set of braces is a special character or characters that cannot easily be listed on a printer. When you come across such a special statement, refer to "How to Type In Programs" (Appendix B).

About DATA Statements

Some programs contain a section or sections of DATA statements. These lines provide information needed by the program. Some DATA statements contain actual programs (called machine language); others contain graphics codes. These lines are especially sensitive to errors.

If a single number in any one DATA statement is mistyped, your machine could lock up, or crash. The keyboard and STOP key may seem dead, and the screen may go blank. Don't panic — no damage is done. To regain control, you have to turn off your computer, 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. *The error is still in the DATA statements, though.*

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 at least need to know how to backspace. Do you know how to enter reverse video, lowercase, and control characters? It's all explained in your computer's manuals.

A Quick Review

1. Type in the program a line at a time, in order. Press RETURN at the end of each line. Use backspace or the back arrow to correct mistakes.

2. Check the line you've typed against the line in the magazine. You can check the entire program again if you get an error when you run the program.

3. Make sure you've entered statements in braces as the appropriate control key (see "How to Type In Programs").

How to Type In Programs

Many of the programs in this book contain special control characters (cursor control, color keys, reverse video, etc.). To make it easy to know exactly what to type when entering one of these programs into your computer, we have established the following listing conventions.

Generally, any VIC-20 or Commodore 64 program listings will contain words within braces which spell out any special characters: {DOWN} would mean to press the cursor down key. {5 SPACES} would mean to press the space bar five times.

To indicate that a key should be *shifted* (hold down the SHIFT key while pressing the other key), the key would be underlined in our listings. For example, S would mean to type the S key while holding the SHIFT key. This would appear on your screen as a "heart" symbol. If you find an underlined key enclosed in braces (e.g., {10 N}), you should type the key as many times as indicated (in our example, you would enter ten shifted N's).

If a key is enclosed in special brackets, [<>], you should hold down the *Commodore key* while pressing the key inside the special brackets. (The Commodore key is the key in the lower left corner of the keyboard.) Again, if the key is preceded by a number, you should press the key as many times as necessary.

Rarely, you'll see a solitary letter of the alphabet enclosed in braces. These characters can be entered on the Commodore 64 by holding down the CTRL key while typing the letter in the braces. For example, {A} would indicate that you should press CTRL-A. You should never have to enter such a character on the VIC-20.

About the *quote mode:* You know that you can move the cursor around the screen with the CRSR keys. Sometimes a programmer will want to move the cursor under program control. That's why you see all the {LEFT}'s, {HOME}'s, and {BLU}'s in our programs. The only way the computer can tell the difference between direct and programmed cursor control is the quote mode.

Once you press the quote (the double quote, SHIFT-2), you are in the quote mode. If you type something and then try to change it by moving the cursor left, you'll only get a bunch of reverse-video lines. These are the symbols for cursor left. The only editing key that isn't programmable is the DEL key; you can

still use DEL to back up and edit the line. Once you type another quote, you are out of quote mode.

You also go into quote mode when you INSerT spaces into a line. In any case, the easiest way to get out of quote mode is to just press RETURN. You'll then be out of quote mode and you can cursor up to the mistyped line and fix it.

Use the following table when entering cursor and color control keys:

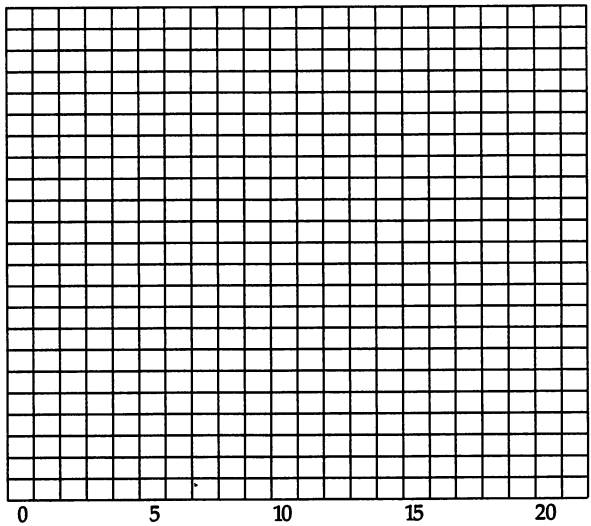
When You Read:	Press:	See:
{CLR}	SHIFT + CLR/HOME	
{HOME}	CLR/HOME	
{UP}	SHIFT + CRSR ↑	
{DOWN}	CRSR ↑	
{LEFT}	SHIFT + CRSR ←	
{RIGHT}	CRSR ←	
{RVS}	CTRL + 9	
{OFF}	CTRL + 0	
{BLK}	CTRL + 1	
{WHT}	CTRL + 2	
{RED}	CTRL + 3	
{CYN}	CTRL + 4	
{PUR}	CTRL + 5	
{GRN}	CTRL + 6	
{BLU}	CTRL + 7	
{YEL}	CTRL + 8	

When You Read:	Press:	See:
{1}	COMMODORE + 1	
{2}	COMMODORE + 2	
{3}	COMMODORE + 3	
{4}	COMMODORE + 4	
{5}	COMMODORE + 5	
{6}	COMMODORE + 6	
{7}	COMMODORE + 7	
{8}	COMMODORE + 8	
{F1}	f1	
{F2}	SHIFT + f1	
{F3}	f3	
{F4}	SHIFT + f3	
{F5}	f5	
{F6}	SHIFT + f5	
{F7}	f7	
{F8}	SHIFT + f7	
←	←	
↑	SHIFT + ↑	

Screen Location Table (VIC)

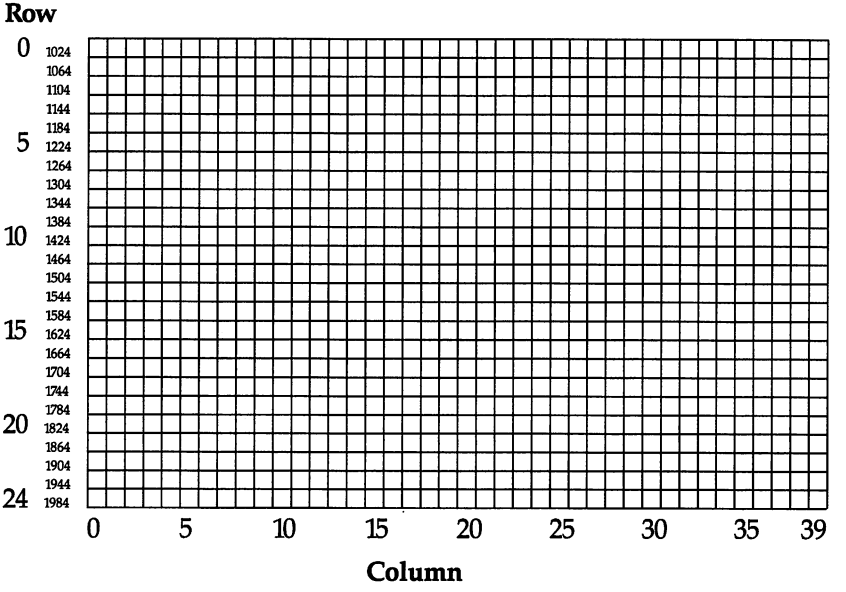
Row

0	7680 (4096)
	7702 (4118)
	7724 (4140)
	7746 (4162)
	7768 (4184)
5	7790 (4206)
	7812 (4228)
	7834 (4250)
	7856 (4272)
	7878 (4294)
10	7900 (4316)
	7922 (4338)
	7944 (4360)
	7966 (4382)
	7988 (4404)
15	8010 (4426)
	8032 (4448)
	8054 (4470)
	8076 (4492)
	8098 (4514)
20	8120 (4536)
	8142 (4558)
22	8164 (4580)



Note: Numbers in parentheses are for VICs with 8K or more of memory expansion.

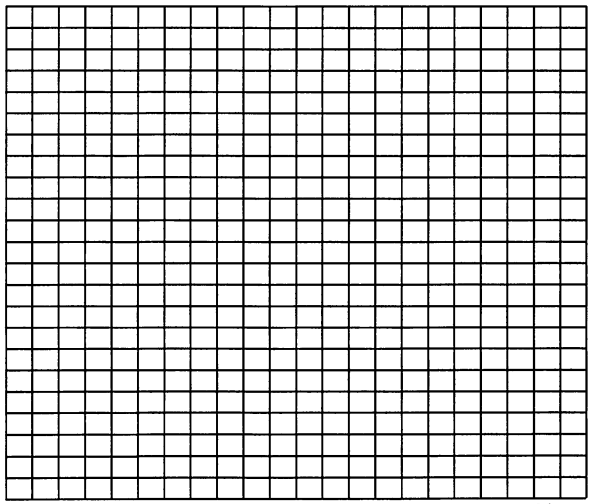
Screen Location Table (64)



Screen Color Memory Table (VIC)

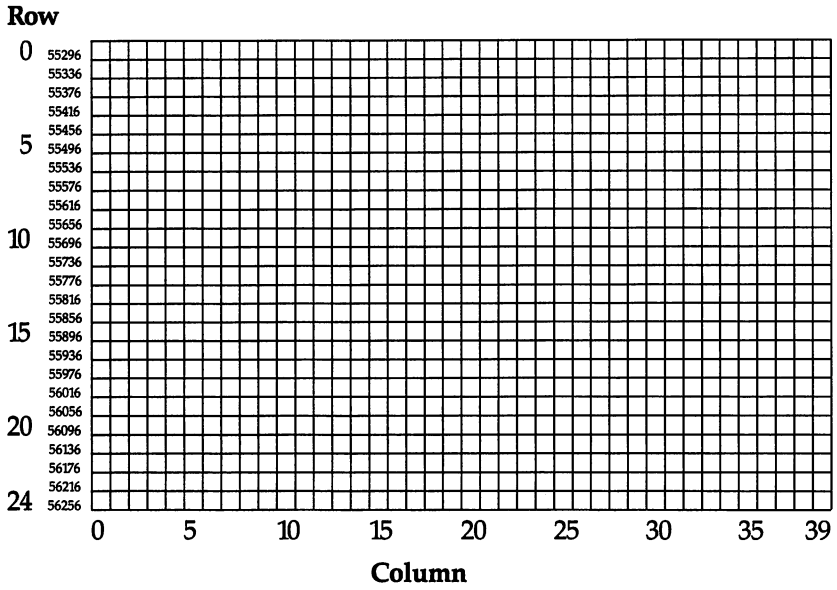
Row

0	38400 (37888)
	38422 (37910)
	38444 (37932)
	38466 (37954)
	38488 (37976)
5	38510 (37998)
	38532 (38020)
	38554 (38042)
	38576 (38064)
	38598 (38086)
10	38620 (38108)
	38642 (38130)
	38664 (38152)
	38686 (38174)
	38708 (38196)
15	38730 (38218)
	38752 (38240)
	38774 (38262)
	38796 (38284)
	38818 (38306)
20	38840 (38328)
	38862 (38350)
22	38884 (38372)



Note: Numbers in parentheses are for VICs with 8K or more of memory expansion.

Screen Color Memory Table (64)



Screen Color Codes

Color:	Black	White	Red	Cyan	Purple	Green	Blue	Yellow
Code:	0	1	2	3	4	5	6	7

Additional Color Codes for 64

Color:	Orange	Brown	Light Red	Dark Gray	Medium Gray	Light Green	Light Blue	Light Gray
Code:	8	9	10	11	12	13	14	15

Screen and Border Colors (VIC Only)

	Border							
Screen	Black	White	Red	Cyan	Purple	Green	Blue	Yellow
Black	8	9	10	11	12	13	14	15
White	24	25	26	27	28	29	30	31
Red	40	41	42	43	44	45	46	47
Cyan	56	57	58	59	60	61	62	63
Purple	72	73	74	75	76	77	78	79
Green	88	89	90	91	92	93	94	95
Blue	104	105	106	107	108	109	110	111
Yellow	120	121	122	123	124	125	126	127
Orange	136	137	138	139	140	141	142	143
Light Orange	152	153	154	155	156	157	158	159
Pink	168	169	170	171	172	173	174	175
Light Cyan	184	185	186	187	188	189	190	191
Light Purple	200	201	202	203	204	205	206	207
Light Green	216	217	218	219	220	221	222	223
Light Blue	232	233	234	235	236	237	238	239
Light Yellow	248	249	250	251	252	253	254	255

To set screen and border colors, select the desired combination from the table above and POKE the corresponding value into location 36879.

ASCII Codes

ASCII	Character	ASCII	Character
5	WHITE	50	2
8	DISABLE SHIFT-COMMODORE	51	3
		52	4
9	ENABLE SHIFT-COMMODORE	53	5
		54	6
13	RETURN	55	7
14	LOWERCASE	56	8
17	CURSOR DOWN	57	9
18	REVERSE VIDEO ON	58	:
19	HOME	59	;
20	DELETE	60	<
28	RED	61	=
29	CURSOR RIGHT	62	>
30	GREEN	63	?
31	BLUE	64	@
32	SPACE	65	A
33	!	66	B
34	"	67	C
35	#	68	D
36	$	69	E
37	%	70	F
38	&	71	G
39	'	72	H
40	(	73	I
41	)	74	J
42	*	75	K
43	+	76	L
44	,	77	M
45	-	78	N
46	.	79	O
47	/	80	P
48	0	81	Q
49	1	82	R

ASCII	Character	ASCII	Character
83	S	120	
84	T	121	
85	U	122	
86	V	123	
87	W	124	
88	X	125	
89	Y	126	
90	Z	127	
91	[	129	ORANGE
92	£	133	f1
93	]	134	f3
94	↑	135	f5
95	←	136	f7
96		137	f2
97		138	f4
98		139	f6
99		140	f8
100		141	SHIFT-RETURN
101		142	UPPERCASE
102		144	BLACK
103		145	CURSOR UP
104		146	REVERSE VIDEO OFF
105		147	CLEAR SCREEN
106		148	INSERT
107		149	BROWN
108		150	LIGHT RED
109		151	GRAY 1
110		152	GRAY 2
111		153	LIGHT GREEN
112		154	LIGHT BLUE
113		155	GRAY 3
114		156	PURPLE
115		157	CURSOR LEFT
116		158	YELLOW
117		159	CYAN
118		160	SHIFT-SPACE
119		161	

ASCII	Character	ASCII	Character
162		200	
163		201	
164		202	
165		203	
166		204	
167		205	
168		206	
169		207	
170		208	
171		209	
172		210	
173		211	
174		212	
175		213	
176		214	
177		215	
178		216	
179		217	
180		218	
181		219	
182		220	
183		221	
184		222	
185		223	
186		224	SPACE
187		225	
188		226	
189		227	
190		228	
191		229	
192		230	
193		231	
194		232	
195		233	
196		234	
197		235	
198		236	
199		237	

ASCII	Character
238	
239	
240	
241	
242	
243	
244	
245	
246	
247	
248	
249	
250	
251	
252	
253	
254	
255	

Notes:

1. 0–4, 6–7, 10–12, 15–16, 21–27, 128–132, 143, and 149–155 have no effect.
2. 192–223 same as 96–127; 224–254 same as 160–190; 255 same as 126.

Screen Codes

POKE	Uppercase and Full Graphics Set	Lower- and Uppercase
0	@	@
1	A	a
2	B	b
3	C	c
4	D	d
5	E	e
6	F	f
7	G	g
8	H	h
9	I	i
10	J	j
11	K	k
12	L	l
13	M	m
14	N	n
15	O	o
16	P	p
17	Q	q
18	R	r
19	S	s
20	T	t
21	U	u
22	V	v
23	W	w
24	X	x
25	Y	y
26	Z	z
27	[	[
28	£	£
29	]	]
30	↑	↑
31	←	←
32	-space-	
33	!	!
34	"	"
35	#	#
36	$	$
37	%	%
38	&	&
39	'	'
40	(	(
41	)	)
42	*	*
43	+	+
44	,	,
45	-	-
46	.	.
47	/	/
48	0	0
49	1	1
50	2	2
51	3	3
52	4	4
53	5	5
54	6	6
55	7	7
56	8	8
57	9	9
58	:	:
59	;	;
60	<	<
61	=	=

POKE	Uppercase and Full Graphics Set	Lower- and Uppercase
62	>	>
63	?	?
64		
65		A
66		B
67		C
68		D
69		E
70		F
71		G
72		H
73		I
74		J
75		K
76		L
77		M
78		N
79		O
80		P
81		Q
82		R
83		S
84		T
85		U
86		V
87		W
88		X
89		Y
90		Z
91		
92		
93		
94		
95		
96	-space-	
97		
98		

POKE	Uppercase and Full Graphics Set	Lower- and Uppercase
99		
100		
101		
102		
103		
104		
105		
106		
107		
108		
109		
110		
111		
112		
113		
114		
115		
116		
117		
118		
119		
120		
121		
122		
123		
124		
125		
126		
127		

128–255 reverse video of 0–127

VIC Keycodes

Key	Keycode	Key	Keycode
A	17	6	58
B	35	7	3
C	34	8	59
D	18	9	4
E	49	0	60
F	42	+	5
G	19	–	61
H	43	£	6
I	12	CLR/HOME	62
J	20	INST/DEL	7
K	44	←	8
L	21	@	53
M	36	*	14
N	28	↑	54
O	52	:	45
P	13	;	22
Q	48	=	46
R	10	RETURN	15
S	41	,	29
T	50	.	37
U	51	/	30
V	27	CRSR ↑↓	31
W	9	CRSR ⇄	23
X	26	f1	39
Y	11	f3	47
Z	33	f5	55
1	0	f7	63
2	56	SPACE	32
3	1	RUN/STOP	24
4	57	NO KEY	
5	2	PRESSED	64

The keycode is the number found at location 197 for the current key being pressed. Try this one-line program:

```
10 PRINT PEEK (197):GOTO 10
```

Values Stored at Location 653

Code	Key(s) pressed
0	(No key pressed)
1	SHIFT
2	Commodore
3	SHIFT and Commodore
4	CTRL
5	SHIFT and CTRL
6	Commodore and CTRL
7	SHIFT, Commodore, and CTRL

Commodore 64 Keycodes

Key	Keycode	Key	Keycode
A	10	6	19
B	28	7	24
C	20	8	27
D	18	9	32
E	14	0	35
F	21	+	40
G	26	−	43
H	29		48
I	33	CLR/HOME	51
J	34	INST/DEL	0
K	37	←	57
L	42	@	46
M	36	*	49
N	39		54
O	38	:	45
P	41	;	50
Q	62	=	53
R	17	RETURN	1
S	13	,	47
T	22	.	44
U	30		55
V	31	CRSR ↑↓	7
W	9	CRSR ⇄	2
X	23	f1	4
Y	25	f3	5
Z	12	f5	6
1	56	f7	3
2	59	SPACE	60
3	8	RUN/STOP	63
4	11	NO KEY	
5	16	PRESSED	64

The keycode is the number found at location 197 for the current key being pressed. Try this one-line program:

```
10 PRINT PEEK (197):GOTO 10
```

Values Stored at Location 653

Code	Key(s) pressed
0	(No key pressed)
1	SHIFT
2	Commodore
3	SHIFT and Commodore
4	CTRL
5	SHIFT and CTRL
6	Commodore and CTRL
7	SHIFT, Commodore, and CTRL

The Automatic Proofreader

"The Automatic Proofreader" will help you type in program listings without typing mistakes. It is a short error-checking program that hides itself in memory. When activated, it lets you know immediately after typing a line from a program listing if you have made a mistake. Please read these instructions carefully before typing any programs.

Preparing the Proofreader

1. Using the listing below, type in the Proofreader. The same program works on both the VIC-20 and Commodore 64. Be very careful when entering the DATA statements — don't type an l instead of a 1, an O instead of a 0, extra commas, etc.

2. Save the Proofreader on tape or disk at least twice *before running it for the first time.* This is very important because the Proofreader erases this part of itself when you first type RUN.

3. After the Proofreader is saved, type RUN. It will check itself for typing errors in the DATA statements and warn you if there's a mistake. Correct any errors and save the corrected version. Keep a copy in a safe place — you'll need it again and again, every time you enter a program from this book.

4. When a correct version of the Proofreader is run, it activates itself. You are now ready to enter a program listing. If you press RUN/STOP-RESTORE, the Proofreader is disabled. To reactivate it, just type the command SYS 886 and press RETURN.

Using the Proofreader

All VIC and 64 listings in this book have a *checksum number* appended to the end of each line, for example, :rem 123. *Don't enter this statement when typing in a program.* It is just for your information. The rem makes the number harmless if someone does type it in. It will, however, use up memory if you enter it, and it will confuse the Proofreader, even if you entered the rest of the line correctly.

When you type in a line from a program listing and press RETURN, the Proofreader displays a number at the top of your screen. *This checksum number must match the checksum number in the printed listing.* If it doesn't, it means you typed the line differently

from the way it is listed. Immediately recheck your typing. Remember, don't type the rem statement with the checksum number; it is published only so you can check it against the number which appears on your screen.

The Proofreader is not picky with spaces. It will not notice extra spaces or missing ones. This is for your convenience, since spacing is generally not important. But occasionally proper spacing *is* important, so be extra careful with spaces, since the Proofreader will catch practically everything else that can go wrong.

There's another thing to watch out for: If you enter the line by using abbreviations for commands, the checksum will not match up. But there is a way to make the Proofreader check it. After entering the line, LIST it. This eliminates the abbreviations. Then move the cursor up to the line and press RETURN. It should now match the checksum. You can check whole groups of lines this way.

Special Tape SAVE Instructions

When you're done typing a listing, you must disable the Proofreader before saving the program on tape. Disable the Proofreader by pressing RUN/STOP-RESTORE (hold down the RUN/STOP key and sharply hit the RESTORE key). This procedure is not necessary for disk SAVEs, *but you must disable the Proofreader this way before a tape SAVE.*

SAVE to tape erases the Proofreader from memory, so you'll have to load and run it again if you want to type another listing. SAVE to disk does not erase the Proofreader.

Hidden Perils

The Proofreader's home in the VIC and 64 is not a very safe haven. Since the cassette buffer is wiped out during tape operations, you need to disable the Proofreader with RUN/STOP-RESTORE before you save your program. This applies only to tape use. Disk users have nothing to worry about.

Not so for VIC and 64 owners with tape drives. What if you type in a program in several sittings? The next day, you come to your computer, load and run the Proofreader, then try to load the partially completed program so you can add to it. But since the Proofreader is trying to hide in the cassette buffer, it is wiped out!

What you need is a way to load the Proofreader after you've loaded the partial program. The problem is, a tape load to the buffer destroys what it's supposed to load.

After you type in and run the Proofreader, enter the following lines in direct mode (without line numbers) *exactly* as shown:

```
A$="PROOFREADER.T": B$="{10 SPACES}": FOR X = 1 TO
   4: A$=A$+B$: NEXTX
FOR X = 886 TO 1018: A$=A$+CHR$(PEEK(X)): NEXTX
OPEN 1,1,1,A$:CLOSE1
```

After you enter the last line, you will be asked to PRESS RECORD & PLAY on your cassette recorder. Put this program at the beginning of a new tape. This gives you a new way to load the Proofreader. Anytime you want to bring the Proofreader into memory without disturbing anything else, put the cassette in the tape drive, rewind, and enter:

OPEN1:CLOSE1

You can now start the Proofreader by typing SYS 886. To test this, PRINT PEEK(886) should return the number 173. If it does not, repeat the steps above, making sure that A$ ("PROOFREADER.T") contains 13 characters and that B$ contains 10 spaces.

You can now reload the Proofreader into memory whenever LOAD or SAVE destroys it, restoring your personal typing helper.

Automatic Proofreader for VIC and 64

```
100 PRINT"{CLR}PLEASE WAIT...":FORI=886TO1018:READ
    A:CK=CK+A:POKEI,A:NEXT
110 IF CK<>17539 THEN PRINT"{DOWN}YOU MADE AN ERRO
    R":PRINT"IN DATA STATEMENTS.":END
120 SYS886:PRINT"{CLR}{2 DOWN}PROOFREADER ACTIVATE
    D.":NEW
886 DATA 173,036,003,201,150,208
892 DATA 001,096,141,151,003,173
898 DATA 037,003,141,152,003,169
904 DATA 150,141,036,003,169,003
910 DATA 141,037,003,169,000,133
916 DATA 254,096,032,087,241,133
922 DATA 251,134,252,132,253,008
928 DATA 201,013,240,017,201,032
934 DATA 240,005,024,101,254,133
940 DATA 254,165,251,166,252,164
946 DATA 253,040,096,169,013,032
952 DATA 210,255,165,214,141,251
958 DATA 003,206,251,003,169,000
964 DATA 133,216,169,019,032,210
970 DATA 255,169,018,032,210,255
```

```
976 DATA 169,058,032,210,255,166
982 DATA 254,169,000,133,254,172
988 DATA 151,003,192,087,208,006
994 DATA 032,205,189,076,235,003
1000 DATA 032,205,221,169,032,032
1006 DATA 210,255,032,210,255,173
1012 DATA 251,003,133,214,076,173
1018 DATA 003
```

Index

Notes

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!** Magazine. Use this form to order your subscription to **COMPUTE!**.

For Fastest Service,
Call Our **Toll-Free** US Order Line

800-334-0868

In NC call 919-275-9809

COMPUTE!
P.O. Box 5406
Greensboro, NC 27403

My Computer Is:
☐ Commodore 64 ☐ TI-99/4A ☐ Timex/Sinclair ☐ VIC-20 ☐ PET
☐ Radio Shack Color Computer ☐ Apple ☐ Atari ☐ Other _____
☐ Don't yet have one...

☐ $24 One Year US Subscription
☐ $45 Two Year US Subscription
☐ $65 Three Year US Subscription

Subscription rates outside the US:

☐ $30 Canada
☐ $42 Europe, Australia, New Zealand/Air Delivery
☐ $52 Middle East, North Africa, Central America/Air Mail
☐ $72 Elsewhere/Air Mail
☐ $30 International Surface Mail (lengthy, unreliable delivery)

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Payment must be in US Funds drawn on a US Bank; International Money Order, or charge card.

☐ Payment Enclosed ☐ VISA
☐ MasterCard ☐ American Express
Acct. No. _____ Expires ___/___

COMPUTE! Books

P.O. Box 5406 Greensboro, NC 27403

Ask your retailer for these **COMPUTE! Books**. If he or she has sold out, order directly from **COMPUTE!**

For Fastest Service
Call Our **TOLL FREE US Order Line**

800-334-0868

In NC call 919-275-9809

Quantity	Title	Price	Total
_____	Machine Language for Beginners	**$14.95***	_____
_____	Home Energy Applications	**$14.95***	_____
_____	COMPUTE!'s First Book of VIC	**$12.95***	_____
_____	COMPUTE!'s Second Book of VIC	**$12.95***	_____
_____	COMPUTE!'s First Book of VIC Games	**$12.95***	_____
_____	COMPUTE!'s First Book of 64	**$12.95***	_____
_____	COMPUTE!'s First Book of Atari	**$12.95***	_____
_____	COMPUTE!'s Second Book of Atari	**$12.95***	_____
_____	COMPUTE!'s First Book of Atari Graphics	**$12.95***	_____
_____	COMPUTE!'s First Book of Atari Games	**$12.95***	_____
_____	Mapping The Atari	**$14.95***	_____
_____	Inside Atari DOS	**$19.95***	_____
_____	The Atari BASIC Sourcebook	**$12.95***	_____
_____	Programmer's Reference Guide for TI-99/4A	**$14.95***	_____
_____	COMPUTE!'s First Book of TI Games	**$12.95***	_____
_____	Every Kid's First Book of Robots and Computers	**$ 4.95†**	_____
_____	The Beginner's Guide to Buying A Personal Computer	**$ 3.95†**	_____

* Add $2 shipping and handling. Outside US add $5 air mail; $2 surface mail.

†Add $1 shipping and handling. Outside US add $5 air mail; $2 surface mail.

Please add shipping and handling for each book ordered. _____

Total enclosed or to be charged. _____

All orders must be prepaid (money order, check, or charge). All payments must be in US funds. NC residents add 4% sales tax.

☐ Payment enclosed Please charge my: ☐ VISA ☐ MasterCard
☐ American Express Acc't. No. _____ Expires ___/___

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Allow 4-5 weeks for delivery.

If you've enjoyed the articles in this book, you'll find the same style and quality in every monthly issue of **COMPUTE!'s Gazette** for Commodore.

For Fastest Service
Call Our **Toll-Free** US Order Line
800-334-0868
In NC call 919-275-9809

COMPUTE!'s GAZETTE
P.O. Box 5406
Greensboro, NC 27403

My computer is:
☐ Commodore 64 (01) ☐ VIC-20 (02) ☐ Other (03) _____

☐ $20 One Year US Subscription
☐ $36 Two Year US Subscription
☐ $54 Three Year US Subscription

Subscription rates outside the US:

☐ $25 Canada
☐ $45 Air Mail Delivery
☐ $25 International Surface Mail

Name _____

Address _____

City _____ State _____ Zip _____

Country _____

Payment must be in US Funds drawn on a US Bank, International Money Order, or charge card. Your subscription will begin with the next available issue. Please allow 4–6 weeks for delivery of first issue. Subscription prices subject to change at any time.

☐ Payment Enclosed ☐ VISA
☐ MasterCard ☐ American Express

Acct. No. _____ Expires ____/____

The *COMPUTE!'s Gazette* subscriber list is made available to carefully screened organizations with a product or service which may be of interest to our readers. If you prefer not to receive such mailings, please check this box ☐.

Whether you've just gotten your Commodore computer or are an experienced VIC-20 or Commodore 64 user, you know that you have a powerful machine at your command. Now, with Volume 1 of *COMPUTE!'s Commodore Collection*, you'll find it even more useful.

This book is a collection of never-before-published programs and articles, each of the high quality you've come to expect from COMPUTE! Publications. It's packed with listings and information that you'll use over and over again.

COMPUTE!'s Commodore Collection, Volume 1, combines the work of dozens of experienced Commodore programmers. From exciting games and educational programs to practical applications and helpful utilities, it's sure to offer something for everyone. Here's a sample of what's inside:

- "Reaction," an unbelievably fast arcade-style game that offers hours of challenging fun while sharpening your reflexes.
- "File Cabinet," a comprehensive file manager to help you organize that stack of papers—or that shelf of books.
- "Spider Math," a delightful educational program to help your child learn addition, subtraction, multiplication, and division.
- "Advertiser," an intriguing commercial application for your VIC or 64, which produces exciting, eye-catching, marquee-style displays.
- Multicolor character editors for both the VIC and the 64.
- An autoscaling bar chart routine, easily incorporated into your own programs.

You don't have to be a computer expert to use this book. Each program has been carefully tested and is ready to type in. Detailed documentation is also included so you can easily customize these programs as you gain experience with your computer.

If you're familiar with COMPUTE! books, you know that you're in for a treat. But if this is your first COMPUTE! book, get ready for a pleasant surprise.

ISBN 0-942386-55-8

COMPUTE!'s Commodore Collection: 1

COMPUTE! Books