Securities Analysis:
A Sophisticated, Ready-To-Run Program For Commodore 64, VIC-20, Atari, And IBM PC And PCjr

The Shakeout Begins, A Watershed CES Show

The Worm Of Bemer:
A Complete Action Game For Commodore 64, VIC-20, Atari, TI-99/4A, And IBM PC And PCjr

Plus Applications, Tutorials, Reviews, And Much More

FEATURES

EDUCATION AND RECREATION

REVIEWS

COLUMNS AND DEPARTMENTS

THE JOURNAL

NOTE: See page 180 before typing in programs.

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

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

Worm Of Bemer

Stephen D. Fultz

Nerm the worm is lost in Bemer Castle and needs your help to get home. You must guide him through 11 rooms and help him find magic mushrooms to eat along the way. The journey is a navigator's nightmare, because you never know where the next mushroom will grow, and if Nerm hits a wall or gets trapped by his tail, he loses one of his lives. Written for the Atari, versions are included for the VIC, 64, TI-99/4A, and IBM PC/PCjr. A joystick is required for some versions.

"Worm of Bemer" is a fast-paced arcade game in which Nerm the Worm travels through rooms eating magic mushrooms. Nerm is lost in Bemer Castle and wants to return home. Guide Nerm to a mushroom so he can keep up his strength for the journey. After eating five mushrooms in a room, Nerm can exit to the next room. You must guide Nerm through 11 rooms before he finds his home. You start out with four lives. If you touch anything besides a mushroom you will lose a life.

At the top of the screen will be the current score, what room Nerm is in, how many mushrooms Nerm must eat to open the exits, and how many lives Nerm has left, including the current life. You get 100 points, plus bonus points, for every mushroom you eat. Nerm gets a bonus life after completing the first two rooms and another for every third room thereafter.

Special Features

The game takes advantage of Atari's graphic capabilities. Special features include custom-designed display lists, a display list interrupt, a redefined character set, and special four-color graphics mode.

Nerm uses four custom display lists. The most important display list is for the main screen. This display list mixes three graphics modes on one screen. The first two display lines are in
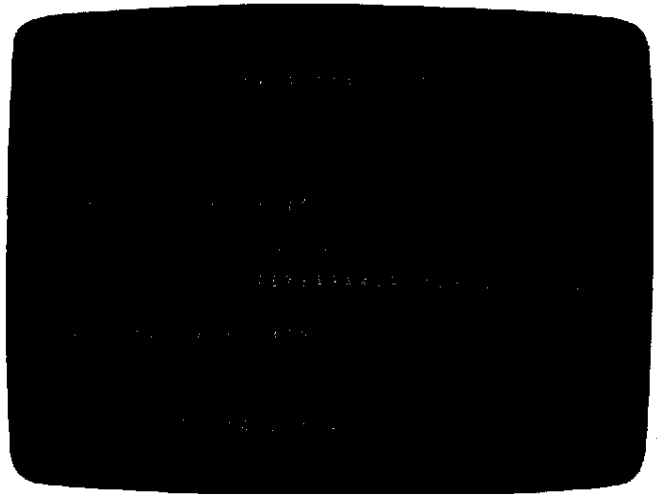
The player has reached Room 7 in the Atari version of "Worm of Bemer."

graphics mode 2. The next ten lines are in a special graphics mode that allows four-color characters. The last part of the screen is in graphics mode 0. A display list interrupt is used to change the background color. The other three custom display lists mix modes 0, 1, and 2.

A special character set was needed to take advantage of the four-color character graphics mode. Characters were redesigned for walls, the mushroom, and the body of Nerm. The original set was copied to a location in memory not used by BASIC, and the new characters added.

Worm of Bemer is written in BASIC with two machine language subroutines; one makes a fast copy of the character set, and the other is the display list interrupt.

Be sure to save the game before running it. If you make a mistake in entering the two machine language subroutines, it is likely that your system will crash and you will have to reboot.

Adding More Features

You can learn a lot about programming and games by modifying the action and settings in Worm of Bemer. Some features you might add include a routine to save the high score to disk, adding more players, or having Nerm go to a different room depending on which exit he takes. Simpler enhancements would be changing the number of mushrooms that Nerm must eat or changing his speed.

Program 1: Worm Of Bemer—Atari Version

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

```
PB 5    SCREEN=PEEK(88)+256*PEEK(89)
FP 10   GOTO 5000
IP 100  POKE 53761,0:S=STICK(0):FOR D=1
        TO SPEED:NEXT D
FF 110  IF S=7 OR S=6 OR S=5 THEN DXA=1
        :DYA=0:DIR=1:IF ODIR=2 THEN DXA
        =-1:DYA=0:DIR=2
KP 120  IF S=11 OR S=10 OR S=9 THEN DXA
        =-1:DYA=0:DIR=2:IF ODIR=1 THEN
        DXA=1:DYA=0:DIR=1
LN 130  IF S=14 THEN DYA=-1:DXA=0:DIR=4
        :IF ODIR=3 THEN DIR=3:DYA=1:DXA
        =0
LD 140  IF S=13 THEN DYA=1:DXA=0:DIR=3:
        IF ODIR=4 THEN DIR=4:DYA=-1:DXA
        =0
NL 145  COLOR 42:PLOT XA,YA:ODIR=DIR
DN 150  XA=XA+DXA:YA=YA+DYA:L=LEN(XA$):
        XA$(L+1)=CHR$(XA):YA$(L+1)=CHR$
        (YA):LOCATE XA,YA,Z:IF Z<>32 TH
        EN 200
IB 162  SOUND 0,40,8,6:COLOR 170:PLOT X
        A,YA:IF L<WORMZ THEN 190
NH 190  COLOR 32:PLOT ASC(XA$),ASC(YA$)
        :XA$=XA$(2):YA$=YA$(2):GOTO 100
PD 200  SOUND 0,200,10,16:POKE SCREEN+Y
        A*40+XA,132:GOSUB 6600:IF Z<>BU
        G THEN 260
DD 210  WORMZ=WORMZ+15+(3*LOC):IF WORMZ
        >240 THEN WORMZ=240
AD 220  XX=RND(2)*36+2:X=RND(1)*18+2:CO
        LOR BUG:LOCATE XX,X,Y:IF Y<>32
        THEN 220
HD 221  SCORE=SCORE+100+LOC*7:FOR DEL=8
        TO 16:SOUND 0,55,10,17-DEL:NEX
        T DEL
HD 226  HIT=HIT-1:IF HIT<1 THEN COLOR 1
        60:PLOT 20,1:PLOT 19,20:PLOT 0,
        12:POKE SCREEN+12*40+39,128:HIT
        =0:GOTO 100
GE 227  GOSUB 6600:PLOT XX,X:SOUND 0,10
        0,10,16
FP 230  GO TO 100
HP 260  IF Z<>160 AND LIVES>1 THEN CNT
        =CNT-1:GOSUB 7500:GOTO 290
GK 265  IF Z<>160 THEN CNT=CNT-1:GOTO
        7500
MG 270  FOR DEL=1 TO 16:SOUND 0,90,10,1
        7-DEL:NEXT DEL
FB 271  GOSUB 7000:POSITION 0,23
PE 273  FOR DEL=1 TO 24:PRINT :SOUND 0,
        DEL,10,10:NEXT DEL
MG 280  LOC=LOC+1:WORMZ=5:CNT=0
EH 285  IF LOC>EXTRA THEN GOSUB 9100
EL 290  GRAPHICS 0:COLOR 35:POKE 752,1:
```

```
        POKE 710,0:GOSUB 6500:POKE 712,
        162
DH 291  POKE 710,ASC(BOL$(LOC))
PD 300  ON LOC GOTO 5020,400,500,550,60
        0,700,800,450,550,1000,1100,1200
KJ 399  GO TO 5015
PE 400  REM SECOND SCREEN
JG 410  PLOT 5,10:DRAWTO 35,10
JG 420  GO TO 5020
DN 450  REM SCREEN
OA 460  PLOT 5,10:DRAWTO 35,10:PLOT 18,
        5:DRAWTO 18,20
JL 470  GO TO 5020
KA 500  REM THE FOUTH SCREEN
DP 510  PLOT 5,5:DRAWTO 35,5
KE 520  PLOT 5,16:DRAWTO 35,16
JI 530  GO TO 5020
BD 550  REM FRAME 5
EG 560  PLOT 7,6:DRAWTO 33,6
KB 575  PLOT 18,7:DRAWTO 18,20
JN 580  GO TO 5020
NG 600  REM FRAM 6
AC 610  PLOT 1,10:DRAWTO 18,10:PLOT 22,
        10:DRAWTO 38,10
JI 620  GOTO 5020
NI 700  REM FRAM 7
EB 710  FOR I=6 TO 14:PLOT 6,I:DRAWTO 1
        2,I:PLOT 20,I:DRAWTO 32,I:NEXT I
JJ 720  GOTO 5020
NK 800  REM FRAM 8
EG 811  PLOT 1,8:DRAWTO 18,8
KD 812  PLOT 1,15:DRAWTO 18,15
MF 813  PLOT 15,12:DRAWTO 38,12
KB 890  GOTO 5020
AB 900  REM THE 8 FRAME
KG 910  FOR I=5 TO 12 STEP 3:PLOT 11,I:
        DRAWTO 30,I:NEXT I
JL 920  GO TO 5020
DH 1000 FOR I=2 TO 19:PLOT 1,I:DRAWTO
        38,I:NEXT I:COLOR 32:FOR I=2 T
        O 19:PLOT 1,I:DRAWTO 22,I:NEXT
        I:GOTO 5020
CD 1100 FOR I=2 TO 19:PLOT 1,I:DRAWTO
        38,I:NEXT I:COLOR 32:FOR I=2 T
        O 19:PLOT 1,I:DRAWTO 30,I:NEXT
        I:COLOR 35
JA 1101 GOTO 400
JC 1200 REM YOU WIN
JD 1205 FOR QZ=1 TO 3
HD 1210 GRAPHICS 18
ED 1211 POSITION 4,5:PRINT #6;"NERM'S
        HOME"
JU 1212 PRINT #6;"(5 SPACES)THANK YOU"
DN 1215 FOR G=1 TO 5
GH 1220 FOR I=1 TO 10
FA 1229 SOUND 1,I+200,10,16-I
MA 1230 SOUND 0,I+6,10,I+5:POKE 712,I*
        16
AP 1240 NEXT I:NEXT G
DI 1245 FOR I=1 TO 200:SOUND 0,I,10,15
        :NEXT I
PD 1250 NEXT QZ:GOTO 7700
PE 5000 GOSUB 10000:REM UP THE GAME
PI 5005 GOSUB 11100:BUG=33
EN 5010 DIM XA$(250),YA$(250),XB$(250)
        ,YB$(250),A$(15),B$(15)
EB 5011 SPEED=35:LIVES=4:SCORE=0:LOC=1
        :GOSUB 5500:HIT=5:WORMZ=5:EXTR
        A=2
BC 5015 GRAPHICS 0:POKE 752,1:POKE 710
        ,0:GOSUB 6500:POKE 710,ASC(BOL
        $(LOC)):POKE 77,0:POKE 712,162
FA 5020 XA$="":YA$="":XB$="":YB$="":XA
```

```
11270 RETURN                          :rem 219
12000 DATA 195,209,0,209,215,0,215,219,22
      5,219,225,219,209,0,195,209,0,209
                                       :rem 52
12100 DATA 209,0,0,195,191,195,201,201,19
      5,0,0,0,207,207,209            :rem 108
```

Program 4: Worm Of Bemer—64 Version
Translation by Kevin Martin, Editorial Programmer

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

```
1 POKE52,48:POKE56,48:CLR             :rem 230
2 POKE53270,PEEK(53270)AND15          :rem 62
5 POKE53280,0:POKE53281,0             :rem 138
10 GOTO5000                           :rem 95
100 S=PEEK(56320)AND15:FORD=1TOSP:NEXT
                                      :rem 98
110 IFS=7ORS=6ORS=5THENDX=1:DY=0:DI=1:IFO
    D=2THENDX=-1:DY=0:DI=2             :rem 18
120 IFS=11ORS=10ORS=9THENDX=-1:DY=0:DI=2:
    IFOD=1THENDX=1:DY=0:DI=1          :rem 108
130 IFS=14THENDY=-1:DX=0:DI=4:IFOD=3THEND
    I=3:DY=1:DX=0                     :rem 122
140 IFS=13THENDY=1:DX=0:DI=3:IFOD=4THENDI
    =4:DY=-1:DX=0                     :rem 123
145 PO=1024+XA+YA*40:OD=DI:POKEPO,42:POKE
    PO+SO,L1                          :rem 3
150 XA=XA+DX:YA=YA+DY:L=LEN(XA$):XA$=XA$+
    CHR$(XA):YA$=YA$+CHR$(YA)         :rem 0
155 Z=PEEK(1024+XA+YA*40):IFZ<>32THEN200
                                      :rem 73
161 POKESO+1,40:POKESO+4,17           :rem 83
162 PO=1024+XA+YA*40:POKEPO,42:POKEPO+SO,
    10:POKESO+4,16:IFL<WOTHEN100      :rem 3
190 PO=1024+ASC(XA$)+40*ASC(YA$):LL=LEN(X
    A$)-1:XA$=RIGHT$(XA$,LL)          :rem 238
191 POKEPO,32:POKEPO+SO,0             :rem 43
195 YA$=RIGHT$(YA$,LL):GOTO100        :rem 19
200 POKESO+1,20:POKESO+4,17:POKESO+4,16
                                      :rem 72
201 PO=1024+XA+40*YA:POKEPO,42:POKEPO+SO,
    10:GOSUB6600:IFZ<>BUTHEN260       :rem 202
210 WO=WO+15+3*LO:IFWO>240THENWO=240
                                      :rem 187
220 XX=INT(RND(1)*36+2):X=INT(RND(1)*18+3
    ):IFPEEK(1024+XX+40*X)<>32THEN220
                                      :rem 9
221 SC=SC+100+LO*7:POKESO+4,55:POKESO+4,1
    7                                 :rem 223
225 HI=HI-1:GOSUB6600:IFHI>0THEN229
                                      :rem 112
226 PO=1024+20+40*2:POKEPO,160:POKEPO+SO,
    0:PO=1024+20+21*40:POKEPO,160     :rem 2
227 POKEPO+SO,0:PO=1024+40*12:POKEPO,160:
    POKEPO+SO,0:POKESO+1,100:POKESO+4,17
                                      :rem 169
228 PO=1024+40*12+39:POKEPO,160:POKEPO+SO
    ,0:GOTO100                        :rem 242
229 PO=1024+XX+X*40:POKEPO,BUG:POKEPO+SO,
    13                                :rem 163
230 GOTO100                           :rem 95
260 IFZ<>160ANDLI>1THENGOSUB7500:GOTO290
                                      :rem 242
265 IFZ<>160THEN7500                  :rem 146
270 POKESO+1,90:POKESO+4,17           :rem 89
271 GOSUB7000:PRINT"{HOME}{24 DOWN}"
                                      :rem 151
275 FORDE=1TO24:PRINT:POKESO+1,DEL:POKESO
    +4,17:NEXT:POKESO+4,16            :rem 40
280 LO=LO+1:WO=5:IFLO=12THEN1200     :rem 177
```

```
281 L1=L1+1:IFL1>15THENL1=11          :rem 99
285 IFLO>EXTHENGOSUB9100              :rem 29
287 PRINT"{CLR}":GOSUB 4100          :rem 132
290 GOSUB6600                        :rem 231
300 ONLO GOTO5020,400,500,550,600,700,800
    ,450,550,1000,1100,1200          :rem 176
399 GOTO5015                         :rem 169
400 REM SECOND SCREEN                :rem 244
```

```
7005 L=LEN(XA$)                    :rem 66
7010 FORI=1TOL-1                   :rem 179
7020 POKESO+1,I:POKESO+4,17:FORQQ=1TO10:N
     EXT                          :rem 23
7190 PO=1024+ASC(XA$)+40*ASC(YA$):LL=LEN(
     XA$)-1:XA$=RIGHT$(XA$,LL)     :rem 37
7195 YA$=RIGHT$(YA$,LL):POKEPO,32:POKEPO+
     SO,1                         :rem 17
7200 NEXT:POKESO+4,16             :rem 5
7210 RETURN                       :rem 170
7500 REM OOPS                     :rem 241
7510 PRINT"{CLR}{PUR}"            :rem 205
7515 SP=SP-5                      :rem 183
7520 PRINT"{12 DOWN}{18 RIGHT}OOPS"
                                  :rem 182
7521 LI=LI-1                      :rem 148
7525 FORDE=1TO20:NEXT             :rem 47
7530 FORDE=1TO10:POKESO+1,DE*20:POKESO+4,
     17:FORQQ=1TO10:NEXT:NEXT:POKESO+4,16
                                  :rem 123
7550 FORDE=1TO20:NEXT             :rem 45
7560 IFLI<1THEN7700               :rem 96
7599 PRINT"{CLR}":RETURN          :rem 92
7700 REM THE GAMES OVER           :rem 60
7705 POKESO+4,16                  :rem 150
7710 PRINT"{CLR}":POKE53270,PEEK(53270)AN
     D15                          :rem 121
7715 IF SC>HSTHENHS=SC:GOSUB9000:PRINT"
     {CLR}{8}"                    :rem 43
7718 PRINT"{6 DOWN}{18 RIGHT}NERM":rem 74
7720 PRINT"{YEL}{4 DOWN}YOUR SCORE ";SC
                                  :rem 31
7730 PRINT"{6}{4 DOWN}HIGH SCORE ";HS
                                  :rem 241
7735 GOSUB 7800                   :rem 37
7736 PRINT"{WHT}{2 DOWN}PRESS THE TRIGGER
      TO PLAY AGAIN, Q=QUIT"      :rem 135
7740 FORX=1TO15:POKESO+1,PN(X):POKESO+4,1
     7:FORD=1TO100:NEXT:NEXT      :rem 89
7745 POKESO+4,16                  :rem 154
7780 S=PEEK(56320)AND16:IFS=0THEN5011
                                  :rem 149
7783 IFPEEK(197)=62THENPOKE198,0:SYS2048
                                  :rem 135
7785 GOTO7780                     :rem 234
7800 REM RANK THE GAMER           :rem 44
7810 PRINT"{CYN}{2 DOWN}{7 SPACES}YOUR NE
     W RANK IS ";                 :rem 158
7820 IFLO=1THENPRINT"ZERO"        :rem 169
7830 IFLO=2THENPRINT"ROOKIE"      :rem 52
7840 IFLO=3THENPRINT"NOVICE"      :rem 49
7850 IFLO=4THENPRINT"AVERAGE"     :rem 106
7860 IFLO=5THENPRINT"MASTER"      :rem 61
7870 IFLO=6THENPRINT"GRAND MASTER"
                                  :rem 171
7880 IFLO=7THENPRINT"WIZARD"      :rem 70
7890 IFLO=8THENPRINT"GRAND WIZARD"
                                  :rem 180
7900 IFLO=9THENPRINT"SUPER STAR"  :rem 57
7910 IFLO>9THENPRINT"HALL OF FAME":rem 65
7920 RETURN                       :rem 178
9000 REM NEW HIGH SCORE           :rem 51
9002 PRINT"{CLR}"                 :rem 47
9003 PRINT"{CYN}{8 DOWN}{16 RIGHT}NEW HIG
     H"                           :rem 158
9004 PRINT"{4 DOWN}{17 RIGHT}SCORE"
                                  :rem 75
9005 FORY=1TO3                    :rem 82
9010 FORN=1TO5                    :rem 69
9020 FORD=1TO5:POKESO+1,D*20:POKESO+4,17:
     NEXT                         :rem 22
```

```
9030 POKESO+1,N*30:POKESO+4,17    :rem 254
9050 NEXT                         :rem 13
9060 NEXT                         :rem 14
9065 FORD=1TO30:NEXT              :rem 236
9070 RETURN                       :rem 176
9100 REM EXTRA LIFE               :rem 82
9110 PRINT"{CLR}"                 :rem 47
9115 PRINT"{CYN}{12 DOWN}{15 RIGHT}BONUS
     {SPACE}LIFE"                 :rem 102
9120 FORJ=100TO200                :rem 0
9140 POKESO+1,J:POKESO+4,17       :rem 111
9150 NEXT                         :rem 14
9160 POKESO+4,16                  :rem 147
9170 EX=EX+3                      :rem 166
9180 LI=LI+1                      :rem 149
9190 RETURN                       :rem 179
10000 DIM PN(15),NN(18):PRINT"{CLR}{8}"
                                  :rem 219
10005 SO=54272:POKESO+24,15:POKESO+5,17:P
      OKESO+6,241:POKESO,100      :rem 253
10010 PRINT"{6 DOWN}{15 RIGHT}WELCOME TO"
                                  :rem 139
10020 PRINT"{CYN}{4 DOWN}{14 RIGHT}NERM O
      F BEMER"                    :rem 111
10025 PRINT"{1}{3 DOWN}{13 RIGHT}USE JO
      YSTICK #2"                  :rem 168
10030 PRINT"{6}{6 DOWN}{11 RIGHT}HIT TR
      IGGER TO START"             :rem 45
10045 S=PEEK(56320)AND16:IFS=0THENRETURN
                                  :rem 210
10060 GOTO10045                   :rem 42
11100 PRINT"{CLR}{CYN}{12 DOWN}{9 RIGHT}R
      EDEFINING{2 SPACES}CHARACTERS"
                                  :rem 130
11109 POKE56334,PEEK(56334)AND254:POKE1,P
      EEK(1)AND251               :rem 29
11110 FORI=12288TO12288+64*8:POKEI,PEEK(I
      +40960):NEXTI              :rem 146
11120 POKE1,PEEK(1)OR4:POKE56334,PEEK(563
      34)OR1                     :rem 228
11180 FORI=0TO39:READA:POKE12288+I+32*8,A
      :NEXT                      :rem 248
11185 FORI=0TO7:READA:POKE12288+I+42*8,A:
      NEXT                       :rem 201
11190 POKE53272,(PEEK(53272)AND240)+12
                                  :rem 27
11200 FORI=1TO18:READNN(I):NEXT   :rem 163
11210 FORI=1TO15:READPN(I):NEXT   :rem 163
11240 DATA 0,0,0,0,0,0,0,0        :rem 198
11250 DATA 0,20,85,85,255,40,40,0 :rem 71
11260 DATA 85,85,85,85,85,85,85,85
                                  :rem 176
11261 DATA 170,190,190,190,190,190,170,17
      0                          :rem 19
11262 DATA 255,255,255,255,255,255,255,25
      5                          :rem 42
11263 DATA 0,20,255,255,255,255,20,0
                                  :rem 223
11270 RETURN                      :rem 219
12000 DATA 33,44,0,44,50,0,50,56,67,56,67
      ,56,44,0,33,44,0,44         :rem 114
12100 DATA 44,0,0,33,31,33,37,37,33,0,0,0
      ,42,42,44                   :rem 108
```
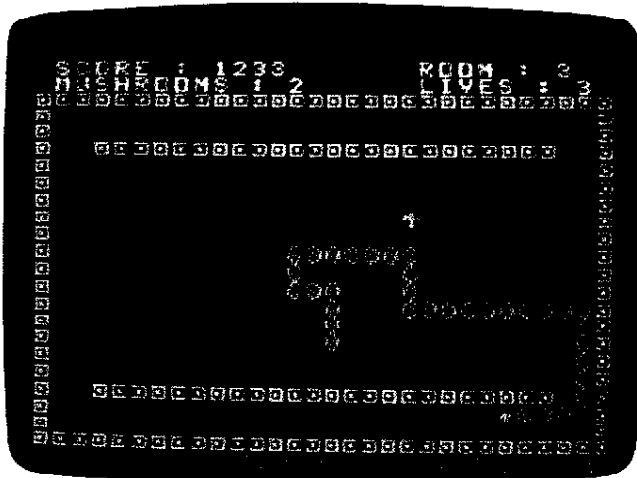
Program 5: Worm Of Bemer—TI Version

Translation by Patrick Parrish, Programming Supervisor

```
3 DIM NN(29),RANK$(12)
5 GOSUB 11000
10 GOTO 5000
20 FOR I=1 TO LEN(H$)
```

TI version of "Worm of Bemer"

```
30 CALL HCHAR(ROW,COL+I,ASC(SEG$(H$
   ,I,1)))
35 NEXT I
40 RETURN
100 CALL KEY(0,K,ST)
105 IF (K<>68)+(OD=2)THEN 110
106 DX=1
107 DY=0
108 DI=1
110 IF (K<>83)+(OD=1)THEN 115
111 DX=-1
112 DY=0
113 DI=2
115 IF (K<>69)+(OD=3)THEN 120
116 DY=-1
117 DX=0
118 DI=4
120 IF (K<>88)+(OD=4)THEN 140
125 DY=1
130 DX=0
135 DI=3
140 CALL HCHAR(YA,XA,136)
145 OD=DI
150 XA=XA+DX
152 YA=YA+DY
154 L=LEN(XA$)
156 XA$=XA$&CHR$(XA)
158 YA$=YA$&CHR$(YA)
160 CALL GCHAR(YA,XA,Z)
162 IF Z<>32 THEN 200
164 CALL HCHAR(YA,XA,128)
166 CALL SOUND(1,622,2)
168 IF L<WO THEN 100
170 CALL HCHAR(ASC(YA$),ASC(XA$),32)
172 LL=LEN(XA$)-1
174 XA$=SEG$(XA$,2,LL)
176 YA$=SEG$(YA$,2,LL)
180 GOTO 100
200 CALL SOUND(100,311,2)
201 CALL HCHAR(YA,XA,128)
203 GOSUB 6600
205 IF Z<>MUSH THEN 260
210 WO=WO+15+2*LO
212 IF WO<185 THEN 215
214 WO=185
215 RANDOMIZE
216 XX=RND*28+3
218 X=RND*19+4
220 CALL GCHAR(X,XX,H1)
222 IF H1<>32 THEN 216
```

```
224 SC=SC+100+LO*7
228 HI=HI-1
230 GOSUB 6600
232 IF HI>0 THEN 245
234 CALL HCHAR(3,17,104)
236 CALL HCHAR(13,2,104)
238 CALL HCHAR(13,31,104)
240 CALL HCHAR(23,17,104)
241 FOR I=3 TO 30 STEP 3
242 CALL SOUND(100,1900,I)
243 NEXT I
244 GOTO 100
245 CALL HCHAR(X,XX,MUSH)
250 GOTO 100
260 IF Z=104 THEN 270
261 IF LI=1 THEN 7500
264 GOSUB 7500
266 GOTO 290
270 CALL HCHAR(YA,XA,136)
272 GOSUB 7000
275 FOR DE=110 TO 880 STEP 32
277 PRINT
279 CALL SOUND(1,DE,2)
280 CALL SOUND(-1,DE,2)
281 NEXT DE
282 LO=LO+1
283 IF LO=12 THEN 1200
284 WO=5
285 L1=L1+1
286 IF LO>EX THEN 9100
287 CALL COLOR(14,L1,1)
288 CALL CLEAR
289 GOSUB 1300
290 GOSUB 6600
300 ON LO GOTO 5080,400,500,550,600
   ,700,800,450,550,1000,1100,1200
399 GOTO 5080
400 REM   SECOND SCREEN
410 CALL HCHAR(13,5,120,24)
420 GOTO 5080
449 REM   SCREEN
450 CALL VCHAR(7,15,120,16)
455 CALL HCHAR(9,6,120,22)
460 GOTO 5080
499 REM   FOURTH SCREEN
500 CALL HCHAR(6,5,120,24)
505 CALL HCHAR(20,5,120,24)
510 GOTO 5080
549 REM   FIFTH SCREEN
550 CALL HCHAR(7,6,120,22)
555 CALL VCHAR(8,15,120,16)
560 GOTO 5080
599 REM   FRAME 6
600 CALL HCHAR(12,3,120,13)
610 CALL HCHAR(12,19,120,12)
620 GOTO 5080
699 REM   FRAME 7
700 FOR I=8 TO 18
710 CALL HCHAR(I,7,120,7)
715 CALL HCHAR(I,18,120,8)
720 NEXT I
725 GOTO 5080
799 REM   FRAME 8
800 CALL HCHAR(8,3,120,13)
805 CALL HCHAR(14,12,120,19)
810 CALL HCHAR(18,3,120,13)
815 GOTO 5080
999 REM   FRAME 9
1000 GOSUB 1400
1015 FOR T=5 TO 21
1020 CALL HCHAR(T,4,32,16)
```

```
1025 NEXT T
1030 GOTO 5080
1100 GOSUB 1400
1110 FOR T=5 TO 21
1115 CALL HCHAR(T,4,32,20)
1120 NEXT T
1125 GOTO 400
1199 REM   YOU WIN!!
1200 CALL CLEAR
1205 CALL SCREEN(3)
1206 FOR I=4 TO 8
1207 CALL COLOR(I,2,1)
1208 NEXT I
1210 PRINT TAB(9);"NERM'S HOME!"
1220 PRINT
1230 PRINT
1240 PRINT TAB(10);"THANK YOU!"
1250 FOR T=1 TO 9
1260 PRINT
1270 NEXT T
1275 FOR T=1 TO 3
1280 FOR I=110 TO 880 STEP 30
1283 CALL SOUND(1,I,2)
1284 CALL SOUND(-1,I,2)
1285 NEXT I
1286 FOR I=880 TO 110 STEP -30
1287 CALL SOUND(1,I,2)
1288 CALL SOUND(-1,I,2)
1289 NEXT I
1290 NEXT T
1291 CALL SCREEN(2)
1293 GOTO 7700
1300 CALL CLEAR
1305 PRINT "SCORE :";TAB(20);"ROOM
     :"
1310 PRINT "MUSHROOMS :";TAB(20);"L
     IVES :"
1320 FOR T=1 TO 21
1330 PRINT
1340 NEXT T
1350 RETURN
1400 FOR T=5 TO 21
1410 CALL HCHAR(T,4,120,26)
1420 NEXT I
1430 RETURN
4999 REM   UP THE GAME
5000 GOSUB 10000
5005 MUSH=112
5010 LI=4
5015 SC=0
5020 LO=1
5035 HI=5
5040 WO=5
5045 EX=2
5050 LI=3
5055 GOSUB 5500
5060 CALL CLEAR
5065 CALL SCREEN(2)
5066 FOR I=3 TO 8
5067 CALL COLOR(I,16,1)
5068 NEXT I
5070 GOSUB 1300
5075 GOSUB 6600
5080 XA$=""
5081 YA$=""
5085 XA=17
5086 YA=19
5091 DX=0
5093 DY=-1
5103 IF HI<6 THEN 5107
5105 HI=5
5107 IF HI>-1 THEN 5110
5109 HI=0
5110 DI=4
5115 FOR I=2 TO 31 STEP 29
5120 CALL VCHAR(3,I,120,21)
5125 NEXT I
5130 FOR I=3 TO 23 STEP 20
5135 CALL HCHAR(I,3,120,28)
5140 NEXT I
5145 CALL HCHAR(24,3,137,28)
5150 IF HI>0 THEN 5174
5155 CALL HCHAR(3,17,104)
5160 CALL HCHAR(12,2,104)
5165 CALL HCHAR(12,31,104)
5167 CALL HCHAR(23,17,104)
5171 GOTO 150
5174 RANDOMIZE
5175 XX=RND*28+3
5178 X=RND*19+4
5180 CALL GCHAR(X,XX,H1)
5185 IF H1<>32 THEN 5174
5190 CALL HCHAR(X,XX,MUSH)
5200 GOTO 150
5500 CALL CLEAR
5505 PRINT TAB(10);"GET READY!"
5510 FOR T=1 TO 12
5515 PRINT
5520 NEXT T
5525 FOR I=1 TO 14
5530 CALL SOUND(100,NN(I),2)
5535 NEXT I
5540 RETURN
6599 REM   PRINT SCORE
6600 H$=STR$(SC)
6603 ROW=1
6604 COL=10
6605 GOSUB 20
6607 H$=STR$(LO)
6608 COL=28
6609 GOSUB 20
6610 H$=STR$(HI)
6611 ROW=2
6620 COL=14
6625 GOSUB 20
6630 H$=STR$(LI)
6635 COL=29
6640 GOSUB 20
6650 RETURN
6999 REM   NERM LEAVES
7000 SP=SP-5
7005 GOSUB 6600
7010 HI=5
7015 L=LEN(XA$)
7020 FOR I=1 TO L
7025 CALL SOUND(2,110+I*2,2)
7030 CALL HCHAR(ASC(YA$),ASC(XA$),3
     2)
7035 LL=LEN(XA$)-1
7040 XA$=SEG$(XA$,2,LL)
7045 YA$=SEG$(YA$,2,LL)
7050 NEXT I
7060 RETURN
7499 REM   OOP!!
7500 CALL CLEAR
7505 PRINT TAB(13);"OOPS"
7510 FOR I=1 TO 12
7515 PRINT
7520 NEXT I
7525 LI=LI-1
7547 FOR I=14 TO 24
7549 CALL SOUND(10,I*40,2)
7551 NEXT I
7553 FOR I=1 TO 30
```

```
7555 NEXT I
7560 IF LI<1 THEN 7700
7575 GOSUB 1300
7600 RETURN
7699 REM    THE GAME ENDS
7700 CALL CLEAR
7704 FOR I=3 TO 8
7705 CALL COLOR(I,16,1)
7706 NEXT I
7710 IF HS>SC THEN 7750
7720 HS=SC
7721 FOR I=1 TO 5
7722 PRINT
7723 NEXT I
7725 PRINT TAB(8);"NEW HIGH SCORE"
7728 FOR T=110 TO 1760 STEP 50
7729 CALL SOUND(2,T,2)
7730 NEXT T
7740 FOR I=1 TO 5
7743 PRINT
7745 NEXT I
7750 PRINT TAB(7);"YOUR SCORE: ";SC
7755 PRINT
7760 PRINT TAB(7);"HIGH SCORE: ";HS
7770 FOR I=1 TO 5
7775 PRINT
7780 NEXT I
7785 PRINT TAB(5);"YOUR NEW RANK IS
     :"
7790 PRINT
7795 PRINT TAB(9);RANK$(LO)
7796 FOR I=15 TO 29
7797 CALL SOUND(100,NN(I),2)
7798 NEXT I
7800 PRINT
7805 PRINT
7806 PRINT
7810 PRINT "(C TO CONTINUE   Q TO QU
     IT)"
7815 FOR T=1 TO 4
7816 PRINT
7817 NEXT T
7820 CALL KEY(0,K,ST)
7830 IF ST=0 THEN 7820
7840 IF (K<>67)*(K<>81)THEN 7820
7845 IF K=67 THEN 5000
7850 STOP
7899 REM    EXTRA LIFE
9100 CALL CLEAR
9110 PRINT TAB(11);"BONUS LIFE"
9120 FOR I=1 TO 12
9125 PRINT
9130 NEXT I
9132 FOR I=1 TO 30 STEP 2
9134 CALL SOUND(100,1175,I)
9136 NEXT I
9140 EX=EX+3
9145 LI=LI+1
9150 GOTO 287
10000 CALL CLEAR
10001 FOR T=3 TO 8
10003 CALL COLOR(T,2,1)
10006 NEXT T
10010 CALL COLOR(14,3,1)
10015 CALL SCREEN(15)
10020 PRINT TAB(10);"WELCOME TO"
10021 FOR T=1 TO 4
10022 PRINT
10023 NEXT T
10025 PRINT TAB(8);"NERM OF BEMER"
10028 FOR T=1 TO 9
10030 PRINT
```

```
10032 NEXT T
10034 PRINT "USE E,S,D, & X KEYS TO
      MOVE"
10036 PRINT
10040 CALL HCHAR(21,3,136,4)
10042 CALL HCHAR(21,8,128)
10045 FOR I=1 TO 22
10047 CALL HCHAR(21,6+I,136)
10050 CALL HCHAR(21,7+I,128)
10052 CALL SOUND(10,622,2)
10055 CALL HCHAR(21,2+I,32)
10057 FOR T=1 TO 20
10058 NEXT T
10060 NEXT I
10065 FOR T=1 TO 100
10070 NEXT T
10075 RETURN
10999 REM    REDEFINE CHARS
11000 FOR I=104 TO 136 STEP 8
11015 READ A$
11020 CALL CHAR(I,A$)
11025 NEXT I
11030 DATA FFFFFFFFFFFFFFFF,187EFFF
      F18181818,FF81BDA5A5BD81FF
11032 DATA 8142243C7E5A3C18,387CFEF
      EFEFE7C38
11033 CALL COLOR(10,2,2)
11035 CALL COLOR(11,14,1)
11040 CALL COLOR(12,2,10)
11045 CALL COLOR(13,7,1)
11050 CALL CHAR(137,"FFFFFFFFFFFFFF
      FF")
11060 FOR I=1 TO 9
11065 READ RANK$(I)
11070 NEXT I
11075 FOR I=10 TO 12
11080 RANK$(I)="HALL OF FAME"
11085 NEXT I
11090 DATA ZERO,ROOKIE,NOVICE,AVERA
      GE
11092 DATA MASTER,GRAND MASTER,WIZA
      RD,GRAND WIZARD
11094 DATA SUPER STAR
11100 FOR I=1 TO 27
11110 READ NN(I)
11120 NEXT I
11130 DATA 262,349,40000,349,392,40
      000,392,440,523,440,523,440,3
      49,40000
11135 DATA 349,40000,40000,262,247,
      262,294,294,262,40000,40000,4
      0000,330,330,349
11140 RETURN
```

Program 6: Worm Of Bemer—PC/PCjr Version

Translation by Patrick Parrish, Programming Supervisor

```
2 DEF SEG=0:POKE 1047,80:SCREEN 0,1
4 WIDTH 40:KEY OFF
5 DEF SEG=&HB800
7 DIM RANK$(12):GOSUB 11000
10 GOTO 5000
100 A$=RIGHT$(INKEY$,1):FOR D=1 TO SP:NE
XT D
110 IF A$=CHR$(77) THEN DX=1:DY=0:DI=1:I
F OD=2 THEN DX=-1:DY=0:DI=2
120 IF A$=CHR$(75) THEN DX=-1:DY=0:DI=2:
IF OD=1 THEN DX=1:DY=0:DI=1
130 IF A$=CHR$(72) THEN DY=-1:DX=0:DI=4:
IF OD=3 THEN DI=3:DY=1:DX=0
140 IF A$=CHR$(80) THEN DY=1:DX=0:DI=3:I
F OD=4 THEN DI=4:DY=-1:DX=0
```

long and ID is a two-character identification code. Because of the way the format command works, this should be done only when the disk is new, or when the files on the disk are no longer needed.

As you can see, when a disk is formatted, any programs that were on that disk are lost.

TI LIST And RUN Protector

I would like to know if there is a way to put an access code on a program for the TI-99/4A. In my family I do all the work and everybody else does all the playing; but they cannot play if it has a code on it.

Aaron Rawn

C. Regena replies:

At the beginning of your program you could try these lines:

```
100 A$="HELLO"
110 INPUT "ENTER CODE: ":C$
120 IF C$=A$ THEN 200
130 PRINT "SORRY, CODE NOT ACCEPTED
    ."
140 STOP
200 PRINT "PROGRAM CONTINUES."
```

When you RUN your program, the computer first asks you to enter the code. If you type in the correct code name the program will continue; otherwise, the program will stop. We used HELLO for the code name. Note that anybody can just LIST the program and get your code name. Instead of HELLO between the quote marks type in your own code word, but hold down the CONTROL key (to the left of the space bar) while you are typing. Now if someone LISTs the program they can't read what your word is—it's either spaces or some funny-looking characters.

VIC Memory Requirements

Say a VIC program requires 11K, and you have only a 16K memory expansion cartridge. Will the program run with the extra memory, or are you required to use the exact memory needed?

My second question is about word processing. When typing with or using a word processor, do you actually need a printer? All I'd really like to do is store my information on disk.

Dwain Young

When a program or article states requires 11K, this means it needs 11K bytes of memory or more. Both the VIC and 64 have pointers that manage BASIC memory. Examples are memory locations 45 and 46, which point to the start of BASIC variables (the end of the BASIC program proper), and bytes 55 and 56, which point to the highest address used by BASIC. These pointers automatically manage the extra memory for you.

However, on the VIC you have another thing to consider—screen and color memory locations. In the

unexpanded VIC, the screen, color, and BASIC memory areas start at locations 7680, 38400, and 4096, respectively. When you plug in a 3K expander (like the Super Expander cartridge), these locations do not change. However, when you plug in one of the larger expansion cartridges (like the 8K or the 16K), these memory locations do change, to 4096, 37888, and 4608, respectively.

This is important with programs that perform POKEs or PEEKs to color or screen memory. For example, a program that POKEs or PEEKs screen or color memory and states needs 3K expansion might not work with an 8K expander because of the relocation of these memory blocks.

On the other hand, a program written for the unexpanded VIC that contains no POKEs or PEEKs to screen or color memory should run with any memory expansion. If you LOAD a program and it doesn't run properly, simply turn off your VIC, change memory expansion cartridges, and reLOAD your program. This will not harm the BASIC program. Just be sure you SAVE your BASIC programs before turning off your VIC.

No, you don't need a printer to use that word processor, if onscreen viewing and storing the text to disk or tape are all you require.

Extra Functions For SuperBASIC 64

The "SuperBASIC 64" program (COMPUTE!, December 1983) is very interesting and helpful. One of its good features is the clarity with which it is laid out. This makes it easy to write and test modifications.

Two sample modifications are given below. To use them, simply LOAD and RUN SuperBASIC 64, then enter and RUN the modification as a normal BASIC program.

The first program creates a new function, [SCRN. [SCRN sets the background, border, and text colors in one operation. For example, [SCRN12,11,1 sets up a light gray background, dark gray border, and white letters.

```
1 REM CREATE [SCRN FUNCTION
10 FORI=0TO9:READA:POKE52168+I,A:NEXT

20 DATA32,57,195,32,66,195,32,218,197,96

30 FORJ=0TO5:READA:POKE49892+J,A:NEXT

40 DATA83,67,82,78,199,203
```

The second program modifies the [DRAW command, allowing it to either draw or erase lines. In its new version, [DRAW must be followed by either 1 or 0. [DRAW1,x1,y1,x2,y2 will draw a line from x1,y1 to x2,y2. [DRAW0,x1,y1,x2,y2 will erase the line between those coordinates.

```
1 REM MODIFY [DRAW TO ALLOW ERASURE
```

In the newer versions of the 64, there is no default filling of color memory. When you POKE characters to the screen (locations 1024–2047) in the newer 64s, you also have to POKE values to the corresponding location in color RAM (bytes 55296–56319). For example, if you POKE 1025,1 (the second position in screen RAM) you also need to POKE 55297,x (the corresponding second position in color RAM). The x can be a value from 0 to 15, corresponding to the 16 character colors available on the 64.

It's good practice to always POKE color memory when POKEing screen memory. That way, your programs will run on any version of the Commodore 64.

Modem Tariff Update

I am happy to report that Southwestern Bell has eliminated the requirement that modem users obtain Information Terminal Service for their modems. While the problem has been taken care of here in Oklahoma, I have not heard anything which rescinds rumors of similar tariffs in other states. Therefore, I would like to extend an invitation to your readers to telephone me should they experience similar problems with their local telephone company.

Robert Braver
MCI Mail: RBRAVER
Source: STY801
Telex: 650-116-9625
Telephone: (405) 360-7462

Robert Braver is the author of "Guest Commentary: The High Cost Of Personal Telecommunications" (COMPUTE!, November 1983). His commentary dealt with the then-unresolved question of whether Southwestern Bell, and other telephone companies, should charge home computerists a higher rate (tariff) to connect modems to their residential telephone lines.

Automatic LOAD And RUN For Commodore

I own a Commodore 64 and would like to know if, after LOADing a program from tape or disk, there is a way to make it automatically RUN without having to actually type in the RUN command?

Jeffrey Stevens

On Commodore computers, you can LOAD and automatically RUN a program from disk or tape. For tape LOADs, simply hit the RUN/STOP key while the SHIFT key is depressed. The first program that is found on the tape will LOAD and RUN. If you wish to LOAD and RUN a program other than the first one, type in LOAD "Program Name",1: and then press the RUN/STOP key with the SHIFT key down. In this case, the program designated by Program Name will LOAD and automatically RUN.

A similar procedure can be followed if you are using

a disk drive. Type LOAD "",8: and then press SHIFT RUN/STOP to LOAD and RUN the last program accessed (if no program resides in memory, the first program listed in the directory will be LOADed). For programs other than the first one on the disk, type LOAD "Program Name",8: and press SHIFT RUN/STOP.*

TI Cartridge Loading Problem

In answer to Charles Smith's inquiry (COMPUTE!, January 1984) regarding difficulty in repeated loading of TI-99/4A cartridges, several of us in a user group have had the same problem. It is the result of TI's use of a right-angle adapter for the cartridge socket.

The horizontal socket is mounted 90 degrees to a vertical connector board, which then plugs downward into the motherboard. Continued horizontal motion eventually loosens the pin tension in the motherboard socket. Unfortunately, the solution requires disassembly of the computer to gain access to the motherboard socket.

Charles Smith will have to decide whether he wants to tackle this on his own (possibly voiding his warranty), or refer the job to an authorized service center.

Donald C. McMahon

Long Commodore Tape Saves

With long programs it is often not possible to finish typing in a listing in one sitting. I have been saving part of the program on tape and when I continue the second part, I erase the first part on the tape and rerecord the whole thing.

I would like to just continue recording with the next line number of the program. Please advise if there is a way.

J. R. Blundin

When you SAVE to tape, the computer first writes a header which contains information such as program name and starting address. Then the program is saved twice, and an end-of-file marker is written when the tape file is closed. (Commodore computers save programs twice as a safety measure.)

These SAVE procedures make it impossible to position the tape and append the second half of the program onto the first. Attempting this would probably result in a DATA error, and both halves might be lost.

During long typing sessions, you should SAVE your incomplete program occasionally (every hour or so). This way, if some sort of catastrophe happens, you won't lose the entire program. You can load the last version you saved and continue typing.

Also, it's not a good idea to continually rewind the tape and keep saving programs over the same section of the tape. Saving programs consecutively will afford another safety measure—you'll always have backup

PROGRAMMING THE TI

C. Regena

Part 2: File Processing

I have had *lots* of requests for programs that use TI Extended BASIC and peripherals (printers, the RS-232 interface, and disk drives). In these next two columns I will try to satisfy those readers, plus the readers who have requested file processing, by discussing a report writer that uses disk files and a printer.

Keep in mind that there are many ways to program (as many ways as there are programmers). I'm going to show you one method I use to set up a file, then generate several reports from that file. For an example, I'm going to use a hypothetical situation—a teacher has divided the class into three reading groups. The students are given the goal of presenting one oral book report each week of the term (for this example the term is ten weeks). The teacher will grade the students on how well the goal is attained. These reports keep a tally of the book presentations. As part of the necessary school paperwork, the teacher also needs to keep track of each student's address and phone number. As a personal touch, the teacher also wants to be able to acknowledge the students on their birthdays. We'll get to these last two programs next month.

Writing A Data File

First, a data file is set up using Program 1. The student information is listed in DATA statements within this program, and the program will generate a data file on diskette. You could use DATA statements within a regular program which also writes the report, but with many students a shortage of computer memory could be a problem Also, I'm going to use this data for several programs, and I'll only have to type it once—in this data file generator program. All reports will use this one data file.

Program 1 READs the information in from DATA statements, then writes it directly out on a disk file. Line 130 is the key statement to set up the disk system for storing the data. The OPEN statement says device #3 (you can use any number) is disk drive number one, and the file of data will be called SAMPLE. The data will be stored in internal format as output and can have a variable length of 192.

To use this program with cassette instead of diskette, use this line:

```
130 OPEN #3:"CS1",INTERNAL,OUTPUT,F
    IXED 192
```

In later programs change any DSK1 and titles to CS1 within quotes, and change the VARIABLE to FIXED since cassettes can handle only FIXED-length files.

Reading The Data

Line 140 reads the data in the following order for our situation: group G, last name N$, first name F$, address A$, phone number P$, birthday expressed as a number BD, report R$, and comments C$. Line 150 prints the same information onto the diskette. Lines 160–180 stop the process if the name read is "ZZZ", which indicates the end of the file. Line 170 counts the names.

Lines 190–490 contain the data. Notice that the last DATA statement contains "ZZZ" for N$, the last name. These names are sample names only and are not meant to represent any real people. The other information is also made up for purposes of illustration. The DATA statements are in order by last name.

The first number in the DATA statement is a group number—the sample class is divided into three reading groups. We're assuming the students all live in the same city, so only the street address is stored as A$. The phone number P$ is stored as a four-digit number because they all have the same prefix. You can change this if you wish. Since the variable name is P$, you may include a hyphen in the phone number. The birthday BD is expressed as a number which consists of the month number then two digits representing the day number. For example, September 24 is month 9 and day 24 for 924. November 25 is 1125. October 5 is 1005 for month 10 and day 05.

Encode The Special Cases

The next series of numbers represents whether the student presented a book report or not. I combined all the weeks into one ten-digit number. A 1 means the student gave a report that week and 0 means he or she did not. A dash means the student was not enrolled. You may have a report that uses other symbols for other purposes an

asterisk for a different assignment, for example. The comments C$ are just to illustrate more versatility in later reports. I used AUDIT to represent a student who will not be counted for credit, and MOVED indicates the student is no longer in the class.

To try the sample programs this month, start with a new initialized diskette. Type in Program 1 then save it with a command such as SAVE DSK1.LIST1 or SAVE DSK1.NAMES or whatever label you wish. Next RUN this program with the diskette in the disk drive. The drive light will blink on and off as data is being recorded. The screen shows how many names have been processed. This program only sets up the data files—it does not write any reports.

Generating A Roster

Now go on to Program 2. This program will read information from the data file we created previously with Program 1 and print out a roster of information—the student, which group the student is in, the address, the phone, the birthdate, and any comments.

Lines 130–150 contain a procedure that reads the month names, which are later used in printing the birthdays. Line 160 initializes some variables. Note that Extended BASIC allows several variables at once to be set equal to zero. The variable I is used to count the total number of names. N is the number of auditing students, and MM is the number of students who have moved. L is a line counter.

Printer Features

My printer has a feature that will skip over perforations at the end of a page. In case your printer does not automatically do that, I have included this method. L is used to count the lines that have been printed. Line 360 can then check to see if it is time to change pages. PRINTing CHR$(12) goes to the top of a new page. If your printer cannot PRINT CHR$(12), you can put in a pause so you can reset to the top of a new page. (The sample data included here does not go more than one page. Add more names if you want to see this part work.)

Line 180 defines the printer configuration for device #2 (use any device number). Use the parameters you need for your particular printer. Consult the RS-232 manual and your printer manual to figure out your configuration. Line 190 is the OPEN statement for device #3 (any number) to read in the data. Notice that this statement matches the OPEN #3 statement in Program 1, except we use INPUT instead of OUTPUT because we will be reading in data.

Lines 200–240 print the heading. Line 250 is an INPUT #3 statement that tells the computer to

READ data from device #3. This statement is just like using READ and DATA statements within the program, only we use data files instead of program statements to store the data. The items listed may be read in as needed or all on one line, but must be in the same order as we previously saved them.

Ordering The Data

Line 260 skips the name on the roster if the student has moved and counts the number of people who have moved. Line 270 skips a line when the name starts with a different letter of the alphabet. By listing the original data in alphabetical order, this roster and other reports will automatically be in alphabetical order. A blank line is printed between groups of names starting with different letters. Line 280 checks for the last data item.

Line 290 combines the last and first names. Lines 300–310 determine the birthday from the number BD. Line 330 combines the common prefix 586- with the data P$ for the phone number.

Lines 320 and 410 are IMAGE statements, a feature of Extended BASIC which makes the module worth its price if you do lots of reports. An IMAGE statement allows you to specify how a line will be printed.

You can also use IMAGE statements PRINT USING statements to line up columns of money. For example, $###.## will print a number in dollar format with the cents rounded off. Line 340 uses line 320 to print the information.

Lines 350 and 360 increment the number of names and number of lines printed. Line 370 checks for a student who is just auditing the class. Line 380 causes the computer to branch back to the INPUT statement to read the next data items.

Typing Errors

If you get a data error, the most likely cause would be mistyping the DATA statements in Program 1. When the program stops with an error, you can PRINT N$ to see which was the last name that was accepted. Use that information to try to pinpoint a typing error in Program 1. (Be sure to SAVE Program 2 before you go back to Program 1.) You also need to make sure the data items in line 250 of Program 2 are in the same order as the data items in line 150 of Program 1. You may use different variable names if you wish, as long as the strings and numeric variables are in the right order.

Lines 390–460 print the totals information. Line 470 uses CHR$(12) to tell the printer to go to a new page. Line 480 CLOSEs both devices used in the program.

Program 1: Write Data Files
```
80 REM TI EXTENDED BASIC
90 REM DISK, PRINTER
```

```
100 REM WRITE DATA FILES
110 CALL CLEAR
120 DISPLAY AT(12,5):"WRITING DATA
    FILES"
130 OPEN #3:"DSK1.SAMPLE",INTERNAL,
    OUTPUT,VARIABLE 192
140 READ G,N$,F$,A$,P$,BD,R$,C$
150 PRINT #3:G.N$.F$.A$.P$.BD.R$.C$
160 IF N$="ZZZ" THEN 180
170 I=I+1 :: DISPLAY AT(23,3):I ::
    GOTO 140
180 CLOSE #3 :: STOP
190 DATA 1,ADAMS,JENNIFER,1207 W 80
    0 S,2314,924,1001110111,""
200 DATA 2,ALLEN,MICHAEL,204 N 300
    W,3273,1125,0010001101,""
210 DATA 3,ANDERSON,PAM,112 S 350 E
    ,1122,129,1101101101,""
220 DATA 2,BAKER,MICHELLE,310 S 350
    E,4054,1005,1101110111,""
230 DATA 2,BROWN,JUSTIN,971 EVERGRE
    EN,9656,212,1111111111,""
240 DATA 1,CARTER,JODI,918 JUNIPER,
    8803,502,0110011111,""
250 DATA 3,CHRISTENSEN,WES,1804 S 8
    00 W,3102,726,1110111011,""
260 DATA 1,DAINES,BRETT,123 S 350 E
    ,4765,1020,1001110111,""
270 DATA 1,GIFF,BEAU,222 CABBAGE LA
    NE,5733,901,--11011111,""
280 DATA 2,HANSEN,GRANT,209 N 300 W
    ,8996,425,111100111-,MOVED
290 DATA 3,JENSEN,CINDY,932 EVERGRE
    EN,7532,415,1011101101,""
300 DATA 2,JENSEN,NATALIE,420 S 100
    E,3487,512,,""
310 DATA 1,JOHNSON,ROCHELLE,355 S 1
    00 E,0217,110,1101111100,""
320 DATA 1,JONES,CHERY,1502 CEDAR,0
    157,802,1111111111,""
330 DATA 3,LARSEN,BOB,120 S 350 E,8
    674,510,---1111111,""
340 DATA 1,LARSEN,BILL,56 S 300 E,1
    299,520,1111111111,""
350 DATA 3,MORRIS,TERRY,375 E 100 N
    ,5607,114,1111011110,AUDIT
360 DATA 2,NELSON,ROGER,1362 N 1700
    E,4355,1005,11001111--,MOVED
370 DATA 1,NIELSEN,TOM,236 E 500 N,
    5670,806,0111111011,""
380 DATA 2,OLSON,RICHARD,801 W 300
    S,4587,0,1100001111,""
390 DATA 3,PETERSON,RANDY,233 E 500
    N,9007,302,1111111111,""
400 DATA 2,RICH,BRITTANY,725 E 525
    S,5683,407,1110110111,AUDIT
410 DATA 2,SAWYERS,JILL,806 W 200 N
    ,,611,1010101011,""
420 DATA 3,SMITH,JEFF,305 S 350 E,7
    789,315,1110111110,""
430 DATA 2,SMITH,TROY,855 S 300 E,6
    834,722,111--11111,""
440 DATA 3,SORENSON,GREG,315 E 525
    S,4379,104,1111111111,""
450 DATA 2,TAYLOR,CAMILLE,225 E 525
    S,3225,826,1111001101,""
460 DATA 1,THATCHER,DON,534 SPRUCE,
    2758,627,1110011101,""
470 DATA 1,WHITE,DEANNA,289 CEDAR,8
    453,121,1110100100,""
480 DATA 3,WILSON,ED,342 S 300 E,21
    55,0,1001111011,""
490 DATA 5,ZZZ,ZZZ,,,0,,""
```

Program 2: Roster

```
80 REM TI EXTENDED BASIC
90 REM DISK, PRINTER
100 REM ROSTER
110 CALL CLEAR
120 DISPLAY AT(12,5):"PRINTING ROST
    ER"
130 DIM M$(12)
140 FOR I=1 TO 12 :: READ M$(I):: N
    EXT I
150 DATA JAN,FEB,MAR,APR,MAY,JUN,JU
    L,AUG,SEP,OCT,NOV,DEC
160 I,N,MM,L=0 :: L$="A"
170 REM PRINTER CONFIGURATION
180 OPEN #2:"RS232.BA=600"
190 OPEN #3:"DSK1.SAMPLE",INTERNAL,
    INPUT ,VARIABLE 192
200 PRINT #2:TAB(34);"SAMPLE CLASS"
210 PRINT #2: :TAB(37);"ROSTER"
220 PRINT #2: :TAB(33);"APRIL 15, 1
    984"
230 PRINT #2: : :"(3 SPACES)NAME";T
    AB(25);"PHONE";TAB(34);"ADDRESS
    ";TAB(61);"BIRTHDAY";TAB(71);"C
    OMMENTS"
240 PRINT #2:"-    ----";TAB(25);"---
    --";TAB(34);"-------";TAB(61);"
    ----------";TAB(71);"--------"; :
250 INPUT #3:G,N$,F$,A$,P$,BD,R$,C$
260 IF C$="MOVED" THEN MM=MM+1 :: G
    OTO 250
270 IF SEG$(N$,1,1)<>L$ THEN PRINT
    #2 :: L$=SEG$(N$,1,1):: L=L+1
280 IF N$="ZZZ" THEN 390
290 N$=N$&", "&F$
300 IF BD=0 THEN BD$="---" :: D=0 :
    : GOTO 330
310 BD$=STR$(BD):: D=VAL(SEG$(BD$,L
    EN(BD$)-1,2)):: M=VAL(SEG$(BD$,
    1,LEN(BD$)-2)):: BD$=M$(M)
320 IMAGE # ############### ##
    ###### ########################
    ##(3 SPACES)### ##(3 SPACES)###
    #######
330 P$="596-"&P$
340 PRINT #2,USING 320:G,N$,P$,A$,B
    D$,D,C$
350 I=I+1
360 L=L+1 :: IF L=49 THEN PRINT #1:
    CHR$(12):: L=0 :: GOTO 200
370 IF SEG$(C$,1,5)="AUDIT" THEN N=
    N+1
380 GOTO 250
390 PRINT #2: : :
400 T=I-N
410 IMAGE ### ###################
420 PRINT #2,USING 410:I,"NAMES"
430 PRINT #2
440 PRINT #2,USING 410:N,"AUDIT"
450 PRINT #2,USING 410:MM,"MOVED"
460 PRINT #2,USING 410:T,"PRESENTLY
    ENROLLED"
470 PRINT #2:CHR$(12)
480 CLOSE #2 :: CLOSE #3
490 END
```

TI Tricks And Tips

Michael A. Covington

Here are 13 ways to get more out of your TI-99/4A and Extended BASIC.

Here is a collection of handy hints for TI-99/4A programmers.

1. You can get white characters on a black screen, for the duration of a program run, by executing statements such as:

```
10 FOR J = 1 TO 14
20 CALL COLOR(J,16,1)
30 NEXT I
40 CALL SCREEN(2)
```

The results look rather good on a color TV, but bad on a black-and-white set, because scan lines break each letter up into separate dots. (The most readable black-and-white display is obtained by executing a CALL SCREEN(15), making the screen gray while leaving the characters black.)

2. In both TI BASIC and Extended BASIC, you can use * for the logical operator AND and + for OR. For instance, the statement IF (X = 0) + (Y = 0) THEN 1500 means "if X = 0 or Y = 0 then go to line 1500." The parentheses are essential to show that you don't want to add 0 to Y. Extended BASIC allows you to use the alternative notation IF X = 0 OR Y = 0 THEN 1500.

3. In Extended BASIC, but not in TI BASIC, pressing any key while a program is being LISTed temporarily halts the listing; pressing any key then causes the listing to resume. In each case the key must be held down for half a second or so in order to get any response.

4. If you RESEQUENCE a program that contains references to nonexistent lines, those references will be changed to references to line 32767. For instance, if you have a GO TO 500 and there isn't a line 500, a RES command will change that statement to GO TO 32767.

5. In Extended BASIC, the command RUN "CS1" loads a program from the cassette drive and immediately runs it. It is equivalent to OLD CS1 followed by RUN. With a disk drive and Extended BASIC, you can use RUN "DSK1.filename", where *filename* is the name of the program on disk that you want to LOAD and RUN.

6. TI BASIC gives you 608 more bytes of available memory than Extended BASIC. However, you can usually write your program more compactly in Extended BASIC, so the difference is of little practical consequence.

7. Built-in subprograms that require integer arguments, such as CALL HCHAR, CALL VCHAR, CALL SOUND, and the CHR$ function, will in fact accept numbers that are not integers. The argument is rounded to the nearest integer before being used, so that for instance CHR$(10.8) is the same as CHR$(11). CHR$(10.4) would be equivalent to CHR$(10).

8. In TI BASIC, you can include multiple colons (for example, :::::) in PRINT statements to produce multiple line skips. A TI BASIC program using this feature which is loaded from disk or cassette under Extended BASIC will run correctly, but you cannot type multiple colons while in Extended BASIC unless you want them to be taken as statement separators (::). Put spaces between the colons, as in PRINT A : : B rather than PRINT A :: B, and they will work correctly.

9. In Extended BASIC, you cannot have more than four sprites visible on the same line at the same time; additional sprites will be temporarily invisible. The problem is worse with double-size sprites (CALL MAGNIFY(3) or (4)), since then only part of the sprite generally disappears, distorting its appearance.

10. When you execute a CALL SPRITE statement, the sprite will sometimes momentarily pop

into existence at a random screen location and then jump to the location that you specified. To prevent this, create the sprite with a color of 1 (transparent) and then alter its color with a CALL COLOR statement.

11. The loss of resolution on the screen that occurs with certain color combinations is inherent in the way color is encoded onto the video signal and does not represent a defect in the TV set or modulator. For greatest sharpness, use black on gray or cyan.

12. The TI-99 sound generator will produce frequencies from 110 to 44733 hertz (cycles per second), well above the limit of human hearing. However, the response of the sound section of most TV sets falls off markedly above 2000 (or, at best, above 10,000) hertz. This means that you cannot, as is sometimes suggested, use the TI-99 to test the upper frequency limit of your hearing. It also means that tones above 2000 Hz—still well within the range of human hearing—will sound markedly different on different TV sets.

13. If you want to transmit lines of more than 80 characters to the printer, open it as "RS232.CR" (or "PIO.CR") rather than "RS232" (or "PIO"). You must still end your line of output (by executing a PRINT statement that does not end in a comma or semicolon) before more than 80 characters have been transmitted, but doing so will not cause the printer to start a new line; the printer will stay on the same line until you explicitly transmit a carriage return, CHR$(13), and a line feed, CHR$(10).

This is particularly useful when you are using a dot-matrix printer in graphics mode, using each character code for a single vertical row of dots and putting hundreds of them on a line. **©**

CAPUTE!

Calorie Cop For Atari And TI

In the Atari version of this activity planning program from the December 1983 issue (p. 52), the GOTO 470 in lines 540 and 552 should be changed to GOTO 475. Also, lines 250 and 420 can be deleted.

In the TI version (Program 3), line 360 can be deleted and the following lines should be added or changed:

```
445 PRINT : :
447 PRINT "(OR HIT ENTER FOR MORE)"
750 IF MIN=0 THEN 870
```

Atari Quatrainment

The characters which appear as ⌐ ⌐ in line 20020 of Program 1 (February 1984, p. 78) should be the vertical line character, obtained by pressing SHIFT and the = key.

The Beginner's Page: Program Forms

Line 560 from Program 1 of this February column (p. 102) should read:

560 DATA GALLONS,3.785,LITERS

VIC 3-D Drawing Master

Users of this program (February 1984, p. 146) will encounter a syntax error in line 1803 when attempting to load a picture file from tape. The line should read:

```
1803 IFLEFT$(IN$,1)="T"THENIN=1:GOTO1805
```

64 Sound Tester

There were typographical errors in the corrections to this program (November 1983, p. 187) which appeared in the February "CAPUTE!" page. The modified program lines should have read:

```
310 FOR I=1TO15STEP2: POKEW,X: POKEHF,SO(I
    ,A(2)): POKELF,SO(I+1,A(2))
311 O=O+1: FORN=1TOD(O): NEXT: POKEW,X-1:
    NEXT: FORI=1TO10000: NEXT
```

Hidden 64 Memory

If you use the techniques outlined in this article (January 1984, p. 172) to access the RAM under the Kernal ROM, you should be aware that at least one of the 64's Kernal routines writes to the RAM beneath it. Raymond Quiring notes that the Kernal routine RESTOR at address 65418 (SFF8A) writes to the RAM from 64816–64846 ($FD30–$FD4E). The RESTOR routine is called when the RESTORE key is pressed, or when a BRK instruction is encountered in machine language.

Making Cents

The February "Readers' Feedback" suggestion for a program line which formats numbers as dollars and cents (p. 10) produces improper values in those cases where the number is negative or zero. David Gamache suggests instead the following line, which works for both positive and negative numbers and zero:

```
V$=STR$(X+(.001*SGN(X))):V$=LEFT$(V$,LEN(
V$)-1):IFX=.THENV$="0.00"
```

Program Line Addresses For VIC And 64

The instructions given in January "Readers' Feedback" (p. 10) for appending this program to an existing BASIC program work only for the Commodore 64. For the unexpanded VIC, you should use POKE 44,16 instead of POKE 44,8 in step 4. The proper value for the 3K expanded VIC is POKE 44,4, and with 8K or more expansion you should use POKE 44,18.

VIC Modem Save And Download

There are still bugs in this program (November 1983, p. 215) after the corrections in the January "CAPUTE!" page. Load your version of the terminal program (Program 2 with the data from Program 1 added). Type POKE 45,49:POKE 46,20 and RETURN, then make the following changes and additions, suggested by Larry Flohaug:

```
130 PRINT"{CLR}{DOWN}1-SAVE TO TAPE":PRIN
    T"2-PRINTER":PRINT"3-TAPE & PRINTER"
140 PRINT:PRINT"WHICH?"
170 OPEN4,4:FORK-6656TOPEEK(0)*256+PEEK(1
    ):PRINT#4,CHR$(PEEK(K)AND127);:NEXT
175 PRINT#4:CLOSE4
200 GOTO225
```

After making the modifications, type POKE 45,1:POKE 46,26 and save a copy of the revised program. The test of the Program 1 data in the original article (p. 216) will insure that the data has not been affected by these changes.

Colorbot

When you lose your last man in the VIC or 64 version of this game from the January issue (p. 84), the screen clears immediately. Cliff Tener suggests adding the following line to either version to create a sufficient delay to check your final score:

352 FOR T = 1 TO 2000: NEXT

Atari XL Compatibility Problems

Based on mail we have received, we suspect that the "Polycopy" program from the November 1983 issue will not run properly on any of the new Atari XL computers. The following programs work on the 1200XL, but apparently not on the 600 or 800XL: "Chopperoids" (December 1983), "Demons Of Osiris" (January 1984), and "Circus" (February 1984). ©

NEWS&PRODUCTS

Color Computer Cassette-Based Games

Radio Shack has produced several cassette-based computer games for the TRS-80 Color Computer with Extended BASIC for children seven years of age and older. The three games require joysticks and encourage players to work together.

Peanut Butter Panic! is a two-person game in which players jump for stars to make peanut butter sandwiches. By working together, players are able to catch the most valuable stars.

In *Taxi*, players get behind the wheel of a cab. By maneuvering around the street grids based on city maps, players try to deliver as many passengers as possible before time runs out. When played cooperatively, the game encourages communication and division of labor.

Star Trap is a maze game in which players attempt to trap stars by blocking their paths. Two players working as a team are more effective than one player.

Each of these games is priced at $19.95.

Another game now available from Radio Shack for the Color Computer on tape is *ZAXXON*. The player becomes the pilot of a fighter spacecraft on a mission to meet and destroy the deadly *ZAXXON* robot. Points are scored by destroying a variety of threats from enemy planes, base missiles, firing gun emplacements, and radar towers.

The game sells for $34.95 and requires a joystick.

Tandy Corporation/Radio Shack
1800 One Tandy Center
Fort Worth, TX 76102
(817) 390-3300

Games For Apple II, Atari

Kangaroo, Inc., has released two new games, *Jeepers Creatures* and *My House My Home*, both for the Apple II and Atari.

Jeepers Creatures takes children through a zoo of 30 basic animals, with interchangeable heads, torsos, legs, and tails. More than 20,000 different combinations are possible for children to create.

My House My Home is an electronic playhouse. A moving conveyor belt with furniture, pets, and people can be controlled by either a joystick or the keyboard. The user can furnish a cutaway view of a house in a variety of different ways.

Jeepers Creatures and *My House My Home* are priced at $34.95 on disk.

Kangaroo, Inc.
322 South Michigan Avenue
Suite 700
Chicago, IL 60604
(312) 987-9050

Atari Tape Interfaces

RC Systems, Inc., has introduced two models of tape interfaces which allow a standard cassette recorder to be used for loading and saving programs with any of the Atari computers.

The models are the AA-2

and AA-1, the second of which operates with the Atari 400 and 800 computers. The AA-1 has all the features of the AA-2, but incorporates additional circuitry to duplicate and rejuvenate program tapes (a second recorder is required for this).

The assembled and tested AA-2 cassette interface board is priced at $27.95. In kit form, it is available for $19.95. The AA-1 is priced at $39.95; in kit form, $29.95.

RC Systems, Inc.
121 West Winesap Road
Bothell, WA 98012
(206) 771-6883

Filing, Mailing Systems For TI

TI File, TI Mail, and *TI File Junior* are three packages developed for the TI-99/4A computer system by Kinetic Designs.

TI File is a multipurpose filing system which allows the user to construct, sort, maintain, and print out a variety of files for home or business.

TI File requires Extended BASIC, 32K memory expansion, and cassette deck. Options include disk drive and printer. It is available on tape for $14.95 and on disk for $17.95. A $2 postage fee should be added.

TI Mail is a mailing list management system which also requires Extended BASIC, 32K memory expansion, and cassette deck. An optional heading allows the user to keep track of expiration dates, scores, or anything else related to those listed. *TI Mail* is available on cassette

for $10.95 or on disk for $13.95. A postage fee of $2 is additional.

TI File Junior is similar to the *TI File,* but requires only a standard 16K TI-99/4A and a cassette deck. It is available on cassette for $9.95, plus a $2 postage fee.

Kinetic Designs
P.O. Box 1585
Orange Park, FL 32067
(904) 264-6777

Desert Game For Apple

Sands of Egypt, a desert adventure game from Datasoft, has been introduced for the Apple II, IIe, and II+ computers.

The game incorporates action with riddles using a split screen image. It follows Lord Charles Buckingham III, a desert traveller, as the player attempts to find hidden treasure by answering a series of riddles and by dodging dangers and obstacles.

The game is priced at $29.95, and is also available for the Atari 400, 800, and 1200 computers.

Datasoft, Inc.
9421 Winnetka Avenue
Chatsworth, CA 91311
(213) 701-5161

Basketball Simulation Game

Electronic Arts has introduced *One-on-One,* a basketball simulation game that pits Boston Celtics star Larry Bird against Julius (Dr. J) Erving of the Philadelphia 76ers.

It is available on disk for the Apple II, II+, and IIe, and will be available this year for the Atari and Commodore 64 computers.

Statistical tables reflecting the players' strengths all over the court have been built into the game. The two players also advised the game creators on their individual strategies and

maneuvers. Among the features are a 24-second-shot clock, a game clock, and a scoreboard that shatters when a slam dunk is made. The game also has hot and cold shooting functions, instant replays, and visible measures of player fatigue.

One-on-One has a suggested retail price of $40.

Electronic Arts
2755 Campus Drive
San Mateo, CA 94403
(415) 571 7171

Commodore 64 Checkwriting, Accounts Package

COMP-U-CHECK is a checkwriting and accounts servicing package designed for personal or small business applications by Hot Data Software Development.

The package requires a Commodore 64 computer, disk drive, and a printer.

COMP-U-CHECK provides checkwriting, check account balancing, credit account management, letter writing, automatic bill paying, and form letters. Also included are tax time reports, constant balance and totals paid to date, new year accounts reset, and user support through a direct-help phone number.

The package is available on disk for $64.95.

Hot Data Software Development
1021 Lincoln Boulevard
Santa Monica, CA 90403
(213) 393-6405

Popular Games Released

Atari, Inc., has released 12 of its games for competing computers and videogame consoles in a new line of software called ATARISOFT.

The initial entries in the new line include *Centipede, Defender, Dig-Dug, Donkey Kong,*

Pac-Man, Robotron, Stargate, Picnic Paranoia, Protector, Shamus, Super Storm, and *Galaxian.*

The first seven games will run on the Apple II and IIe, IBM PC, Commodore 64, VIC-20, and Texas Instruments 99/4A computers. The remaining games, except *Galaxian,* will operate on the Texas Instruments machine. *Galaxian, Centipede,* and *Defender* will run on ColecoVision, while *Centipede, Defender,* and *Pac-Man* will operate on Intellivision.

The computer games carry suggested retail prices of $34.95 for disks and $44.95 for cartridges. Suggested retail for the ColecoVision and Intellivision versions is $40.95.

Atari Incorporated
1265 Borregas Avenue
P.O. Box 427
Sunnyvale, CA 94086
(408) 743-4810

War Strategy Games For Apple, Atari, 64

Strategic Simulations Inc. has produced several war strategy games for the Apple, Atari, and Commodore 64 computers.

Carrier Force is a simulation of the four major aircraft carrier battles fought in the Pacific during World War II—Coral Sea, Midway, Eastern Solomons, and Santa Cruz. Every major warship and plane is accounted for and rated in the game. Weather, time of day, visibility, inaccurate sightings, and other aspects of actual naval warfare conditions are recreated.

The game comes with two maps, rule book, and 48K disk for the Apple II with Applesoft ROM, II+, IIe, and III. It is also available on 40K disk for the Atari 400/800/1200. The price is $59.95.

Battle for Normandy is a simulation of the D-Day invasion and the 24 days that followed.

Now available for the Commodore 64 computer in a 64K disk and a cassette format, the game includes all historical details. The player may be the Supreme Allied Commander or commander of the German defenses. A solitaire option is available, with the computer directing the German defenses.

The game comes with two player-aid cards, maps and rule book. Versions are also available for Atari, TRS-80 models I and III, Apple, and IBM PC computers. *Battle for Normandy* is priced at $39.95.

Tigers in the Snow is a recreation of the World War II Battle of the Bulge, Germany's last major counteroffensive. The game is now available for the Commodore 64 computer.

In the simulation, the Germans attempt to destroy the Allies' stronghold while the outnumbered Allies try to hold until their reinforcements arrive. The forces are division/regiment scale. Both sides command infantry, artillery, and air power.

The price for the game is $39.95 on 64K disk for the Commodore 64. It is also available for the Apple, Atari, and TRS-80 computers.

Strategic Simulations Inc.
883 Stierlin Road, Building A-200
Mountain View, CA 94043
(415) 964-1353

Color Graphics Packages For 64

Sophisticated Software of America has released *Grafix-Artist,* a high-resolution color-graphics package for the Commodore 64, and *Grafix-Printer,* a high-resolution graphics printer-dump for use with printers/interfaces which emulate the Commodore 1525E printer.

Grafix-Artist was created to introduce children and adults to computer graphics and to pro-

gramming by allowing them to use either a specially designed language or a joystick to control the onscreen graphics capabilities. The product can be used to create detailed screens which can then be included with other programs written in either BASIC or machine language. Design layouts, storyboarding, and ad displays are among the uses of *Grafix-Artist.*

Available on disk, *Grafix-Artist* has a price of $39.95. *Grafix-Printer* is available for $29.95.

Another Sophisticated Software product, *Lesson Designer,* which allows free-form lesson design, will be available this spring.

Sophisticated Software of America
198 Ross Road
King of Prussia, PA 19406
(215) 265-2277

Educational Games On Disk

Unicorn Software has produced four educational games available on disk for the Atari, Commodore 64, IBM PC, and Apple computers.

Ten Little Robots is composed of five learning games that introduce children, from ages two to seven, to the computer. They include Little Robot Story, an interactive nursery tale that presents the concept of subtraction and aids in reading preparation; Robot Letter Match, which teaches upper- and lowercase letters; Count The Robots; Robot Addition; and Robot Sketch.

Ships Ahoy is a series of four games for children from 5 through 13 years of age. These four games include Ships Ahoy and Mine Sweeper, two math skills programs; Treasure Hunt, a maze game included as a reward; and Sailing Sketch, a screen painting program.

Race Car 'Rithmetic, for those five years old to adult, is a math game designed for the entire