SEGA® COMPUTER

September/October Issue 1985

The Official Sega User Club Magazine

PROGRAMS BY

Michael Howard
Adam Tommy
Ian Nicholson
D Van Kan

FEATURES

Disc Bonanza
Death of Philip Lloyd
Hypo-Qix
Ecological Modelling
Artificial Intelligence
And more . . .

INTRODUCTION

Dear Readers

It is encouraging to see that our fame is spreading across the Tasman and I welcome all you Aussie Sega owners to our club.

Indeed we have heard of several fights breaking out amongst local area user club members trying to get their hands on a copy of this magazine at their meetings. Steps are being taken to make this publication more available over there, so don't despair.

In this month's issue we have some quality programs for you to type in. The Death of Philip Lloyd being a cluedo type game, and Jet Ranger for all you flight simulation fans. There are also a few useful programs that have been converted to run on SF-7000 disc crive systems. All in all a very informative issue.

Happy Programing!

S Kenyon
DIVISIONAL MANAGER

Contents

OVERSEAS SEGA USER's CLUBS

Camden Sega User's Club
Contact Steve MacDonald
2 Coolalie Ave
Camden 2570
Australia

LOCAL SEGA USERS CONTACTS

AUCKLAND CENTRAL SEGA USERS CLUB
C/o 287 Broadway Furniture
Newmarket
Contact: George Shaw
Ph. 547-543

B.O.P SEGA USERS CLUB
Sega 102B Hinewa Road
TAURANGA
Sec P.J. McDonald
Ph 64826 a/h
Tauranga

CHRISTCHURCH USER'S CLUB
Contact Graham Rudman
29 Primrose Street
CHRISTCHURCH 5

GISBORNE AREA USER'S CLUB
Trevor Gardiner
Ph. 83-068 HM
or 87-175 WK

HAMILTON SEGA USER'S CLUB
P.O. Box 1548 Hamilton
President: Mr Colin Bell 73-826
Secretary: Mrs Anne Thrush Ph 437-312
Meetings held fortnightly on Monday at Waikato Technical Institute in Room F129.

HASTINGS AREA USER'S CLUB
Contact: Robin
8 Liverpool Cresent
Flaxmere
Hastings
Ph 798-116

NAPIER SEGA USERS CLUB
Sec E.P. Lins
41 Higgins Street
NAPIER

PUKEKOHE SEGA USERS CLUB
C/o 4 Roose Avenue
Pukekohe
Contact: Selwyn
Ph. Pukekohe 86-583

ROTORUA SEGA USERS CLUB
C/- 61 Devon Street
ROTORUA

TOKOROA SEGA USERS GROUP
C/o 1 Pio Pio Place
TOKOROA
Contact Geoff Phone Number 67-105
Tokoroa

SOUTH CANTERBURY USERS CLUB
P.O. Box 73 Timaru New Zealand
President: Geoff McCayghan
Secretary: Lloyd Van Der Krogt
Contact Phs: 60-756, 61-412 TIMARU

SOUTH COROMANDEL USERS CLUB
P.O. Box 183
WHANGAMATA
Contact: Sid
Ph. 58-775 Whangamata

SOUTH TARANAKI MICROCOMPUTER SOCIETY
D.M. Beale
7A Clive Street
HAWERA

WELLINGTON USERS CLUB
Contact: Shaun Parsons
P.O. Box 1871
Postal Central
Wellington
Ph 897-095 a/h
Ph 727-666 work

The above are contact names and addresses for Sega Users Clubs. If you wish to have your club advertised write to Sega Users Club P.O. Box 2353, Auckland.

If you have set up a local area user's club and you would like us to publish the details concerning your club please send them to us and we will publish the information for no charge.

READER'S LETTERS

DEAR EDITOR

I have been recently doing a little programming with the use of sprites, and have come up against a brick wall!! My probelm is this . . . how many sprites can you have on the screen at once?

Grant Philips
Auckland

EDITORS REPLY

The Sega allows a very powerful method of sprite use . . . it is a lot more powerful in fact than most other computers, here is how it works. . . You can have 32 sprites on the screen at any given time, the sprites on the screen can be chosen from a "bucket" of 256 sprites. The result of this is that you may define all 256 sprites and have 32 out of that 256 on the screen at once . . . here is a demo program . . . it will show you that there are indeed 256 sprites.

```
10 SCREEN 2,2:CLS
20 FOR A=0 TO 255
30 A$=HEX$(RND(8)*255)
40 FOR B=0 TO 2:A$=A$+A$:NEXT
50 BLINE (20,40)-(45,50),,BF
60 PATTERN S#A,A$
70 SPRITE 0,(20,20),A,1
80 CURSOR 20,40:PRINT A
90 NEXT
```

DEAR EDITOR

I am having a little trouble with the sound on my brilliant Sega. When I make some "white noise" by using channel 5, and then use "tone" by using channel 4, the white noise disappears . . . could you please help me?

PS: the last magazine was simply superb! Keep up the programs.

Philip Tottle
Titirangi, Auckland

EDITORS REPLY

When creating sound, you can use sound channels, 1, 2 and 3 and 4 or 5, in other words 1,2 and 3 can be used as often as you like, but you can use these in conjunction with channels 4 OR 5 . . . not both.

DEAR EDITOR

Firstly, I would like to say what a great change has happened to this magazine, it is now really professional and very helpful . . . thanks. The RS-232 section was very enlightening. Now to my problem. I am the owner of an SF-7000 Disc Unit and would like to know if it is possible to disable the RESET key, thus preventing younger ones from pressing it by accident and upsetting demo programs.

Michael Bluenun
Mosgiel

EDITORS REPLY

To disable the RESET on the SF-7000 unit, enter POKE &H66,&C9 at the start of your program. Note this will not work on the non Disc unit.

DEAR EDITOR

I would like to know if there is the possibility of a Machine Code book coming out for the Sega, as I've got BASIC all worked out and would now like to venture into machine code to write some games.

Vicky Absword
Tauranga

EDITORS REPLY

Just about every person out there has been wanting to know about a book on machine code and the answer in short is YES . . .very soon in fact!! Michael Howard, is at this very moment in time, working on one . . . so watch this space!

CONTACT PAGE

Steve Monk
4 Browne St
Kawerau

Andrew Stoddart
45 Belford St
Waverly
Dunedin
Ph 43-337

Darryl Gore
36 Browning St
Cambridge
Ph 8556

D.G. Stembridge
44 Mossburn Grove
Kelson
Lower Hutt

Hillary and Geoff McLeod
34 Price St
Invercargill
Ph 57859

Andrew Mckenzie
c/o Colyton School
RD 5 Feilding Rd
Colyton

Brian Kelly
Robert Road
Otatara No 9 R.D.
Invercargill
ph. 331-473

Neil Bryce
131 North Street
Timaru
ph. 88-434

Fredrick Curd
64 College St
Palmerston North

The Death of Philip Lloyd

This program is one of those "Cluedo" style games where you play the lead detective and must find the murderer of Philip Lloyd.

The story goes something like this . . . Philip decided to throw a huge party after getting a job . . . I mean he was sick to death of just playing backgammon and table tennis, so he invited a few of his mates over for a few drinks. The party was going really well when all of a sudden a loud scream was heard . . . someone had found the body of Philip . . . not very alive, dead in fact!!

This is where you come in . . . you are a lonely detective, looking for a case to answer . . . when all of a sudden the phone rings . . .

I'm not telling you what happens next . . . but the program is like a book . . . in that there are a few chapters to it.

Okay, here is how to play it. Run the program, and after a short pause you will see part of a story appear on the screen, you will then hear a beep . . . whenever you hear a beep, that means the computer has paused to allow you to see the story on the screen . . . when you have read the stuff on the screen just press a key.

When you arrive at the house you can ask all the suspects a maximum of two out of three questions . . . the questions are . . .

1) where were you at the time of the murder?
2) how many people were with you at the time of the murder?
3) what time did you hear of the death?

You must now remember that the murderer will lie, so watch out for inconsistences in the replies from suspects

The program is fairly straight-forward to use as it is menu driven, but the program plot is fairly complex, and don't forget there is a new story each time you play the game. Happy sleuthing!!!

```
10  CLS
20  DIMA(8,8)
30  PRINT"Chapter 1"
40  R=3:GOSUB1700
50  X=Z:GOSUB1710
60  PRINT"It is ";A$;" as you enter the"
70  R=3:GOSUB1700
80  X=Z+3:GOSUB1710
90  PRINT"squad room tonight.","The ";A$
100 R=3:GOSUB1700
110 X=Z+6:GOSUB1710
120 PRINT"and the ";A$;" had blown a fus
e"
130 R=3:GOSUB1700
140 X=Z+9:GOSUB1710
150 PRINT"- ";A$;"-","you moan,"How did I"
160 PRINT"lumber myself with night duty.
". But worse was yet to come, you could
feel it in your "
170 R=3:GOSUB1700
180 X=Z+12:GOSUB1710
190 PRINTA$
200 PRINT"you settle at your desk & idly
 thumb"
210 R=3:GOSUB1700
220 X=Z+15:GOSUB1710
```

```
230 PRINT"through a copy of ";A$:PRINT"Y
our thoughts turn to the boss ,"I bethe"
s having"
240 R=3:GOSUB1700
250 x=Z+18:GOSUB1710
260 PRINTA$;" . .lucky thing.!" "
270 R=3:GOSUB1700
280 x=Z+21:GOSUB1710
290 PRINT"You pull out a ";A$;" & fumble
 for";:PRINT"your matches when ";
300 GOSUB2120
310 PRINT"the phone rings."
320 GOSUB2120
330 PRINT"'Hello',crackles the line.Is t
hat the police?'."
340 R=3:GOSUB1700
350 x=Z+24:GOSUB1710
360 PRINTA$;",You reply";:PRINT"'There's
been a murder at the manor, come quickl
y'."
370 PRINT"Your migrane starts to come on
 but before you can say anything,the
line goes dead."
380 GOSUB2120
390 CLS:PRINT"Chapter 2"
400 PRINT"You pull your weary bag of bon
es out of the chair & take your "
410 R=3:GOSUB1700
420 x=Z+27:GOSUB1710
430 PRINTA$;" out of the drawer"
440 PRINT"The lift is out of order,so yo
u take";:PRINT"the stairs down to the car
."
450 R=3:GOSUB1700
460 x=Z+30:GOSUB1710
470 PRINT"'This ain't gonna do my ";A$:P
RINT"much good!',you mutter."
480 R=3:GOSUB1700
490 x=Z+33:GOSUB1710
500 PRINT"You climb into your ";A$
510 PRINT"....you put your foot hard dow
n on thegas.As you pull away down the ro
ad,yourreflect on the standard questions
thatsuspects in a murder case are asked.
"
520 PRINT
530 PRINT"Where were you at the time of
murder?"
540 PRINT"How many others were with you?
"
550 PRINT"What time did you here of the
murder?"
560 PRINT
570 PRINT"You realise that time is short
 & you will only be able to ask each su
spect two questions"
580 PRINT"But you know the murderer will
 lie"
590 GOSUB2120
600 CLS:PRINT"Chapter 3"
610 PRINT"You pull up outside the manor
& the butler takes you to the scene of
 the murder"
620 PRINT"All the suspects are waiting,s
even of them...You take their names."
630 RESTORE1870:FORA=0TO6:READA$:PRINTA;
")"A$:NEXT
640 GOSUB2120
650 PRINT"Your next move is to make a qu
ick sketch of the manor in your note
book"
```

```
660  R=4:GOSUB1700
670  D=Z
680  ONZGOTO690,700,710,720
690  K$="X":R$="Kitchen":GOTO730
700  D$="X":R$="Dining room":GOTO730
710  L$="X":R$="Living room":GOTO730
720  S$="X":R$="Study"
730  PRINT:PRINT"-------------------------------------------------
-----"
740  PRINT"I                              I                    I"
750  PRINT"I        Study                 I  Living room       I"
760  PRINT"I";TAB(6);S$;TAB(13);"I";TAB(2
0);L$;TAB(27);"I"
770  PRINT"I--------------------          I--------------------I"
780  PRINT"I                              I                    I"
790  PRINT"I  Dining room                 I      Kitchen       I"
800  PRINT"I";TAB(6);D$;TAB(13);"I";TAB(2
0);K$;TAB(27);"I"
810  PRINT"--------------------------------------------------"
820  PRINT"All you've to do now is find t
he          culprit."
830  GOSUB2120
840  FORI=1TO8:FORJ=1TO8:A(I,J)=0:NEXTJ,I
850  R=7:GOSUB1700
860  M=Z
870  FORS=1TO7
880   IFS=MTHENN940
890  R=4
900  GOSUB1700
910   IFZ=DTHENN900
920  A(S,Z)=1
930  A(S,5)=2
940  NEXTS
950  FORJ=1TO4:A(8,J)=0:FORI=1TO7:A(8,J)=
A(8,J)+A(I,J):NEXTI,J
960  FORS=1TO7
970   IFS=MTHENN990
980  A(S,6)=A(8,A(S,5))-1
990  NEXTS
1000  FORK=1TO4
1010   IFK=DTHENN1090
1020  R=3:GOSUB1700
1030  FORS=1TO7
1040   IFS=MTHENN1070
1050   IFA(S,5)<>KTHENN1080
1060  A(S,7)=Z
1070  NEXTS
1080  NEXTK
1090  R=4:GOSUB1700
1100   IFA(8,Z)<2THENN1090
1110  A(M,5)=Z
1120  R=2:GOSUB1700
1130   IFZ=1THENN1220
1140  A(M,6)=A(8,A(M,5))
1150  S=1
1160   IFS=MTHENN1180
1170   IFA(S,5)=A(M,5)THENN1200
1180  S=S+1
1190  GOTO1150
1200  A(M,6)=A(S,7)
1210  GOTO1290
1220  A(M,6)=A(8,A(M,5))-1
1230  S=1
1240   IFS=MTHENN1260
1250   IFA(S,5)<>A(M,5)THENN1280
1260  S=S+1
1270  GOTO1240
1280  A(M,7)=A(S,7)
1290  CLS:PRINT"Chapter  4"
1300  PRINT"You are in the  ";R$
```

```
1310  PRINT"ready to question the suspects"
1320  L=1
1330  X=40+L
1340  GOSUB1710
1350  PRINTA$;"----------":PRINT"enters the room & sits down"
1360  R=3:GOSUB1700
1370  X=58+Z+((L-1)*3):GOSUB1710
1380  PRINTA$
1390  T=0:FORG=1TO2
1400  INPUT"You ask question #";Q
1410  IFQ<1ORQ>3THEN1400
1420  IFQ=TTHEN1510
1430  T=Q
1440  ONQGOTO1450,1490,1500
1450  X=A(L,5)+36
1460  GOSUB1710
1470  PRINT"'I was in the ";A$;"'"
1480  GOTO1520
1490  X=A(L,6)+48:GOSUB1710:PRINT"'Beside me was ";A$;"'":GOTO1520
1500  X=A(L,7)+55:GOSUB1710:PRINT"'The time was ";A$;"'":GOTO1520
1510  PRINT"'You've already asked me that'":GOTO1400
1520  NEXTG
1530  IFL=7THEN1590
1540  PRINT:PRINT"You ask yourself,'Can I name The murd-erer?'  (Y/N)"
1550  B$=INKEY$:IFB$="Y"THEN1590
1560  IFB$<>"N"THEN1550
1570  L=L+1:IFL=8THEN1590
1580  PRINT:GOTO1330
1590  PRINT"Chapter 5"
1600  PRINT"All the suspects are present and you accuse-------"
1610  INPUTB$
1620  X=40+M:GOSUB1710
1630  IFLEFT$(B$,3)=LEFT$(A$,3)THEN1660
1640  PRINT"You are about to make the arrest,when":PRINTA$;" breaks down & confesses"
1650  PRINT"Your deduction is wrong & you arrest the real murderer":GOTO1690
1660  PRINT"You arrest the murderer & reflect that"
1670  X=48+L:GOSUB1710
1680  PRINT"Your deduction is correct after ";A$:PRINT"suspects."
1690  PRINT:PRINT:PRINT"..THE END..":END
1700  Z=INT(RND(8)*R)+1:RETURN
1710  RESTORE:FORY=1TOX:READA$:NEXTY:RETURN
1720  DATA cold & damp,hot & sticky,wet & windy
1730  DATA coffee was cold,beer was warm,beer was flat
1740  DATA fridge,TV,fan
1750  DATA oh dear,#%+!?@,beep!
1760  DATA water,bones,bum
1770  DATA Health & Efficiency,The Beano,a Sega magazine
1780  DATA a game of The House!,a vasectomy,a booze up
1790  DATA cigar,cig butt,cigarette
1800  DATA'Well it ain't a brothel!'
1810  DATA'Why do you want to know?
1820  DATA'Sorry it's my day off'
1830  DATA sawn-off shotgun,spud gun,rocket launcher
```

```
1840 DATA hernia,rupture,gammy leg
1850 DATA old Anglia,Mini,fast Lamborghi
ni
1860 DATA kitchen,dining room,living roo
m,study
1870 DATA Miss Lustie,Riff-Raff (the but
ler),Lady Wallop
1880 DATA Dr.Kutkitoff,Mr P.Brains,Merli
n (the cat),Vanessa (the vampire)
1890 DATA none,one,two,threee,four,five,s
ix,seven
1900 DATA 7:02,7:04,7:06
1910 DATA She"s perverted & knows it
1920 DATA She has buck teeth & acne
1930 DATA She crosses her legs & flashes
 her thighs
1940 DATA He sits nervously & scratches
his bum
1950 DATA"He belches & says,"Pardon:"
1960 DATA He furtively checks his fly
1970 DATA She sighs deeply
1980 DATA Her weight busts the chair
1990 DATA She says "Come up & see me som
etime"
2000 DATA He spills his drink on his tro
users
2010 DATA You notice he is wearing no un
dies
2020 DATA "You"re a big boy"-he says
2030 DATA He wipes his monocle
2040 DATA He stammers "I never touched h
er"
2050 DATA He squirms in his chair
2060 DATA He starts to purrrrr
2070 DATA He licks himself
2080 DATA He says"Mmeeooowww!!"
2090 DATA She has blood on her neck
2100 DATA A starts to clean her fangs
2110 DATA "She says," I have given the ma
ster much pleasure!"_"
2120 FORI=0TO2:BEEP:NEXT
2130 IFINKEY$=""THEN2130
2140 RETURN
2150 RESTORE:FORA=1TO100:READA$:PRINTA:"
 ":A$
2160 IFINKEY$=""THEN2160
2170 NEXT
```

MORE ON THE JOYSTICK AND MACHINE CODE

By Ian Nicholson

INTRODUCTION

A previous article described a method to control movement of a sprite across the screen using a machine code utility.

It was necessary to write the utility in machine code as the equivalent in BASIC would have been too slow in operation.

One of the exercises at the end of the above mentioned article was to make the sprite move up and down as well as left and right.

This article will show you how to move the sprite in any of the eight available directions. You will be also shown how to detect the operation of the fire button and collision of sprites all in machine code.

The machine code of the program which will produc e sound effects resulting from the operation of the fire button and collision of sprites. An overall description and dissection of the program is also given.

Explanation of the Basic Program

The BASIC program is in the main self explanitory using REM statements, however, further explanation is necessary on a few points.

Line 20 (POKE &H8169, &HC6) sets end of memory pointer to &HC6FF. This prevents the BASIC program from interfering with the machine code routines (as can happen if extra commands are added tot he BASIC program). The top of memory pointer is set just below the start of the machine code routine.

Line 40 (RESORE 70) makes sure that the READ statement in Line 60 reads the data in line 70.

Lines 50 and 60 pokes the machine code data contained in line 70 into memory locations ?HC700 to &HC74F inclusive. Note that the &H is not necessary in the DATA statements as this is taken care of in line 60.

Lines 80 to 310 poke in machine code routines as indicated in the REM statements within the program. Note that these could have been condensed into fewer lines but would make understanding more difficult.

Lines 330 and 340 set the patterns for the two sprites used within the program.

Line 350 selects SCREEN 2 (the graphics screen).

Line 360 sets SPRITE 6 at location (50, 70) with colour 1 (black).

Line 370 similarly sets SPRITE 0 to location (99, 99).

Line 390 calls the machine code routine which was poked in from line no's 50 onwards. This enables you to move SPRITE 0 with the aid of a joystick plugged into prot 1 and also detects the collision of two sprites as well as responding to the fire button. An explanation is given later.

Line 410 (OUT &HDF, &H92) passes control from the joystick back to the keyboard. Note that the program returns from the machine code routine to this point on either the left hand fire button being pressed or on a collision between sprites.

Line 420 looks at the value placed in memory location &HC7F0 during the operation of the machine code routein. The value placed depends on weather the fire button has been pressed or a collision between sprites occured. If the value is 16 (10 HEX) this indicates that the left hand fire button has been activated and the program advances to line 470 to make a gunshot noise. Otherwise the program continues to line 450 to make a collision noise.

On completion of either sound the program returns to line 390 to recall the machine code routine.

Explanation of the Flow Diagram

Figure 1 shows the machine code part of the program to move the sprite in any direction as well as detecting if the fire button has been pressed or a collision between sprites taken place.

A brief explanation is as follows. (for a more detailed explanation see the previous article)

1. From BASIC a call is made to the machine code routine.
2. On entry to machine code the computer is initialised to read the condition of the joystick and fire button states.
3. If no action has taken place from the joy stick then the program simply rotates about the first loopl.
4. On a change of stte from the joystick the program is set further in motion until the state of the joystick is recognised by one of the decision boxes.
5. From the appropriate decision box the program branches out to the right to obtain the current X and Y values; increasing or decreasing them as appropritate.

6. On exit from the above a check is made to determine if a collision between the two sprites has occured.
7. If collision has taken place then a return to BASIC is made and the appropriate noise made.
8. On a state of no collision the program then enters a delay loop to slow the movement of the sprite down, otherwise its speed would be excessive.
9. The program now returns to sample the state of the joystick again.

Note that if the fire button had been pressed then the program will advance down to the lower decision box where on exit from the "YES PORT" will return the program to BASIC to make a gunshot noise.

Machine Code

To allow the program to work with either the IIIA or IIIB memories the machine code is inserted from &HC700 onwards.

The code has been written in a form such that is grouped in small sections to corespond to each block of the flow diagram. Additional comments are written against the machine code listing. These correspond to the comments within the boxes of the flow diagram.

Hopefully this exercise has given you a further insite on how machine code programming can be used to further your use of the SEGA computer.

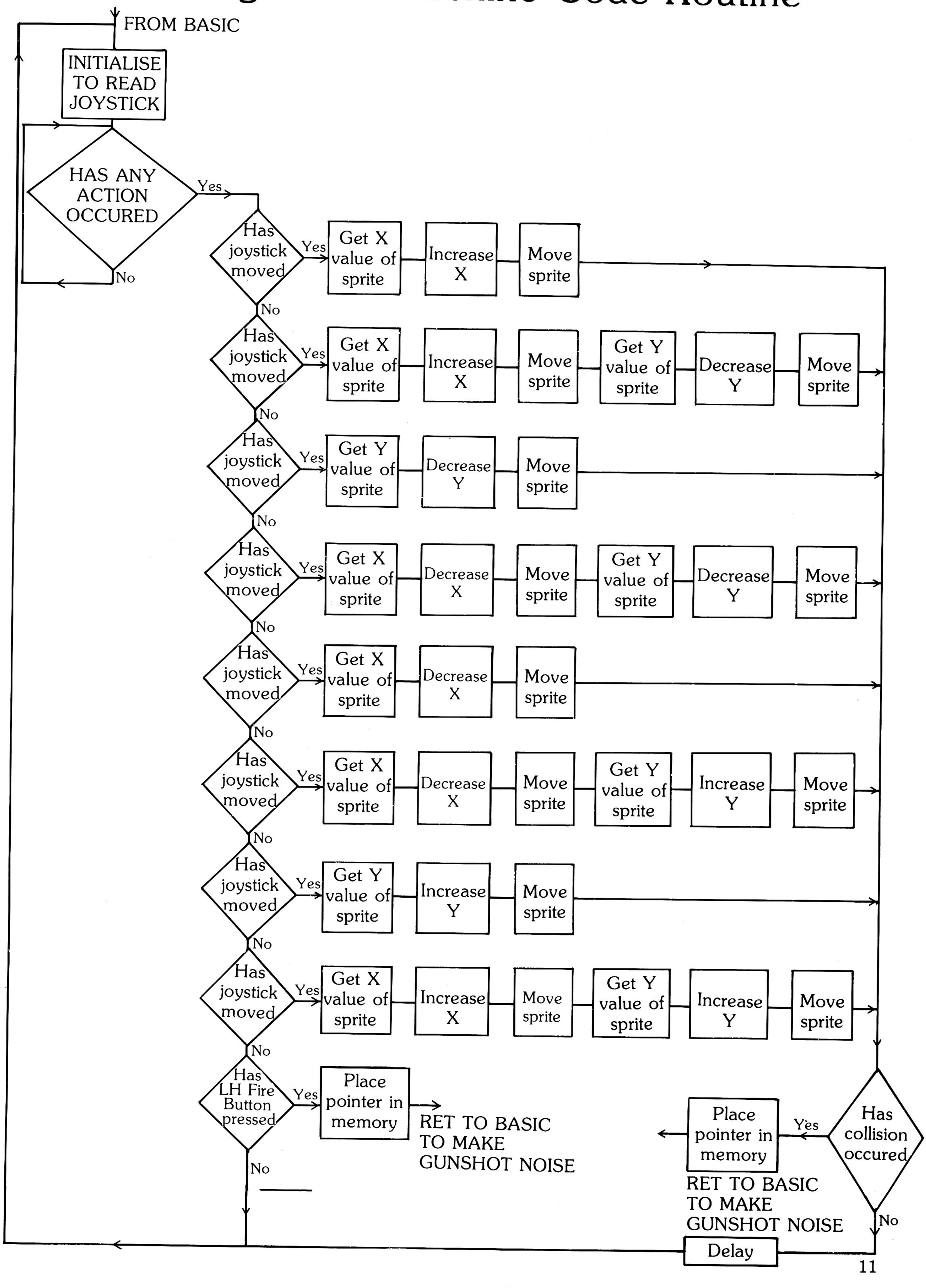
For those of you so inclined, you could try to rewrite the noises, which are written in BASIC into machine code routines and incorporate this into the machine code used within this article.

Finally, a word of caution. Make certain that the machine code contained within the DATA statements is typed in exactly as listed otherwise when the program is run a crash is likely. You have been warned. Happy computing.

```
C700 3E9B        LD A,9B       }
C702 D3DF         OUT (DF),A   }  INITIALISATION TO READ JOYSTICK
C704 DBDC         IN A,(DC)       GET VALUE FROM JOYSTICK
C706 FEFF         CP FF           HAS ANY ACTION OCCURED
C708 28FA         JR Z C704       IF NOT JUMP TO C704
C70A FEF7         CP F7           HAS JOYSTICK MOVED →
C70C 200A         JR NZ C718      JUMP TO C718 IF NOT
C70E CDC0C7       CALL C7C0       GET X VALUE OF SPRITE
C711 3C           INC A           INCREASE X
C712 CDE0C7       CALL C7E0       MOVE SPRITE
C715 C3A1C7       JP C7A1         JUMP TO COLLISION ROUTINE
C718 FEF6         CP F6           HAS JOYSTICK MOVED ↗
C71A 2011         JR NZ C72D      JUMP TO C72D IF NOT
C71C CDC0C7       CALL C7C0       GET X VALUE OF SPRITE
C71F 3C           INC A           INCREASE X
C720 CDE0C7       CALL C7E0       MOVE SPRITE
C723 CDD0C7       CALL C7D0       GET Y VALUE OF SPRITE
C726 3D           DEC A           DECREASE Y
C727 CDE0C7       CALL C7E0       MOVE SPRITE
C72A C3A1C7       JP C7A1         JUMP TO COLLISION ROUTINE
C72D FEFE         CP FE           HAS JOYSTICK MOVED ↑
C72F 200A         JR NZ C73B      JUMP TO C73B IF NOT
C731 CDD0C7       CALL C7D0       GET Y VALUE OF SPRITE
C734 3D           DEC A           DECREASE Y
C735 CDE0C7       CALL C7E0       MOVE SPRITE
C738 C3A1C7       JP C7A1         JUMP TO COLLISION ROUTINE
C73B FEFA         CP FA           HAS JOYSTICK MOVED ↖
C73D 2011         JR NZ C750      JUMP TO C750 IF NOT
C73F CDC0C7       CALL C7C0       GET X VALUE OF SPRITE
C742 3D           DEC A           DECREASE X
C743 CDE0C7       CALL C7E0       MOVE SPRITE
C746 CDD0C7       CALL C7D0       GET Y VALUE OF SPRITE
C749 3D           DEC A           DECREASE Y
C74A CDE0C7       CALL C7E0       MOVE SPRITE
C74D C3A1C7       JP C7A1         JUMP TO COLLISION ROUTINE
C750 FEFB         CP FB           HAS JOYSTICK MOVED ←
C752 200A         JR NZ C75E      JUMP TO C75E IF NOT
C754 CDC0C7       CALL C7C0       GET X VALUE OF SPRITE
C757 3D           DEC A           DECREASE X
C758 CDE0C7       CALL C7E0       MOVE SPRITE
C75B C3A1C7       JP C7A1         JUMP TO COLLISION ROUTINE
C75E FEF9         CP F9           HAS JOYSTICK MOVED ↙
C760 2011         JR NZ C773      JUMP TO C773 IF NOT
```

```
C762 CDC0C7     CALL C7C0     GET X VALUE OF SPRITE
C765 3D         DEC A         DECREASE X
C766 CDE0C7     CALL C7E0     MOVE SPRITE
C769 CDD0C7     CALL C7D0     GET Y VALUE OF SPRITE
C76C 3C         INC A         INCREASE Y
C76D CDE0C7     CALL C7E0     MOVE SPRITE
C770 C3A1C7     JP C7A1       JUMP TO COLLISION ROUTINE
C773 FEFD       CP FD         HAS SPRITE MOVED ↓
C775 200A       JR NZ C781    JUMP TO C781 IF NOT
C777 CDD0C7     CALL C7D0     GET Y VALUE OF SPRITE
C77A 3C         INC A         INCREASE Y
C77B CDE0C7     CALL C7E0     MOVE SPRITE
C77E C3A1C7     JP C7A1       JUMP TO COLLISION ROUTINE
C781 FEF5       CP F5         HAS SPRITE MOVED ↘
C783 2011       JR NZ C796    JUMP TO C796 IF NOT
C785 CDC0C7     CALL C7C0     GET X VALUE OF SPRITE
C788 3C         INC A         INCREASE X
C789 CDE0C7     CALL C7E0     MOVE SPRITE
C78C CDD0C7     CALL C7D0     GET Y VALUE OF SPRITE
C78F 3C         INC A         INCREASE Y
C790 CDE0C7     CALL C7E0     MOVE SPRITE
C793 C3A1C7     JP C7A1       JUMP TO COLLISION ROUTINE
C796 FEEF       CP EF         HAS LH FIRE BUTTON PRESSED
C798 C200C7     JP NZ C700    JUMP TO C700 IF NOT
C79B 3E10       LD A,10       LOAD A REG WITH 10
C79D 32F0C7     LD (C7F0),A   C7F0 LOADED WITH A (10)
C7A0 C9         RET           RET TO BASIC
C7A1 DBBF       IN A,(BF)     COLLISION ROUTINE
C7A3 E620       AND 20        "
C7A5 FE20       CP 20         "
C7A7 C2B0C7     JP NZ C7B0    "
C7AA 3E20       LD A,20       "
C7AC 32F0C7     LD (C7F0),A   "
C7AF C9         RET           RET TO BASIC
C7B0 010003     LD BC,0300    DELAY ROUTINE
C7B3 0B         DEC BC        "
C7B4 3E00       LD A,00       "
C7B6 B8         CP  B         "
C7B7 20FA       JR NZ C7B3    "
C7B9 C300C7     JP C700       JUMP TO C700
C7BC 00         NOP           NO ACTION
C7BD 00         NOP           "
C7BE 00         NOP           "
C7BF 00         NOP           "
C7C0 21013B     LD HL,3B01    3B01 HAS X VALUE OF SPRITE 0
C7C3 CD322C     CALL 2C32     CALL ROUTINE TO FIND VALUE AT 3B01
C7C6 DBBE       IN A,(BE)     PUT VALUE INTO A REGISTOR
C7C8 C9         RET           RETURN TO NEXT INSTRUCTION
C7C9 00         NOP           NO ACTION
C7CA 00         NOP           "
C7CB 00         NOP           "
C7CC 00         NOP           "
C7CD 00         NOP           "
C7CE 00         NOP           "
C7CF 00         NOP           "
C7D0 21003B     LD HL,3B00    3B00 HAS Y VALUE OF SPRITE 0
C7D3 CD322C     CALL 2C32     CALL ROUTINE TO FIND VALUE AT 3B00
C7D6 DBBE       IN A,(BE)     PUT VALUE INTO A REGISTOR
C7D8 C9         RET           RETURN TO NEXT INSTRUCTION
C7D9 00         NOP           NO ACTION
C7DA 00         NOP           "
C7DB 00         NOP           "
C7DC 00         NOP           "
C7DD 00         NOP           "
C7DE 00         NOP           "
C7DF 00         NOP           "
C7E0 CD442C     CALL 2C44     ROUTINE TO MOVE SPRITE
C7E3 D3BE       OUT (BE),A    MOVES SPRITE ON SCREEN
C7E5 C9         RET           RETURN TO NEXT INSTRUCTION
```

Flow diagram of Machine Code Routine



GRANDSTAND COMPUTERS LTD

If you are having any problems obtaining any of the pieces of software please contact Grandstand Computers enclosing a cheque or postal order.

SOFTWARE

CARTRIDGES	PRICE
☐ BASIC LEVEL IIIB	$175
☐ MUSIC	$125
☐ N/SUB	$39.95
☐ SAFARI HUNTING	$39.95
☐ BORDER LINE	$39.95
☐ CONGO BONGO	$39.95
☐ YAMATO	$39.95
☐ STAR JACKER	$39.95
☐ CHAMPION TENNIS	$39.95
☐ MONACO GP	$39.95
☐ CHAMPION BASEBALL	$39.95
☐ SINBAD MYSTERY	$39.95
☐ VIDEO FLIPPER	$39.95
☐ PACAR	$39.95
☐ POP FLAMER	$39.95
☐ CHAMPION GOLF	$49.95
☐ EXERION	$49.95
☐ SAFARI RACE	$49.95
☐ LODE RUNNER	$49.95
☐ BOXING	$54.95
☐ ORGUS	$49.95
☐ FLICKY	$59.95

HOME/PERSONAL/FINANCE CASSETTES

☐ FILE SYSTEM	$19.95
☐ CHEQUE BOOK RECONCILIATION	$19.95
☐ LOAN & MORTGAGE CALCULATION	$19.95
☐ ACCOUNTS RECEIVABLE	$30.00
☐ ACCOUNTS PAYABLE	$30.00
☐ 32K WORD PROCESSOR	$30.00
☐ PERSONAL RECORD KEEPER	$30.00
☐ 16K EASY WRITER	$19.95
☐ GRAPH AND CHART	$19.95
☐ HUCAL	$99.97 disc only spread sheet
☐ SEGA BASE	$99.97 disc only database
☐ LOGO	$72.22 disc only language

BOOKS

TEACH YOURSELF BASIC GAMES PROGRAMMING	$39.95	(Book and Cassette)
MORE THAN 50 PROGRAMS FOR THE SEGA SC-3000	$24.95	
SEGA BEGINNERS GUIDE	$14.95	
SELECTED BACK ISSUES OF SEGA MAGAZINE	$ 6.95	

GAME CASSETTES	PRICE
☐ TOWERS OF HANOI	$19.95
☐ HANGMAN	$19.95
☐ CITY-LANDER	$19.95
☐ CUBE-IT	$30.00
☐ BUGALOO	$19.95
☐ LASER BLAST	$19.95
☐ MUNCHMAN	$30.00
☐ ENTERPRISE ESCAPE	$19.95
☐ MARSMOBILE	$19.95
☐ MARS ADVENTURER	$19.95
☐ EMPIRE	$19.95
☐ KALAH	$19.95
☐ CHESS	$19.95
☐ DEATH SATELLITE	$19.95
☐ BIT BYTE	$19.95
☐ THE HOUSE	$30.00
☐ THE PYRAMID	$19.95
☐ CODE BREAKER	$19.95
☐ BACKGAMMON	$19.95
☐ LUMBER JACK	$19.95
☐ TRAMPMAN	$19.95
☐ SEGA GAMES PACK I	$19.95
☐ HELP!	$19.95

EDUCATIONAL CASSETTES	PRICE
☐ ROCKET MATHS	$19.95
☐ LEARNING ALPHABET	$19.95
☐ WATCH ME DRAW	$19.95
☐ SPELLING	$19.95
☐ ADDITION	$19.95
☐ SUBTRACTION	$19.95
☐ MULTIPLICATION	$19.95
☐ SPRITE EDITOR	$19.95
☐ MUSIC EDITOR	$19.95
☐ TYPING TUTOR	$19.95
☐ LEARNING TO COUNT	$19.95
☐ SHAPE & COLOUR QUIZ	$19.95
☐ MUSIC CARTRIDGE DEMONSTRATOR	$19.95
☐ SEGA GRAPHIC DESIGNER	$19.95
☐ GEOGRAPHY QUIZ I	$19.95
☐ GEOGRAPHY QUIZ II	$19.95
☐ MATHS HANG UP	$19.95

I enclose payment

☐ Cash ☐ Cheque ☐ Postal order ☐ Visa ☐ Bankcard

NAME . . . . . CARD NUMBER

ADDRESS . . . . . |_|_|_|_|_|_| |_|_|_|_|_|_| |_|_|_|_|_|_|

. . . . . CARD EXPIRY DATE . . . . . .

SIGNATURE . . . CHEQUE NO . . . . . .

GRANDSTAND COMPUTERS LTD P.O. BOX 2353 AUCKLAND

First Steps into Artificial Intelligence

M Howard

The actual of concepts of AI are extremely complex . . . well let's face it, the human mind (with the possible exception of mine) is a very complex affair and any attempts at trying to simulate it are always going to be hard to understand.

So hopefully this simple . . . but interesting program will introduce you to some of the concepts of AI.

The program acts as a database (well sort of any way!) . . . it starts off by asking you to think of an animal. It then asks you if the animal you have thought of is a cat. If it is then press "Y" else press "N". If you typed "N" then you are asked to enter the name of the animal you thought of and a way of telling the difference between it and a cat. EG;

Animal thought of . . . dog

Is it a cat?

No

What was it? . . . dog

Give a question that is true for a dog and false for a cat.

does it bark?

From here you can set up what is called a binary tree of data, by setting up lots of questions and answers by simply answering "Y" or "N" and ending up with a great big data base. Experiment with the program a little . . . but don't forget this was one of the first AI programs ever . . . so don't expect too much!!

```
10  DIMA$(1024)
20  A$(1)="CAT"
30  N=1
40  PRINT"Think of an animal"
50  IFA$(2*N)<>""THEN120
60  PRINT"Is it a ";A$(N);"?"
70  GOSUB170:IFFTHENPRINT"I got it!!":GOT
030
80  A$(2*N)=A$(N)
90  INPUT"Well what is it ";A$(2*N+1)
100  PRINT"Give me a question that would
be true for a ";A$(2*N+1);"....":PRINT"B
ut false for a ";A$(2*N)
110  INPUTA$(N):GOTO30
120  PRINTA$(N);"?"
130  GOSUB170
140  N=2*N+F
150  IFN<512THEN50
160  END
170  X$=INKEY$:IFX$<>"Y"ANDX$<>"N"THEN170
180  F=0:IFX$="Y"THENF=1
190  RETURN
```

DISC BONANZA

As many people know, the price of the SF-7000 disc unit has dropped to $695 and as you can guess the quantity of sales has taken off drastically . . . so to help all those with the SF-7000, a few of the Machine Code programs that have appeared in previous magazines have been converted to work on the Disc system. So with no more woffle here's a list of programs . . .

****

Sprite Mover

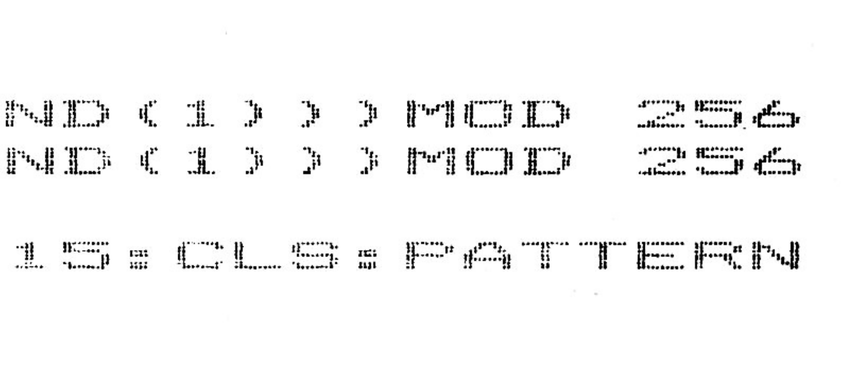
****

This program allows you to move all the sprites about the screen in any direction and at any speed. Touse it, just enter in the program, RUN it and wait for a short while, you'll then see 32 sprites all move about the screen.

Refer to Jan/Feb issue for more details.

```
10 DATA 11,4,0,21,0,3B,DD,21,0,F1,E,D0,6
,20,F3,DD,35,1,20,28,DD,7E,0,DD,77,1,CD,
6F,0,DB,BE,DD,B6,2,B9,20,3,DD,B6,2,CD,69
,0,D3,BE,23,CD,6F,0,DB,BE,DD,B6,3,CD,69,
0,D3,BE,2B,19,DD,19,10,CE,C9
20 FOR A=&HF000 TO &HF041:READA$:POKEA,V
AL("&h"+A$):NEXT
30 FOR A=&HF100 TO &HF180 STEP4
40 POKE A,1
50 POKE A+1,1
60 POKE A+2,(253+INT(10*RND(1)))MOD 256
70 POKE A+3,(253+INT(10*RND(1)))MOD 256
80 NEXT
90 SCREEN 2,2:COLOR1,14,,15:CLS:PATTERN
S#0,"FF8181818181FF"
100 FOR S=0 TO 31
110 SPRITE S,(120,90),0,RND(1)*16
120 NEXT
130 CALL &HF000:GOTO 130
```

SP-400 Printer Dump



This one will let you copy the Hi-re screen to the SP-400 printer plotter. Refer to June issue.

```
10 DATA0,0,0,ED,5B,0,F0,CD,14,01,FE,0,C0
,ED,5B,0,F0,1C,AF,BB,20,A,14,3E,C0,BA,20
,4,32,2,F0,C9,ED,53,0,F0,C3,3,F0
20 FORA=&HF000TO&HF026:READA$:POKEA,VAL(
"&h"+A$):NEXT
30 LPRINTCHR$(18):POKE&HF000,0:POKE&HF00
1,0:POKE&HF002,0:CALL&HF003
40 LPRINT"M";PEEK(&HF000);"","";191-PEEK(&
HF001);LPRINT"J1,0":CALL&HF000:GOTO40
```

Hi-Resolution Scrolling

These two programs will let you scroll information on the Hi-res screen a pixel at a time . . . producing brilliant effects. Refer to June issue for more info.

```
Right Hi-Res Scrolling. .Disc Version

10 DATA0,3E,0,32,0,F0,D0,3C,32,0,F0,C9
20 DATAE5,CD,69,0,D3,BE,0,E1,C9
30 DATAE5,CD,6F,0,DB,BE,0,E1,C9
40 DATAF3,6,20,C5,CD,15,F0,CB,3F,F5,47,3
A,0,F0,FE,0,28,2,CB,F8,f1,CD,1,F0,78,CD,
C,F0,C1,11,8,0,AF,ED,SA,10,DE,C9
50 DATA6,8,21,0,0,C5,E5,AF,32,0,F0,CD,1E
,F0,E1,23,C1,10,F2,C9
60 DATA6,6,C5,CD,44,F0,C1,10,F9,C9
100 T=0:RESTORE:FORA=&HF000TO&HF061:READ
A$:B=VAL("&H"+A$):T=T+B:POKEA,B:NEXT:IFT
<>12557THENPRINT"Error":END
1000 A$="9876543210987654321098765432109
87654321"
1010 SCREEN 2,2:CLS:FORA=1TOLEN(A$):CURS
OR0,0:PRINTMID$(A$,A,1):CALL&HF058:NEXT
1020 FORA=0TO41:CALL&HF058:NEXT

Left Hi-Res Scrolling  Disc Version

10 DATA0,3E,0,32,0,F0,D0,3C,32,0,F0,C9
20 DATAE5,CD,69,0,D3,BE,0,E1,C9
30 DATAE5,CD,6F,0,DB,BE,0,E1,C9
40 DATAF3,6,20,C5,CD,15,F0,CB,27,f5,47,3
A,0,F0,FE,0,28,2,CB,C0,f1,CD,1,F0,78,CD,
C,F0,C1,11,8,0,AF,ED,52,10,DE,C9
50 DATA6,8,21,F8,0,C5,E5,AF,32,0,F0,CD,1
E,F0,E1,23,C1,10,F2,C9
60 DATA6,6,C5,CD,44,F0,C1,10,F9,C9
100 T=0:RESTORE:FORA=&HF000TO&HF061:READ
A$:B=VAL("&H"+A$):POKEA,B:T=T+B:NEXT:IFT
<>12717THENPRINT"Error":STOP
1000 A$="Hello this is a test. .123456789
0........"
1010 SCREEN 2,2:CLS:FORA=1TOLEN(A$):CURS
OR247,0:PRINTMID$(A$,A,1):CALL&HF058:NEX
T
1020 FORA=0TO40:CALL&HF058:NEXT
```

Flip!

!qiF

This silly program will . . . no I won't tell you! Just enter it type in RUN followed by a NEW then CALL &HF000 and see what happens. CALL &HF000 to set things back to normal.

```
10 DATA F3,21,0,18,1,0,8,C5,E5,CD,6F,0,D
B,BE,E,0,6,6,17,CB,15,10,F8,E1,E5,C5,CD,
69,0,C1,79,D3,BE,E1,C1,23,8,78,B1,20,DE,
C9
20 FOR A=&HF000 TO &HF029:READA$:POKEA,V
AL("&H"+A$):NEXT
```

Purple People Eaters versus Man

an exercise in ecological modelling

M Howard

Computers are designed for serious as well as entertainment purposes . . . well this program is SERIOUS . . . Okay no need to cringe or hide behind the chair!! The program can in fact be quite funny and enjoyable to watch . . . here is what it does.

You firstly select a number of Purple People Eaters . . . those who don't yet know what a PPE is ought to be ashamed!! and then select a number of humans. These bodies are then placed in an "Ecological Chamber". From now on the following rules apply:-

1) Lots of humans lead to lots of PPE's, because PPE's eat humans (obvious really) . . .!!
2) Lots of PPE's lead to humans dying out, as they all get munched!
3) Lots of PPE's lead to no humans, thus leading to no PPE's, because if there are no people to chew on then the PPE's die out!
4) Too few PPE's lead to an explosion in the population of humans, but too many humans die out due to lack of food and space.
5) Etc etc etc etc. . . . .

As you can see this program can end up giving quite complex results due to it's many rules and regulations.

Okay, now how do you run the program? Well it's really easy, firstly enter the number of humans and numbers of PPE's you wish to experiment on. Then just wait a while whilst the computer generates the original mode. From now on you just select the option displayed after each generation. These options include:-

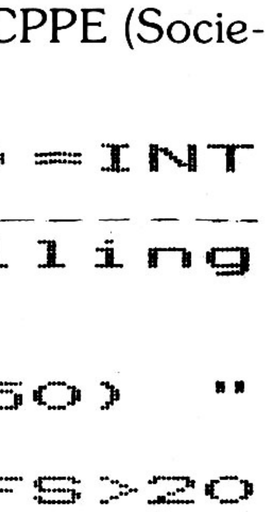
1) Generate a new cycle.
2) Quit and make a new experiment.
3) Draw up a histogram.
4) Draw up a table of data.

And they are all fairly self explanatory so I won't bother explaining them. Any way, I hope you like the program and find it of good use. But remember two things:

A)If the program appears to be doing nothing then look at the values next to the PPE and people population indicators . . . if there are no numbers then the Sega is working out the next generation. . . If, on the other hand, there are some numbers there . . . press a key giving on the menu!

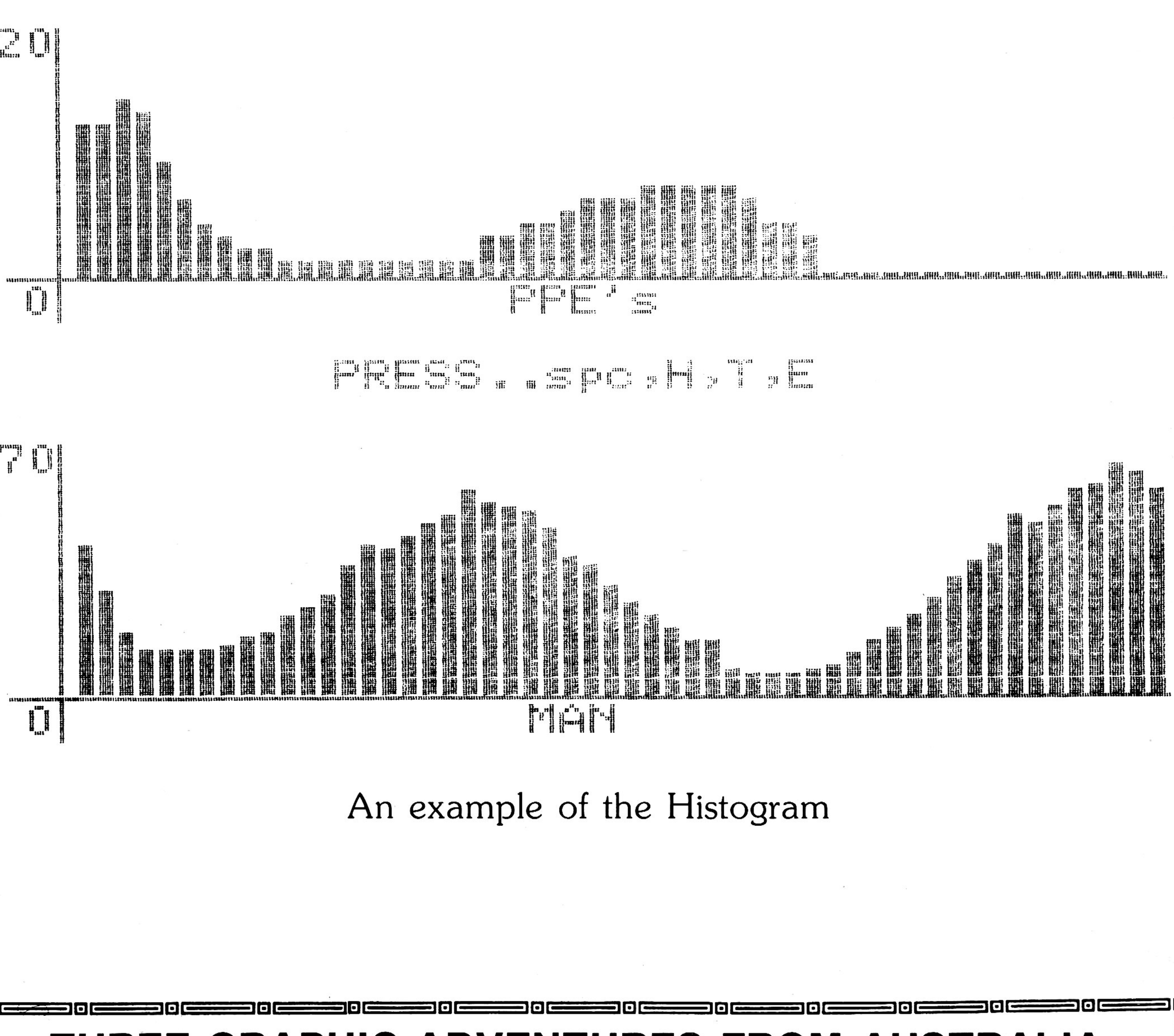
B) Don't go about killing too many PPE's or you'll have a spokesmonster on your back from the SPCPPE (Society for the Prevention of Cruelty to Purple People Eaters!!!)

```
10 DIMA(125),F(200),S(400):DEFFNR(X)=INT
(RND(8)*X)
20 CLS:PRINTTAB(7);"Ecological modelling
"
30 PRINT
40 INPUT"Enter number of people (1-50) "
;F:IFF>50ORF<1THEN40
50 PRINT:INPUT"& PPE's (1-20) ";S:IFS>20
ORS<1THENS0
60 PRINT:PRINT"Press a key"
70 IFINKEY$=""THEN70
80 CLS
90 GOSUB790
100 FORJ=1TOF
110 R1=FNR(100)+11:IFA(R1)THEN110
120 A(R1)=10:NEXTJ
130 FORJ=1TOS
140 R1=FNR(100)+11:IFA(R1)THEN140
150 A(R1)=3:NEXTJ
160 C=1:GOSUB170:GOTO240
170 I=11:S=0:F=0
180 FORJ=4TO13:FORK=3TO12:CURSORK,J
190 IFA(I)=0THENPRINT"  ":GOTO220
200 IFA(I)<100THENPRINT"P":S=S+1
210 IFA(I)=100THENPRINT"M":F=F+1
220 I=I+1:NEXTK,J
230 F(C)=F:S(C)=S:RETURN
```

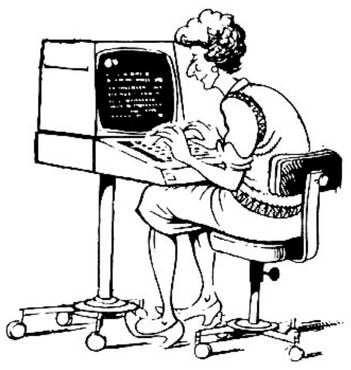


```
240  A$=INKEY$: IFA$=""THEN240
250  SCREEN  1, 1
260  IFA$=""  "ANDG<5BTHEN310
270  IFA$="H"THEN670
280  IFA$="E"THEN20
290  IFA$="T"THEN590
300  BEEP2:GOTO240
310  CURSOR21,5:PRINT"    ":CURSOR22,6:PRIN
T"    "
320  FORI=11TO110
330  IFA(I)=100RA(I)=0THEN440
340  X=INT((A(I-10)+A(I+10)+A(I-1)+A(I+1)
)/100)
350  IFA(I-10)=100THENA(I-10)=0
360  IFA(I+10)=100THENA(I+10)=0
370  IFA(I+1)=100THENA(I+1)=0
380  IFA(I-1)=100THENA(I-1)=0
390  A(I)=A(I)+X-1
400  F=F-X
410  IFA(I)<6THEN440
420  A(I)=3
430  R1=FNR(100)+11:A(R1)=3
440  NEXTI
450  FORI=11TO110
460  IFA(I)=100THENA(I)=0
470  NEXTI
480  F=F+INT(F/2)
490  IFF=0THEN540
500  IFF>100-5THENF=100-5
510  FORJ=1TOF:R1=FNR(100)+11
520  IFA(R1)=0THENA(R1)=100
530  NEXTJ
540  C=C+1:G=C
550  GOSUB170
560  CURSOR21,3:PRINTC-1:CURSOR20,5:PRINT
5:CURSOR21,6:PRINTF
570  FORI=1TO10:A(I)=0:A(I+110)=0:NEXTI
580  GOTO240
590  CLS:PRINT"    PPE  vs  MAN";;;
600  PRINT"  C       P       M"
610  C=2:P=0
620  PRINTC-1;TAB(6);S(C);TAB(11);F(C)
630  C=C+1
640  IFC>0THENPRINT"End":FORI=0TO400:NEXT
:GOSUB790:C=C-1:GOTO240
650  IFINKEY$=""THEN650
660  GOTO620
670  SCREEN  2,2:CLS:CURSOR90,3:COLOR12:PR
INT"PPE's  vs  MAN"
680  LINE(20,20)-(20,90),1:LINE(10,80)-(2
54,80):LINE(20,190)-(20,120):LINE(10,180
)-(254,180)
690  CURSOR13,82:PRINT"0":CURSOR13,182:PR
INT"0":CURSOR8,20:PRINT"20":CURSOR8,120:
PRINT"70"
700  COLOR13:CURSOR110,82:PRINT"PPE's":CO
LOR4:CURSOR113,182:PRINT"MAN"
710  FORI=1TO8
720  IFF(I)>70THENF(I)=70
730  IFS(I)>20THENS(I)=20
740  LINE(20+4*I,79)-(22+4*I,79-(3*S(I)))
,13,BF
750  LINE(20+4*I,179)-(22+4*I,179-(.857*F
(I))),4,BF
760  NEXT
770  COLOR12:CURSOR75,100:PRINT"PRESS..sp
c.H.T.E"
780  GOTO240
790  CLS:CURSOR2,3:PRINT"!----------!":CU
RSOR15,3:PRINT"Cycle#":FORI=4TO13:CURSOR
```

```
2,I:PRINT"!                                    !":NEXT:CURSOR2,14:
PRINT"!————————————————————!":CURSOR2,0:PRINT"PPE'
S VS MAN"
800 CURSOR0,15:PRINT"spc—Next cycle",,"H
—Histogram",,"T—Table of data",,"E—Next
Experiment"
810 CURSOR21,3:PRINTC-1:CURSOR15,5:PRINT
"PPE's":S:CURSOR15,6:PRINT"People":F:RET
URN
```



An example of the Histogram



Hypno-Qix

M Howard

The following program is great for those lazy days when you feel like zonking out in front of the computer and having your mind blown!!

As it stands, it is very simple . . . a line will bounce around the screen at great speed and makes lots of nice sounds as it hits the walls . . . to stop it either wait 11 second or press a key at any time.

As the main body of the program is machine code . . . both ROM and Disc versions are given. All the best, just RUN the program and off you go . . . !!

Disc version

```
10 DATA0,0,0,0,0,0,0,0,8,ED,5F,A6,D3,7F,
3E,10,32,FF,FE,1A,ED,44,12,8,C9
20 DATA3A,FF,FE,FE,20,28,6,D3,7F,3C,32,F
F,FE,2A,4,F0,ED,5B,6,F0,3E,1,6,0,CD,F1,3
9,6,4,11,0,F0,21,4,F0,7E,FE,9,DC,8,F0,23
,13,10,F6,6,2,21,5,F0,11,1,F0,7E
30 DATAFE,BA,D4,8,F0,23,23,13,13,10,F4,6
,2,21,4,F0,11,0,F0,7E,FE,F9,D4,8,F0,23,2
3,13,13,10,F4,6,4,21,4,F0,11,0,F0,4E,1A,
81,77,23,13,10,F8,3A,49,AC,FE,10,DO,CD,4
8,1,FE,0,CA,19,F0,C9
40 T=0:FORA=&HF000TO&HF0BC:READA$:V=VAL(
"&H"+A$):POKEA,V:T=T+V:NEXT:IFT<>14064TH
ENPRINT"Error":STOP
50 DEFFNR(X)=INT(RND(8)*X):V=0
60 FORA=&HF000TO&HF003:POKEA,FNR(6):IFFN
R(10)>SANDPEEK(A)THENPOKEA,256-PEEK(A)
70 NEXT:X=RND(-1)
80 POKE&HF004,FNR(210)+20:POKE&HF005,FNR
(96)+20:POKE&HF006,FNR(210)+20:POKE&HF00
7,FNR(96)+20
90 S=RND(8)+RND(8):T=110:FORA=0TO10:SOUN
D1,T,15:S=S+S:T=T+S:NEXT:FORA=15TO0STEP-
1:SOUND1,,A:NEXT
100 SCREEN2,2:COLORFNR(14),15,,FNR(15):C
LS:TIME$="00:00:00"
110 CALL&HF019:SOUND0:GOTO60
```

Rom Version

```
10 DATA0,0,0,0,0,0,0,0,8,ED,5F,A6,D3,7F,
3E,10,32,FF,FE,1A,ED,44,12,8,C9
20 DATA3A,FF,FE,FE,20,28,6,D3,7F,3C,32,F
F,FE,2A,4,F0,ED,5B,6,F0,3E,1,6,0,CD,F1,3
9,6,4,11,0,F0,21,4,F0,7E,FE,9,DC,8,F0,23
,13,10,F6,6,2,21,5,F0,11,1,F0,7E
30 DATAFE,BA,D4,8,F0,23,23,13,13,10,F4,6
,2,21,4,F0,11,0,F0,7E,FE,F9,D4,8,F0,23,2
3,13,13,10,F4,6,4,21,4,F0,11,0,F0,4E,1A,
81,77,23,13,10,F8,3A,8E,94,FE,10,DO,CD,B
6,42,FE,0,CA,19,F0,C9
40 T=0:FORA=&HF000TO&HF0BC:READA$:V=VAL(
"&H"+A$):POKEA,V:T=T+V:NEXT:IFT<>14396TH
ENPRINT"Error":STOP
50 DEFFNR(X)=INT(RND(8)*X)
60 FORA=&HF000TO&HF003:POKEA,FNR(6):IFFN
R(10)>SANDPEEK(A)THENPOKEA,256-PEEK(A)
70 NEXT:X=RND(-1)
80 POKE&HF004,FNR(210)+20:POKE&HF005,FNR
(96)+20:POKE&HF006,FNR(210)+20:POKE&HF00
7,FNR(96)+20
90 S=RND(8)+RND(8):T=110:FORA=0TO10:SOUN
D1,T,15:S=S+S:T=T+S:NEXT:FORA=15TO0STEP-
1:SOUND1,,A:NEXT
100 SCREEN2,2:COLORFNR(14),15,,FNR(15):C
LS:TIME$="00:00:00"
110 CALL&HF019:SOUND0:GOTO60
```

Program of the Month

JET RANGER

Danny Van Kan

Do you love to blast away innocent little planes? If so then this great program is for you . . .! All you do is enter the program EXACTLY as listed . . . that includes the REM's at the start of the program as this is where the machine code is stored. To operate you simply type in RUN, wait a small while then off you go . . . The controls are simple . . . You use a Joystick, left and right and fire . . . simple.

```
10 REM AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
20 REM AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
30 REM AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA
40 GOSUB 890
50 DIMA$(6,16)
60 GOTO800
70 MAG3:SCREEN 2,2:CLS:PRINT CHR$(16):RE
STORE
80 COLOR15,5,(0,0)-(255,191),5
90 PATTERNS# 0,"0000000000040F07E"
100 PATTERNS#02,"00000000000000001F"
110 PATTERNS#04,"0307071F3F3F0F01"
120 PATTERNS#06,"0080C0C0F0F8FEE0"
130 PATTERNS#8,"010103020404 0F3F"
140 PATTERNS#9,"FFFFFF87030F1F01"
150 PATTERNS#10,"0000808040 40E0F8"
160 PATTERNS#11,"FEFEFEC280E0F0"
170 COLOR14:CURSOR40,4:PRINT"JET RANGER
by MEGA SOFT"
180 CURSOR100,180:PRINT "SCORE : 0"
190 COLOR10,4,(35,0)-(190,14)
200 COLOR10,4,(95,176)-(190,191)
210 LINE (32,0)-(190,14),10,B
220 LINE (88,176)-(190,191),10,B
230 SC=0
240 PATTERNC#48,"70989 8A8C8C870"
250 PATTERNS#32,"2430842 21B0F1FBF"
260 PATTERNS#33,"1F17254190 0101"
270 PATTERNS#34,"8000C484D1E0F0FB"
280 PATTERNS#35,"E0900800 0201"
290 PATTERNS#28,"ff8080808080 8080"
300 PATTERNS#29,"8080808080ff"
310 PATTERNS#30,"c0404040404040 40"
```

```
320  PATTERNS#31,"4040404040c0"
330  PATTERNC#39,"00000000000000000000"
340  PATTERNS#16,"070C3F3360C1C0C0"
350  PATTERNS#17,"643D"
360  PATTERNS#18,"F864268501C14593"
370  PATTERNS#19,"CCB0"
380  PATTERNS#13,"40e0e04040e0a0"
390  PATTERNS#15,"2070702020705050"
400  PATTERNS#20,"0003000C1F3E3F3F"
410  PATTERNS#21,"1E"
420  PATTERNS#22,"0098D8FAFE3EBA6C"
430  PATTERNS#23,"30"
440  FOR I=5 TO 11 STEP 2
450  A=INT(RND(1)*254)+1:B=INT(RND(1)*254
)+1
460  SPRITE1,(A,B),15,14
470  SPRITE1+1,(A,B),20,15
480  NEXT
490  FOR I=15TO 29 STEP 2
500  A=INT(RND(1)*254)+1:B=INT(RND(1)*254
)+1
510  SPRITE1,(A,B),15,14
520  SPRITE1+1,(A,B),20,15
530  NEXT
540  SPRITE0,(120,158),8,1
550  SPRITE1,(INT(RND(1)*255),0),8,1
560  SPRITE2,(120,191),12,10
570  POKE&H9A08,0
580  POKE&H9A0A,0
590  OUT127,228
600  OUT127,245
610  CALL&H984F
620  IF PEEK(&H9A0A)=&H35 THEN 750
630  GOTO 640
640  POKE&H9A08,0
650  VPOKE&H3B08,191:VPOKE&H3B09,VPEEK(&H
3B01)
660  IFVPEEK(&H3B04)>150THEN610
670 SC=SC+25:CURSOR140,180:PRINTCHR$(5):
CURSOR140,180:PRINTSC
680  VPOKE&H3B06,32:FORI=0TO3:VPOKE&H3B07
,6:VPOKE&H3B07,10:NEXT
690  GOSUB730
700  VPOKE&H3B06,16:VPOKE&H3B07,14
710  SPRITE1,(INT(RND(1)*255),192),8,1
720  GOTO 590
730  OUT127,228:FORA=240TO255:OUT127,A:FO
RI=0TO8:NEXT:NEXT:RETURN
740  N=127:OUTN,159:OUTN,191:OUTN,223:OUT
N,255:OUTN,231:OUTN,240:FORB=0TO15:FOR A
=192 TO 207:OUTN,A:OUTN,B:OUTN,240+B:NEX
T:NEXT:RETURN
750  COLOR15,1,(75,96)-(190,109)
760  LINE (72,95)-(192,110),10,B
770  PRINT CHR$(17):COLOR15:CURSOR80,100:
PRINT"GAME OVER"
780  OUT127,&H0F
790  FOR I=0 TO 500:NEXT :VPOKE&H3800,208
:FORI=0TO999:NEXT
800  SCREEN 1,1:CLS
810  COLOR4,15
820  CURSOR 5,6 :PRINT"Type in the skill
level":PRINT
830  PRINT :PRINT "                    - 16 Easy"
840  PRINT :PRINT "                    - 26 Impossibl
e"
850  PRINT :INPUT A
860  IF A=1 THEN POKE&H9AC0,0:POKE&H9AC1,
0:POKE&H9AC2,0:GOTO 70
870  IF A=2 THEN POKE&H9AC0,&HCD:POKE&H9A
C1,&H0D:POKE&H9AC2,&H98:GOTO 70
```

```
880  GOTO  850
890  RESTORE  930:FOR  I=&H980B  TO  &H98FA:R
EAD  A:POKEI,A:NEXTI
900  RESTORE  980:FOR  I=&H990B  TO  &H99E3:R
EAD  A:POKEI,A:NEXT
910  RESTORE  1030:FOR  I=&H9A60  TO  &H9AD5:
READ  A:POKEI,A:NEXT
920  RETURN
930  DATA  &HO,&HO,&HBC,&H3B,&HEE,&HF3,&H
E,&H1,&H32,&HB,&H98,&H3E,&H20,&H32,&H9,&
H98,&H21,&HC,&H3B,&H22,&HA,&H98,&H2A,&HA
,&H98,&HCD,&H32,&H2C,&HDE,&HBE,&H3C,&H32
,&HC,&H98,&H2A,&HA,&H98,&HCD,&H44,&H2C,&
H3A,&HC,&H98,&HD3,&HEE,&H21,&HA,&H98
940  DATA  &H34,&H34,&H34,&H34,&H2B,&H35,&
H20,&HDE,&H2B,&H35,&HFB,&HCB,&HE,&H1,&H6
,&H1,&H10,&HFE,&HD,&H20,&HF9,&H18,&HC4,&
HO,&HO,&HO,&H3E,&H3C,&H32,&H26,&H98,&H3E
,&HC,&H32,&H19,&H98,&H3E,&HA,&H32,&H14,&
H98,&HCD,&HD,&H98,&H3E,&H16,&H32,&H14
950  DATA  &H98,&HCD,&HD,&H98,&H3E,&H20,&H
32,&H14,&H98,&HCD,&HD,&H98,&HF3,&H21,&H4
,&H3B,&HCD,&H32,&H2C,&HDE,&HBE,&HD6,&HC0
,&HC2,&H8E,&H98,&H21,&H5,&H3B,&HC3,&H90,
&H9A,&HED,&H5F,&HCB,&H27,&HD3,&HBE,&HFB,
&HC3,&H8F,&H99,&H26,&H98,&H3E,&H4,&H32,&
H19
960  DATA  &H98,&H3E,&H1,&H3A,&HB,&H9A,&HD
6,&H1,&HC2,&HB,&H99,&HO,&HO,&HO,&HO,&HC3
,&HA5,&H99,&H3B,&HCD,&H32,&H2C,&HC6,&H6,
&H32,&H9,&H9A,&H21,&HB,&H3B,&HCD,&H32,&H
2C,&HDB,&HBE,&H47,&H3A,&H9,&H9A,&H90,&HF
B,&HC2,&HE6,&H98,&HF3,&H21,&HC,&H3E
970  DATA&HCD,&H4,&H2C,&H3E,&HD1,&HD3,&H
BE,&HC3,&H60,&H9A,&HEF,&HEE,&H20,&HCB,&H
21,&HB,&H3B,&HCD,&H44,&H2C,&H3E,&HC0,&HD
3,&HBE,&H3E,&HO,&H32,&HB,&H9A,&HFB,&H3E,
&H3D,&H32,&H26,&H9B,&H3E,&HB,&H32,&H19,&
H98,&H3E,&H1,&H32,&H14,&H9B,&HCD,&HD,&H9
8,&HC3,8,&H99
980  DATA  &HCD,&H1B,&H49,&H3A,&H61,&H94,&
HD6,&H10,&HCC,&H2A,&H99,&H3A,&H65,&H94,&
HD6,&H20,&HCC,&H30,&H99,&H3A,&H66,&H94,&
HD6,&H20,&HCC,&H65,&H99,&HC3,&H52,&H9B,&
HO,&HO,&H3A,&HB,&H99,&HD6,&H1,&HCB,&H3E,
&H1,&H32,&HB,&H9A,&HF3,&H21,&H1,&H3B,&HC
D
990  DATA  &H32,&H2C,&HDB,&HBE,&HC6,&H3,&H
21,&H9,&H3B,&HCD,&H44,&H2C,&HD3,&HBE,&HF
B,&HC9,&HO,&HO,&HO,&HO,&HO,&HO,&HO,&HO,&
H3E,&H3D,&H32,&H26,&H9B,&H3E,&H1,&H32,&H
19,&H98,&H3E,&H1,&H32,&H14,&H9B,&HCD,&HD
,&H98,&HC9,&HO,&HO,&H3E,&H3C,&H32
1000  DATA  &H26,&H9B,&H3E,&H1,&H32,&H19,&
H98,&H3E,&H1,&H32,&H14,&H9B,&HCD,&HD,&H9
8,&HC9,&H6,&HFF,&HDB,&HBF,&HE6,&HEF,&HEE
,&H20,&HC8,&H10,&HF7,&HC3,&HD6,&H9B,&H41
,&H41,&H41,&H41,&H41,&H41,&H41,&H41,&H41
,&H3E,&H3C,&H32,&H26,&H9B,&H3E,&H4,&H32,
&H19
1010  DATA  &H9B,&H3E,&H1,&H32,&H14,&H9B,&
HCD,&HD,&H9B,&HC3,&HCO,&H9A,&H41,&HF3,&H
21,&H4,&H3B,&HCD,&H32,&H2C,&HDB,&HBE,&HO
,&H32,&H9,&H9A,&H21,&HB,&H3B,&HCD,&H32,&
H2C,&HDB,&HBE,&H47,&H3A,&H9,&H9A,&H90,&H
FB,&HCA,&HC4,&H9B,&HF3,&H21,&H4,&H3B,&HC
D
1020  DATA  &H32,&H2C,&HDB,&HBE,&H3D,&H32,
&H9,&H9A,&H21,&HB,&H3B,&HCD,&H32,&H2C,&H
DB,&HBE,&H47,&H3A,&H9,&H9A,&H90,&HFB,&HC
```

```
2,&HE6,&H98,&HC3,&HC4,&H9B
1030 DATA &HF3,&H21,&H5,&H3B,&HCD,&H32,&
H2C,&HCD,&HD0,&H9A,&H21,&H9,&H3B,&HCD,&H
32,&H2C,&HDB,&HBE,&H90,&HF2,&H79,&H9A,&H
C3,&HD6,&H9B,&HD6,&H34,&H0,&HFA,&HB0,&H9
A,&H98,&H41,&H41,&H41,&H41,&H41,&H41,&H4
1,&H41,&H41,&H41,&H41,&H41,&H41,&H41,&H4
1,&H41
1040 DATA &HCD,&H44,&H2C,&HED,&H5F,&HCB,
&H27,&HD3,&HBE,&H3A,&HA,&H9A,&H3C,&H32,&
HA,&H9A,&HD6,&H35,&HCB,&HC3,&HBE,&H9B,&H
41,&H41,&H41,&H41,&H41,&H41,&H41,&H41,&H
41,&H41,&H3A,&HB,&H9A,&H3C,&H32,&HB,&H9A
,&HC9,&HD6,&H9B,&H41,&H41,&H41,&H41,&H41
,&H41
1050 DATA &HCD,&HD,&H98,&HC3,&H9B,&H98,&
H41,&H41,&H41,&H41,&H41,&H41,&H41,&H41,&
H41,&H41,&HDB,&HBE,&HD6,&H1A,&H47,&HC9
```

☆★☆ Flashing lights ★☆

The following piece of code is another one of those stupid programs that I tend to write when I'm bored!!

All it does is flash 8 big sprites on the screen in time to some data you are feeding in through the cassette port, the data fed in may be music if you want a disco light effect!!

Here is how to use it. Firstly enter the program as it stands, (don't forget if you have a Disc Drive to use the second lot of data), now feed in your data into the cassette IN port. If you have a SP-400 printer plotter connected then switch it OFF!! Now RUN the program, and after a short delay you'll see the sprites all flashing in time with the music or data (when you turn the cassette on of course!!) . . . Voila!

Note . . . you can alter the position of the sprites to wherever you like, because they are positioned in BASIC by lines 20-80.

Hope you like it.!!

```
10 DATA2,4,6,7,A,B,D,C,6,8,E,0,DB,DD,FE,
FF,20,2,CB,C1,CB,21,11,0,4,1E,7A,B3,20,F
B,10,EC,DB,DD,FE,FF,20,2,CB,C1,C5,6,8,78
,E,0,CD,98,42,10,FB,C1,6,8,21,FF,EF,CB,4
1,23,C5,20,6,7E,4F,78,CD,98,42,C1,CB,39,
10,EF,11,0,4,1B,7A,B3,20,FB,C3,8,F0
20 FORA=&HF000TO&HF054:READA$:POKEA,VAL(
"&h"+A$):NEXT
30 SCREEN 2,2:COLOR1,1,,1:CLS:MAG3
40 PATTERNS#0,"7FFFFFFFFFFFFFFFFFFFFF"
50 PATTERNS#1,"FFFFFFFFFFFFFFFFFFFF7F"
60 PATTERNS#2,"FEFFFFFFFFFFFFFFFFFFFF"
70 PATTERNS#3,"FFFFFFFFFFFFFFFFFFFFFE"
80 X=1:FORA=1TO2:FORB=1TO4:SPRITEX,(45*B
,A*50),0,0:X=X+1:NEXTB,A
90 CALL&HF008
```

Disc Modifications

```
10 DATA2,4,6,7,A,B,D,C,6,8,E,0,DB,DD,FE,
FF,20,2,CB,C1,CB,21,11,0,4,1B,7A,B3,20,F
B,10,EC,DB,DD,FE,FF,20,2,CB,C1,C5,6,8,78
,E,0,CD,AE,0,10,F8,C1,6,8,21,FF,EF,CB,41
,23,C5,20,6,7E,4F,78,CD,AE,0,C1,CB,39,10
,EF,11,0,4,1B,7A,B3,20,FB,C3,8,F0
```

Machine Code Routines

Once again, here is the next exciting instalment in the "Machine Code for those who want to know" series.

This time around I'll give you a whole host of Machine Code routines. They are all fairly self explanetary and should pose no real problem to those amongst you who wish to get into machine code.

Note that the only version you need be interested in are the ROM:PAL ver 1.0 and the Disk ver 1.0p. Note the latter is for those with the SF-7000 disc drive.

Also note that Michael Howard is working on a really easy to understand book on machine code for the Sega, and this will be available shortly. So keep your eyes peeled and watch this space!

Text screen display

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: DSPTX
(2) ENTRY POINT: 39E5H (ROM:JAP ver 1.1)
39E5H (ROM:USA ver 1.0a)
39E5H (ROM:PAL ver 1.0)
0084H (Disk ver 1.0p)
0084H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To display the text screen.
(4) INPUT DATA: None
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Graphics screen display

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: DSPGRP
(2) ENTRY POINT: 39E2H (ROM:JAP ver 1.1)
39E2H (ROM:USA ver 1.0a)
39E2H (ROM:PAL ver 1.0)
0087H (Disk ver 1.0p)
0087H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To display the graphics display.
(4) INPUT DATA: None
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Text screen back colour set

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: TXTBCL
(2) ENTRY POINT: 39D0H (ROM:JAP ver 1.1)
39D0H (ROM:USA ver 1.0a)
39D0H (ROM:PAL ver 1.0)
008AH (Disk ver 1.0p)
008AH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set a back colour of the text screen.
(4) INPUT DATA: A = Colour code (0 — 15)
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Text screen character colour set

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: TXTCCL
(2) ENTRY POINT: 39D3H (ROM:JAP ver 1.1)
39D3H (ROM:USA ver 1.0a)
39D3H (ROM:PAL ver 1.0)
008DH (Disk ver 1.0p)
008DH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set a character colour of the text screen.
(4) INPUT DATA: A = Colour code (0 — 15)
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Graphics screen "0" colour set

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: GRPCCB
(2) ENTRY POINT: 39D6H (ROM:JAP ver 1.1)
39D6H (ROM:USA ver 1.0a)
39D6H (ROM:PAL ver 1.0)
0090H (Disk ver 1.0p)
0090H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set a back ("0") colour of the graphics screen.
(4) INPUT DATA: A = Colour code (0 — 15)
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Graphics screen "1" colour set

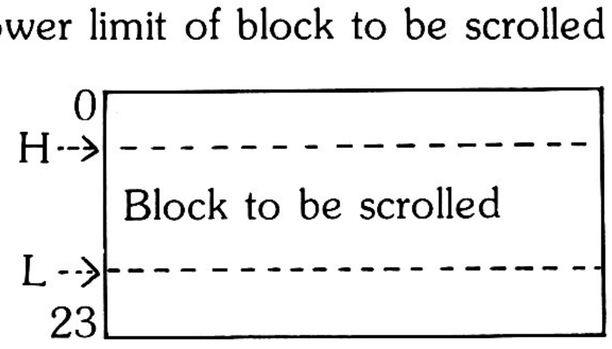
(Common to all ROM and Disk versions)

(1) NAME OF LABEL: GRPCCL
(2) ENTRY POINT: 39D9H (ROM:JAP ver 1.1)
39D9H (ROM:USA ver 1.0a)
39D9H (ROM:PAL ver 1.0)
0093H (Disk ver 1.0p)
0093H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set the colour of character, dots or lines ("1") of the graphics screen.
(4) INPUT DATA: A = Colour code (0 — 15)
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Text screen scroll

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: SCRTXT
(2) ENTRY POINT: 39F4H (ROM:JAP ver 1.1)
39F4H (ROM:USA ver 1.0a)
39F4H (ROM:PAL ver 1.0)
0096H (Disk ver 1.0p)
0096H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To scroll a specified block of the text screen up by one line.
(4) INPUT DATA:
A = Upper limit of block to be scrolled
L = Lower limit of block to be scrolled

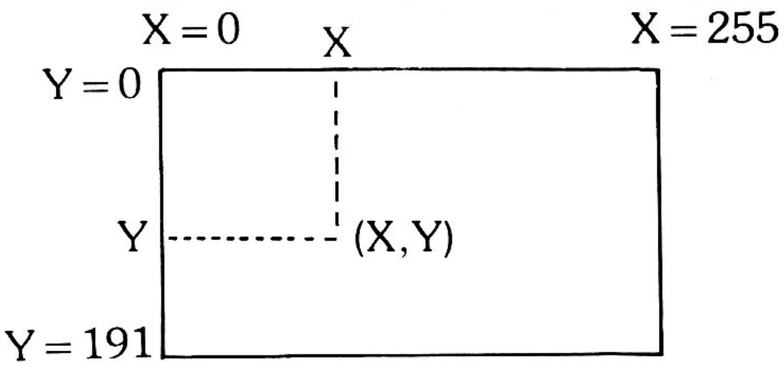


(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Dot plot in graphics screen

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: PLOTPT
(2) ENTRY POINT: 39EEH (ROM:JAP ver 1.1)
39EEH (ROM:USA ver 1.0a)
39EEH (ROM:PAL ver 1.0)
0099H (Disk ver 1.0p)
0099H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To plot or erase a dot in the graphics screen.
(4) INPUT DATA:
D = Y (vertical) coordinate (0 — 191)
E = X (horizontal) coordinate (0 — 255)
A = (0:Erase)
(1:Plot)



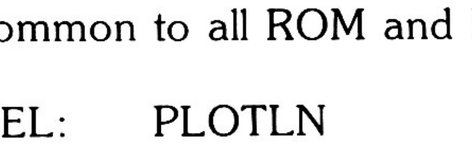
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Line draw/erase in graphics screen

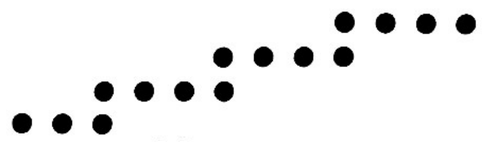
(Common to all ROM and Disk versions)

(1) NAME OF LABEL: PLOTLN
(2) ENTRY POINT: 39FlH (ROM:JAP ver 1.1)
39FlH (ROM:USA ver 1.0a)
39FlH (ROM:PAL ver 1.0)
009CH (Disk ver 1.0p)
009CH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To draw or erase a line in the graphics screen.
(4) INPUT DATA:
D = Start Y (vertical) coordinate 0 — 191
E = Start X (horizontal) coordinate 0 — 255
H = End Y (vertical) coordinate 0 — 191
L = End X (horizontal) coordinate 0 — 255
A = (= 0:Erase)
(<>0:Draw)
B = Mode (= 0:Dot connection)
(<>0:Side connection)

(Dot connection)
Used when drawing a line with the LINE statement.



(Side connection)
Used when painting the enclosed region with the BF specification in the CIRCLE or BCIRCLE statement.



(5) OUTPUT DATA: None
(6) CALLING SEQUENCE: Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Sprite colour set

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: SSPRCL
(2) ENTRY POINT: 42D8H (ROM:JAP ver 1.1)
42D8H (ROM:USA ver 1.0a)
42D8H (ROM:PAL ver 1.0)
00AEH (Disk ver 1.0p)
00AEH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set a sprite colour for a specified surface.
(4) INPUT DATA: A = Plane No. (0 — 31)
C = Colour code (0 — 15)
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction

(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Sprite name set

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: SSPRNA
(2) ENTRY POINT: 42D5H (ROM:JAP ver 1.1)
42D5H (ROM:USA ver 1.0a)
4295H (ROM:PAL ver 1.0)
00BlH (Disk ver 1.0p)
00BlH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set a sprite name for a specified surface.
(4) INPUT DATA: A = Plane No. (0 — 31)
C = Sprite name (0 — 255)
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Sprite position set

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: SSPRPS
(2) ENTRY POINT: 42D2H (ROM:JAP ver 1.1)
42D2H (ROM:USA ver 1.0a)
42D2H (ROM:PAL ver 1.0)
00B4H (Disk ver 1.0p)
00B4H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set a character colour of the text screen.
(4) INPUT DATA:
A = Plane No. (0 — 31)
D = Y (vertical) position 0 — 191
E = X (horizontal) position 0 — 255
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Text screen clear

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: TXTCLR
(2) ENTRY POINT: 430BH (ROM:JAP ver 1.1)
430BH (ROM:USA ver 1.0a)
430BH (ROM:PAL ver 1.0)
00BAH (Disk ver 1.0p)
00BAH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To clear the text screen.
(4) INPUT DATA: None
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Graphics screen 8-bit plot

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: PLOTB8
(2) ENTRY POINT: 4A27H (ROM:JAP ver 1.1)
4A27H (ROM:USA ver 1.0a)
4A27H (ROM:PAL ver 1.0)
00EDH (Disk ver 1.0p)
00EDH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To plot a 1-byte (8-bit) pattern in the graphics screen.
(4) INPUT DATA:
D = Y (vertical) position 0 — 191
E = X (horizontal) position 0 — 255
A = Pattern

(x,y) ○○●●○○○●

(When A is 31H)
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Graphics screen character plot

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: PLOTCH
(2) ENTRY POINT: 4A2DH (ROM:JAP ver 1.1)
4A2DH (ROM:USA ver 1.0a)
4A2DH (ROM:PAL ver 1.0)
00F3H (Disk ver 1.0p)
00F3H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To plot a character pattern in the graphics screen.
(4) INPUT DATA:
D = Y (vertical) coordinate 0 — 191
E = X (horizontal) coordinate 0 — 255
A = Character code
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Graphics screen character size set

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: GRPCSZ
(2) ENTRY POINT: 4A30H (ROM:JAP ver 1.1)
4A30H (ROM:USA ver 1.0a)
4A30H (ROM:PAL ver 1.0)
00F6H (Disk ver 1.0p)
00F6H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set a size of character to be displayed in the graphics screen.
(4) INPUT DATA:
A = (0:Standard size)

(1:Double size)
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Graphics screen cursor position read

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: GRPCRD
(2) ENTRY POINT: 4A33H (ROM:JAP ver 1.1)
4A33H (ROM:USA ver 1.0a)
4A33H (ROM:PAL ver 1.0)
00F9H (Disk ver 1.0p)
00F9H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To read the current cursor position in the graphics screen.
(4) INPUT DATA: None
(5) OUTPUT DATA:
A = Y (vertical) position 0 — 191
E = X (horizontal) position 0 — 255
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: D, E
(8) FLAG CHANGES: None

Graphics screen cursor position set

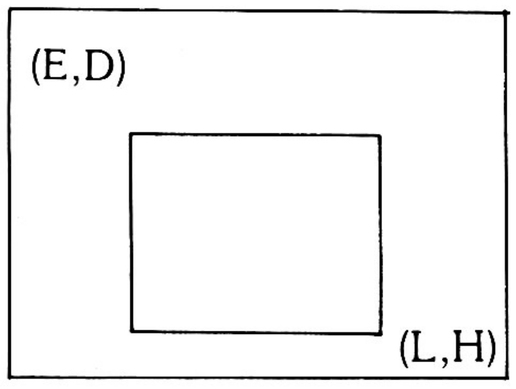
(Common to all ROM and Disk versions)

(1) NAME OF LABEL: CRPCWT
(2) ENTRY POINT: 4A36H (ROM:JAP ver 1.1)
4A36H (ROM:USA ver 1.0a)
4A36H (ROM:PAL ver 1.0)
00FCH (Disk ver 1.0p)
00FCH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set a cursor position in the graphics screen.
(4) INPUT DATA:
A = Y (vertical) position 0 — 191
E = X (horizontal) position 0 — 255
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Graphics screen back-colour set

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: GRPBCL
(2) ENTRY POINT: 4A3FH (ROM:JAP ver 1.1)
4A3FH (ROM:USA ver 1.0a)
4A34H (ROM:PAL ver 1.0)
0105H (Disk ver 1.0p)
0105H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To set a black colour for a specified block of graphics screen.
(4) INPUT DATA:
A = Colour code
D = Vertical position of upper-left corner 0 — 191
E = Horizontal position of upper-left corner 0 — 255
H = Vertical position of lower-right corner 0 — 191
L = Horizontal position of lower-right corner 0 — 255



(5) OUTPUT DATA: None
(6) CALLING SEQUENCE: Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Graphics screen clear

(Common to all ROM and Disk versions)

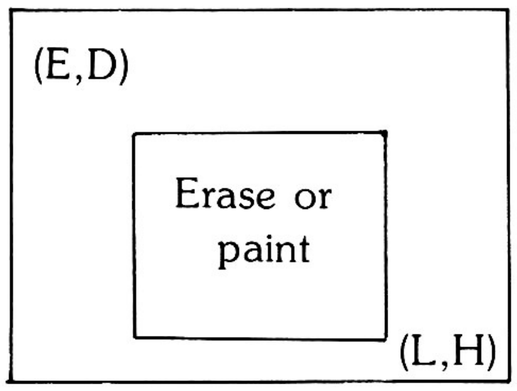
(1) NAME OF LABEL: GRPCLR
(2) ENTRY POINT: 4A42H (ROM:JAP ver 1.1)
4A42H (ROM:USA ver 1.0a)
4A42H (ROM:PAL ver 1.0)
0108H (Disk ver 1.0p)
0108H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To clear the graphics screen.
(4) INPUT DATA: None
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Graphics screen block erase/paint

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: GRPERS (Block erase)
GRPBFS (Block paint)
(2) ENTRY POINT:
(GRPERS) 4A4BH (ROM:JAP ver 1.1)
4A4BH (ROM:USA ver 1.0a)
4A4BH (ROM:PAL ver 1.0)
010EH (Disk ver 1.0p)
010EH (Disk ver 1.0j)

(GRPBFS) 4A96H (ROM:JAP ver 1.1)
4A96H (ROM:USA ver 1.0a)
4A96H (ROM:PAL ver 1.0)
4E72H (Disk ver 1.0p)
4A72H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To erase or paint a block of graphics screen.
(4) INPUT DATA:
D = Vertical position of upper-left corner 0 — 191
E = Horizontal position of lower-right corner 0 — 191
H = Vertical position of lower-right corner 0 — 191
L = Horizontal position of lower-right corner 0 — 255



(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Graphics screen back drop colour set

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: GRPDCL
(2) ENTRY POINT: 4A4EH (ROM:JAP ver 1.1)
4A4EH (ROM:USA ver 1.0a)
4A4EH (ROM:PAL ver 1.0)
0111H (Disk ver 1.0p)
0111H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
In the graphics screen, to set a colour of back drop.
(4) INPUT DATA: A = Colour code
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE: Called by CALL instruction
(7) REGISTER CHANGES: None
(8) FLAG CHANGES: None

Graphics screen point read

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: READPT
(2) ENTRY POINT: 4A5lH (ROM:JAP ver 1.1)
4A5lH (ROM:USA ver 1.0a)
4A5lH (ROM:PAL ver 1.0)
0114H (Disk ver 1.0p)
0114H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To read the pattern generator table for a specified point in the graphics screen.
(4) INPUT DATA:
D = Vertical position 0 — 191
E = Horizontal position 0 — 255
(5) OUTPUT DATA: A = (0:bit "0")
(1:bit "1")
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Listout

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: PUTLST
(2) ENTRY POINT: 4A8AH (ROM:JAP ver 1.1)
4A8AH (ROM:USA ver 1.0a)
4A8AH (ROM:PAL ver 1.0)
012FH (Disk ver 1.0p)
012FH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To list out text contents.
(4) INPUT DATA: DE = Start line No.
HL = End line No.
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

One-character key entry
(for INKEYS function)

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: INKEYS
(2) ENTRY POINT: 42F6H (ROM:JAP ver 1.1)
42F6H (ROM:USA ver 1.0a)
42F6H (ROM:PAL ver 1.0)
0168H (Disk ver 1.0p)
0168H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To receive one character of key entry.
(4) INPUT DATA: None
(5) OUTPUT DATA:
A = Character code (00H for no entry)
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

Beep output

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: BEEPOT
(2) ENTRY POINT: 4A18H (ROM:JAP ver 1.1)
4A18H (ROM:USA ver 1.0a)
4A18H (ROM:PAL ver 1.0)
016BH (Disk ver 1.0p)
016BH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To output a beep.
(4) INPUT DATA: A = Control
(=0) Turn beep off
(=1) Turn 1760-Hz on
(=2) Turn 1760-Hz on for 125 msec, then turn it off
(=3) Turn on consecutive beeps of 880 Hz, 1760 Hz, 880 Hz and 1760-Hz, each for 62 msec, then turn beep off
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER
CHANGES: None
(8) FLAG CHANGES: None

One-line key entry

(Common to al ROM Disk versions)

(1) NAME OF LABEL: INKEYL
(2) ENTRY POINT: 4A48H (ROM:JAP ver 1.1)
4A48H (ROM:USA ver 1.0a)
4A48H (ROM:PAL ver 1.0)
016EH (Disk ver 1.0p)°
016EH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To receive one line of key entry.
(4) INPUT DATA: None
(5) OUTPUT DATA: [KEYBUF] 256 = Key entry image (oDH for final code)
DE = KEYBUF (beginning address)
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES:DE
(8) FLAG CHANGES: None

Cassette tape input scan

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: WVSCAN
(2) ENTRY POINT: 39FDH (ROM:JAP ver 1.1)
39FDH (ROM:USA ver 1.0a)
39FDH (ROM:PAL ver 1.0)
0171H (Disk ver 1.0p)
0171H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To count the time elapsed until the input waveform from cassette tape changes.
(4) INPUT DATA: D = Previous waveform input (MSB)
C = Counter value
(5) OUTPUT DATA: D = Current waveform input
C = Counter value
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES:A, D, C
(8) FLAG CHANGES: All

Cassette tape leader field find

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: LEADSR
(2) ENTRY POINT: 3A00H (ROM:JAP ver 1.1)
3A00H (ROM:USA ver 1.0a)
3A00H (ROM:PAL ver 1.0)
0174H (Disk ver 1.0p)
0174H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To find the leader field recorded on cassette tape. Upon finding it, control returns from this routine.
(4) INPUT DATA: None
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES:A
(8) FLAG CHANGES: All

Cassette tape input/output time delay

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: CDELAY
(2) ENTRY POINT: 3A03H (ROM:JAP ver 1.1)
3A03H (ROM:USA ver 1.0a)
3A03H (ROM:PAL ver 1.0)
0177H (Disk ver 1.0p)
0177H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To delay the time of cassette tape input/output
(4) INPUT DATA: C = Delay time
Delay time = 53 + (14*C) states
State = 279.365 nsec
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES:None
(8) FLAG CHANGES: All

One byte read from cassette tape

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: BYTERD
(2) ENTRY POINT: 3A06H (ROM:JAP ver 1.1)
3Ao6H (ROM:USA ver 1.0a)
3A06H (ROM:PAL ver 1.0)
017AH (Disk ver 1.0p)
017AH (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To read one byte of data recorded on cassette tape.
(4) INPUT DATA: None
(5) OUTPUT DATA: A = Read data
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES:A
(8) FLAG CHANGES: All

One bit write to cassette tape

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: ONE OUT ("1" output)
ZEROUT ("0" output)
(2) ENTRY POINT: ONEOUT
3A09H (ROM:JAP ver 1.1)
3A09H (ROM:USA ver 1.0a)
3A09H (ROM:PAL ver 1.0)
017DH (Disk ver 1.0p)
017DH (Disk ver 1.0j)
ZEROUT
3A0CH (ROM:JAP ver 1.1)
3A0CH (ROM:USA ver 1.0a)
3A0Ch (ROM:PAL ver 1.0)
0180H (Disk ver 1.0p)
0180H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To record one bit on cassette tape.
(4) INPUT DATA: C = Delay time
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES:A, C
(8) FLAG CHANGES: All

Leader field write to cassette tape

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: LEADWR
(2) ENTRY POINT: 3A0FH (ROM:JAP ver 1.1)
3A0FH (ROM:USA ver 1.0a)
3A0FH (ROM:PAL ver 1.0)
0183H (Disk ver 1.0p)
0183H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To write a leader field onto cassette tape.
(4) INPUT DATA: None
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES:None
(8) FLAG CHANGES: None

One byte write to cassette tape

(Common to all ROM and Disk versions)

(1) NAME OF LABEL: BYTEWR
(2) ENTRY POINT: 3A12H (ROM:JAP ver 1.1)
3A12H (ROM:USA ver 1.0a)
3A12H (ROM:PAL ver 1.0)
0186H (Disk ver 1.0p)
0186H (Disk ver 1.0j)
(3) PURPOSE OF THIS PROGRAM:
To write one byte of data onto cassette tape.
(4) INPUT DATA: A = Data to be written
C = Delay
(5) OUTPUT DATA: None
(6) CALLING SEQUENCE:
Called by CALL instruction
(7) REGISTER CHANGES:None
(8) FLAG CHANGES: None
(9) SETTING TIME DELAY PARAMETER C:

The value of parameter C to be inputted is determined as follows:

[For input after call of LEADWR routine]
C = (1168 — Sl)/14
Where Sl denotes the number of statements executed after the LEADWR routine is called, excepting the CALL instruction execute stage.

[For input after call of BYTEWR routine]
C = (1203 — S2)/14
Where S2 denotes the number of statements executed after the BYTEWR routine is called, excepting the CALL instruction execute state.

```
CALL    LEADWR
          :
        C, --
CALL    BYTEWR
          :
LD      C, --
CALL    BYTEWR
          :
```

S1 state

S2 state

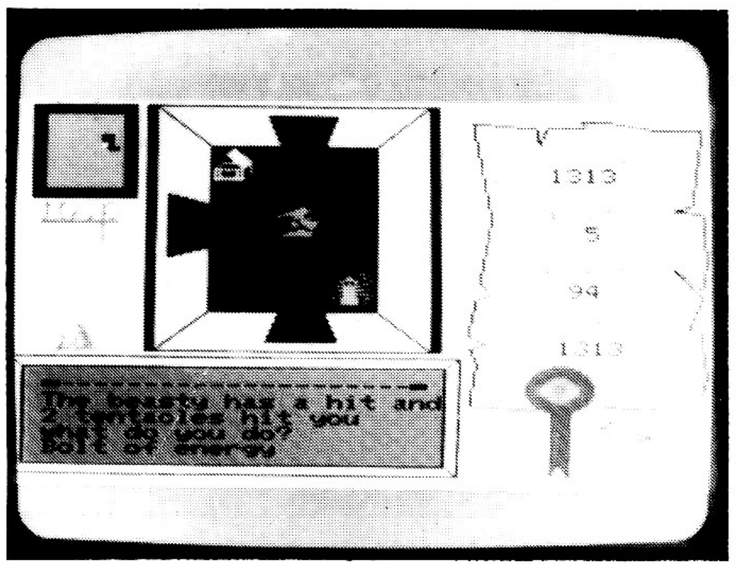
REVIEW

The House! — disk only

At last, the classic machine code graphic adventure game is available on disc! You take the part of the intrepid adventurer, Siddy Spuerspook, who is out to explore nearly 300 rooms of The House, and hopefully murder Vanessa Ze Vampire in the meantime! But as with many games, this is not as easy as it seems, as Vanessa has many allies, called yes you guessed it, Purple People Eaters! The worst things since mouldy bread, these monsters are quite good at murdering people for no apparent reason! They are just very violent in the extreme!

The object of the game, as already mentioned is to try and get hold of Vanessa Ze Vampire, and hopefully killing her, but this is only possible after you have found a crucifix, and The House has only one!!!

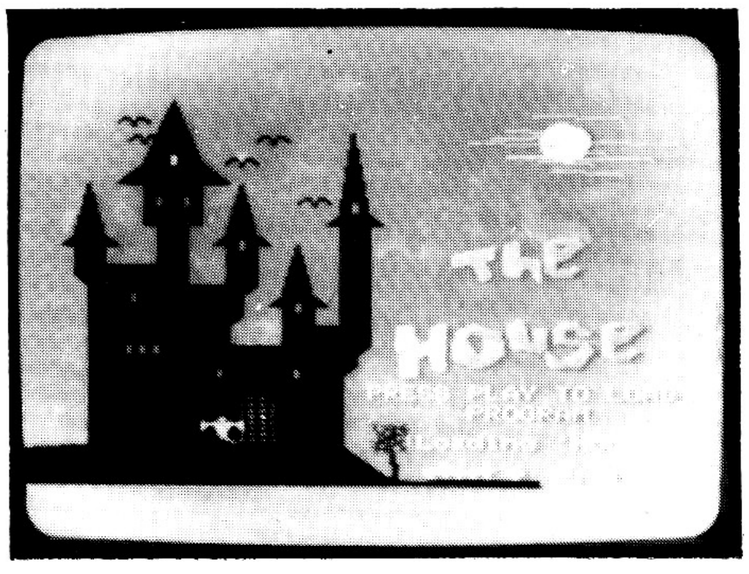
The game is quite complex and is fun to play, and is streaked with a mean sense of humour . . . If you have a disc drive . . . then buy this game!!



It is available for $39.95 on disc from:

User Tronics Developments Ltd
P.O. Box 29124
Greenwoods Corner
Epsom
Auckland 3

Please make cheques/MO payable to M. Howard



Hill St. Blues the theme music

If, like me, you don't really like Hill St. Blues, but love the music, then this program is for you...

To use it, just type it in and then type RUN . . . and just sit back and listen to the brilliant rendition of the music we all know so well.

```
10 CLS:CURSOR 8,10:PRINT "Hill Street Blues"
20 V=8:M=1
30 FOR I=1 TO 24
40 READ T1,T2,T3,DE
50 SOUND 1,T1,V:SOUND 2,T2,V:SOUND 3,T3,V:FOR D=1 TO DE:NEXT D
60 NEXT I
70 RESTORE 210
80 FOR I=1 TO 57
90 READ T1,T2,T3,DE
100 SOUND 1,T1,V:SOUND 2,T2,V:SOUND 3,T3,V:FOR D=1 TO DE:NEXT D
110 NEXT I
120 IF M=2 THEN GOTO 340
130 M=2:GOTO 70
140 DATA 392,466,622,100,349,466,587,50,392,466,622,160,20000,20000,20000,10
150 DATA 392,466,622,100,349,466,587,50,392,466,622,160,20000,20000,20000,10
160 DATA 392,466,622,100,349,466,587,50,392,466,622,120
170 DATA 349,20000,587,120,349,20000,523,120,349,20000,587,190
180 DATA 196,262,311,80,175,233,294,60,156,208,262,200
190 DATA 262,311,392,80,233,294,349,60,208,262,311,200
200 DATA 311,392,523,80,294,349,466,60,262,311,415,200,208,262,311,200
210 DATA 392,466,622,100,349,466,587,50,392,466,622,160,20000,20000,20000,10
220 DATA 392,466,622,100,349,466,587,50,392,466,622,160,20000,20000,20000,10
230 DATA 392,466,622,100,349,466,587,50,392,466,622,100
240 DATA 349,20000,587,40,349,20000,523,40,349,20000,587,40
250 DATA 392,466,622,100,349,466,587,50,392,466,622,160,20000,20000,20000,10
260 DATA 523,622,784,100,20000,20000,698,50,415,523,622,90,20000,20000,587,50
270 DATA 20000,415,523,40,20000,20000,587,40,392,466,622,100,349,466,587,50
280 DATA 392,466,622,200
290 DATA 349,440,523,100,20000,20000,587,50,349,523,622,100,20000,20000,523,40,2000,20000,587,40,20000,20000,622,40
300 DATA 466,622,784,100,20000,20000,698,40,494,622,784,75,20000,20000,698,40,20000,622,40,20000,20000,698,40
310 DATA 659,784,1047,100,20000,20000,932,50,554,659,830,75,20000,20000,554,10,2000,20000,2000,2,20000,20000,554,10,20000,659,10
320 DATA 20000,20000,784,30,20000,740,880,85,20000,659,784,85,20000,587,740,180
330 DATA 196,233,294,80,196,262,311,60,1
```

```
96,294,349,160,20000,2000,20000,2,208,2
62,311,80,208,311,392,60,208,294,349,160
340 RESTORE 430
350 FOR I=1 TO 4
360 READ T1,T2,T3,DE
370 SOUND 1,T1,V:SOUND 2,T2,V:SOUND 3,T3
,V:FOR D=1 TO DE:NEXT D
380 NEXT I
390 FOR I=8 TO 0 STEP -1
400 SOUND 1,194,I:SOUND 2,233,I:SOUND 3,
311,I:FOR D=1 TO 75:NEXT D
410 NEXT I
420 END
430 DATA 194,233,311,100,175,233,294,60,
196,233,311,140,175,233,294,60
```

Every once in a while you may wish to sort some data into alphabetical or numerical order. Well that's fine, but the major problem is that many people use a method known as "Bubble Sorting" . . . and the problem with bubble sorting is that it is exceedingly slow. This program called "Quicksort" is just that . . . quick!!! I'm not going to explain how it workds because it is quite complex . . . just take it from me . . . it works and works very well!!

NOTES

1) Lines 10-70 set up an array of random numbers. Call the routine at line 1000 onwards then print out the sorted array.
2) Data in the array starts at location 1 ie; no 0
   eg; a(1),a(2),a(3) . . . a(30)
   NOT a(0),a(1),a(3) . . . a(30)
3) The variable N holds the number of data elements to be sorted in the array. In this case it is set to 30.
4) The actual quicksort starts at line 1000.
5) The speed of a bubble sort, sorting 50 random numbers is 1 min 16 seconds. The speed of a quicksort, sorting 50 random numbers is 12 seconds!!!!!

```
10 DIM A(30)
20 FOR C=1 TO 30:A(C)=INT(RND(8)*1000):N
EXT C
30 N=30:GOSUB 1000
40 FOR C=1 TO 30
50 PRINTA(C)
60 NEXT
70 GOTO 70
1000 M=40:DIMSL(M),SR(M)
1010 S=1:SL(1)=1:SR(1)=N
1020 L=SL(S):R=SR(S):S=S-1
1030 I=L:J=R:X=A(INT((L+R)/2))
1040 IFA(I)<XTHENI=I+1:GOTO1040
1050 IFX<A(J)THENJ=J-1:GOTO1050
1060 IFI>JTHEN1080
1070 W=A(I):A(I)=A(J):A(J)=W:I=I+1:J=J-1
1080 IFI<=JTHEN1040
1090 IFI>=RTHEN1110
1100 S=S+1:SL(S)=I:SR(S)=R
1110 R=J
1120 IFL<RTHEN1030
1130 IFS<>0THEN1020
1140 RETURN
```


TEACH
YOURSELF
BASIC
PROGRAMMING
FOR USE WITH
16 OR 32K SEGA
STILL
AVAILABLE
SEGA®
Beginners Guide
Practical advice on all aspects of using the Sega Computer Peripheral attachments, and software takes the frustration out of getting to know your SEGA COMPUTER.
NASA
United States
USA
GRANDSTAND
SEGA®