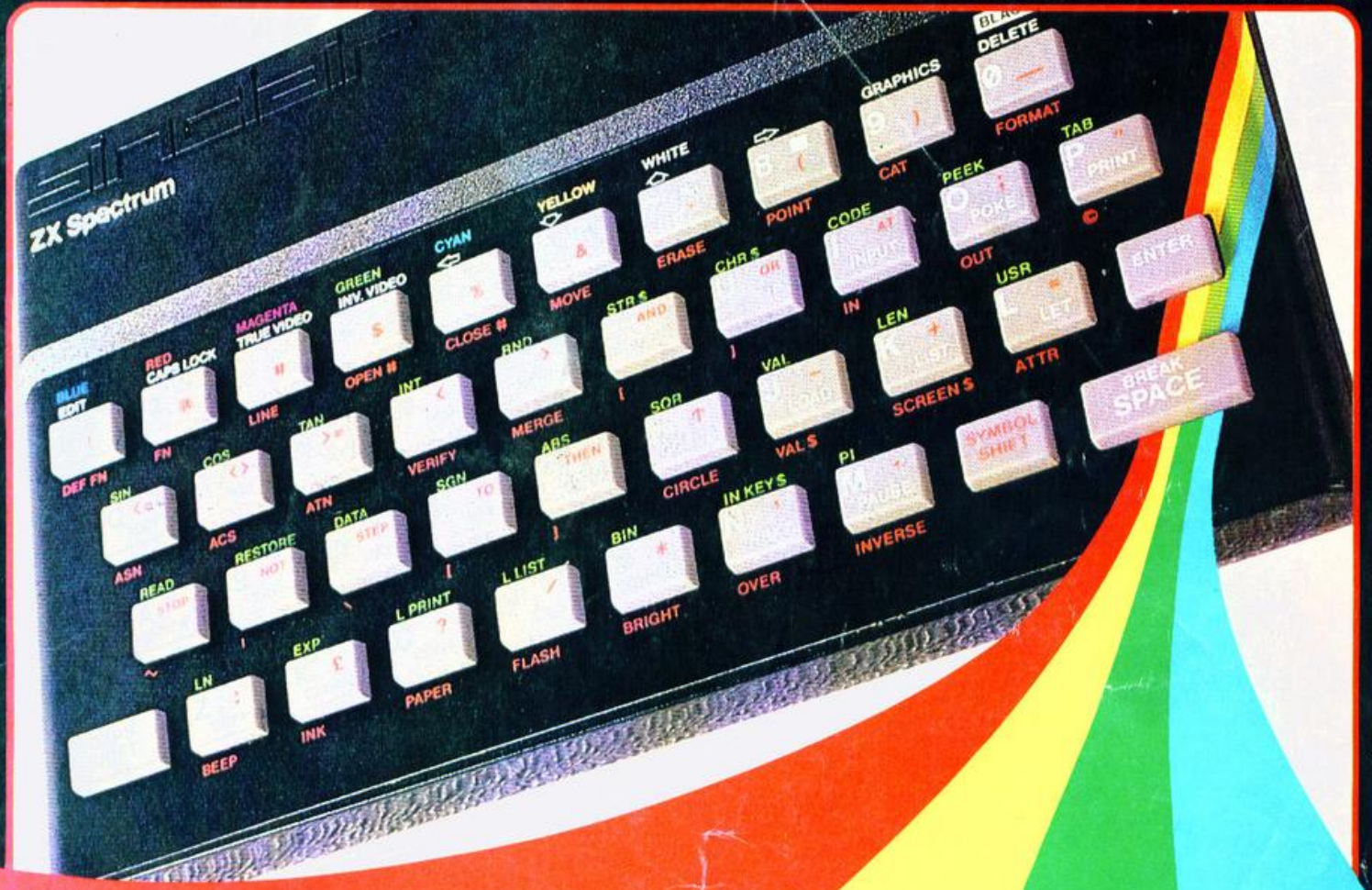
Autumn 1985

An Argus Specialist Publication

Personal SOFTWARE

Volume 4 Number 3 £2.25



Somewhere, Over the Rainbow – Give Your Spectrum Its Wings With Our Graphic Utilities

Not Turtles Again! An All-In-One Logo Program

Zipping and Zapping With Short Routines

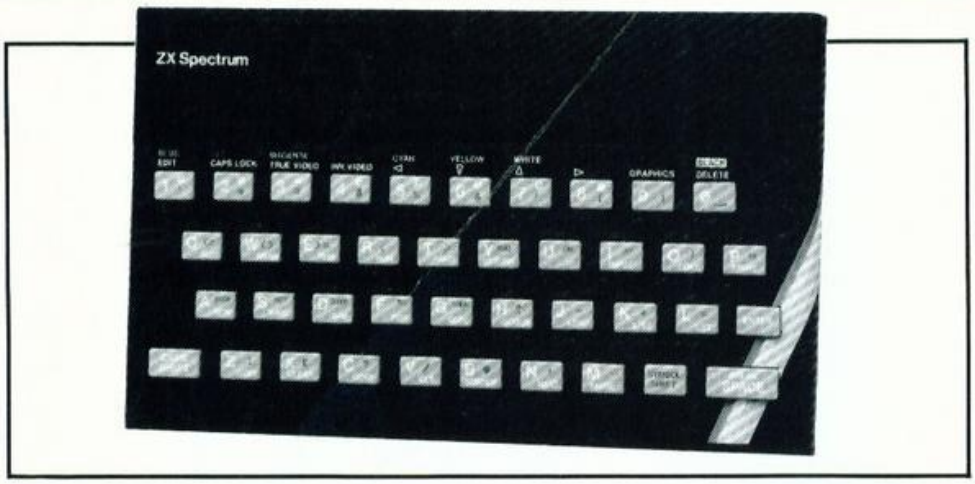
Compose Yourselves With Great Music Features

PLUS The Best Educational And Fun Spectrum Listings

Can You Handle The Galactoids?

Peeking about

Voyeurs will love this chance to PEEK into memory to sort out screen character positions.



The Spectrum is an interesting machine in many ways, but it has one characteristic which can be infuriating. Where a memory mapped display allows the character in a given screen position to be determined by a simple PEEK into memory, the Spectrum display store is arranged in what appears, at first sight, to be a totally chaotic manner, and PEEKs provide no useful information. The manual warns the user of this, and suggests the use of SCREEN$ as an alternative, but that is no use with user-defined graphics, which are seen as spaces.

A little patient experiment shows that the eight bytes forming the character at the top left hand corner of the screen are held in locations:

16384
16640
16896
17152
17408
17664
17920
18176

16384 is the start of the display memory area; and the other addresses are obtained by adding 256 successively. Ah! So we can write:

Address = 16384 + 256*(n−1) + 32*x + y = 16128 + 256*n + 32*x + y

Where the address contains the nth byte of the character in line x column y.

UNBLOCKING THE SCREEN

But this fails after eight screen lines have been examined. An entirely new block starts, 2048 locations after the first, and we have:

Address = 18176 + 256*n + 32*(n−8) + y = 17920 + 256*n + 32*x + y

After a total of sixteen lines, a third block starts, and we have:

Address = 20224 + 256*n + 23*(x−16) + y = 19712 + 256*n + 32*x + y.

AMUSING AND USEFUL

Input a number, and the screen fills with characters. Then a scan starts, in which the eighth byte of each character is compared with the number which was input. If the two are equal, the character is converted to reverse video by PRINT OVER. Otherwise, the character remains unaltered. Input zero, and most of the characters will change. Those which do not are characters with descenders, with at least one high bit in the last byte. Input 52, and p and q will be picked out. To save tedious experiment, the numbers worth trying are 0,6,8,16,24,32,56,60, 64 and 255...

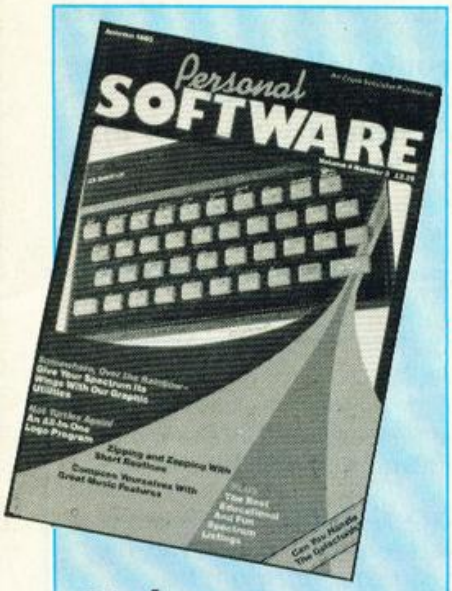
The second input allows you to state a colour and then another check number, so that you can put a colour against a specific last byte. Not a particularly useful program, but quite amusing.

Now create some user graphics and extend the original program so that it prints them, by changing the 95 entries to 132. If you use a bit of craft, you can make the lowest line of each special graphic give a different number, preferably not one in the list given above. The program will then pick out this number, identifying each graphic individually.

It need scarcely be said that the byte examined need not be the last one forming the character. If the graphics make a different byte more convenient, all that needs doing is a change in BASE value.

At a guess, the SCREEN$ function may compare the bytes which it reads from the character with the character pattern data stored near the top of the ROM. It would be possible to perform the same trick with the special graphics data, but that would probably be an overkill. However, there may be cases which would make that worth while, though machine code would be necessary to keep the response time down.

```
10   INPUT N
20   LET A=0
30   LET B=INT (A/95):LET C=A-95*B
40   PRINT CHR$(C+32);
50   IF A<703 THEN LET A=A+1:GOTO 30
60   INK 2
70   FOR X=0 TO 21
80   FOR Y=0 TO 31
90   LET BASE=18176
100  IF X>7 THEN LET BASE=19968
110  IF X>15 THEN LET BASE=21760
120  LET D=(BASE+32*X+Y)
130  IF PEEK (D)=N THEN PRINT OVER 1; AT X,Y; "RV
     SP"
140  NEXT Y:NEXT X
150  INPUT M:INK M
160  INPUT N
170  GOTO 70
```



Volume 4
No 3
Autumn 1985

Editor: Dave Reeder
Group Editor: Dave Bradshaw
Advertisement Manager: Sarah Musgrave
Advertisement Copy Control: Lynn Collis
Publishing Director: Peter Welham
Design by Argus Design

Personal Software is a quarterly magazine appearing on the third Friday in May, August, November and February.

Distribution by: Argus Press Sales & Distribution Ltd, 12-18 Paul Street, London EC2A 4JS Tel: 01-247 8233. Printing by: Alabaster Passmore & Sons Ltd, Tovil, Maidstone, Kent.

Subscription Rates : UK £10.60 for four issues including postage. Airmail and other rates upon application to Personal Software Subscriptions, Infonet Ltd, Times House, 179 The Marlowes, Hemel Hempstead, Herts HP1 1BB.

CONTENTS

Editorial & Advertisement Office
No 1 Golden Square, London W1R 3AB
Telephone: 01-437 0626 Telex: 8811896

GREG TURNBULL

IQ TEST

If people tell you computing makes you dull and introverted, try this test to see if they are right!

This is a program for the 48K Spectrum. It contains a fifty question intelligence test and two short personality tests. The IQ test contains both verbal questions and mathematical ones. The maths questions can be worked out on paper (without the use of a calculator) and the results entered when prompted by the computer.

YOUR STARTER FOR FIVE

The listing may look very long, but don't be put off from typing it in, as many of the lines are very similar and can be edited out, altered and re-entered with a new line number. The tests can be seen in an example screen before use. However, the main IQ test is timed (via a real-time clock which is updated periodically), so you have a maximum of thirty minutes to complete the test.

You may pass on any question, by entering 0 to the answer(s). At the start of each loop you are requested to enter the number of the question that you wish to attempt. This allows you to return to any question you may have passed on. Please note that if you DO attempt a question, whether you get it right or wrong, you will not be allowed to return to it later. You are only allowed one attempt at each question.

SLOW BUT SURE

Take your time with the test and answer each question carefully, but don't spend too much time over a question you may be stuck on. At the end of the test, if the time is up, or you request question 0, or if all of the questions have been tried, then your score, your IQ rating, and some comments will be printed on the screen. You can then try to answer questions in the personality tests. These questions have no time limit, and you only have to enter A or B in answer to them. Don't think too hard about the questions, try to answer them instinctively. At the end of each of these questions a comment is made based on your results.

HOW IT RUNS

2	*POKE keyboard response time/sound. Define names variables for program size and free memory (in K).*
3-9	*Credits*
11-17	*Initialise screen picture (IQ Test). Each line entered as a string. The whole picture is then defined as the sum of these (F$). String is then printed on the screen.*
20-23	*Load machine background noise.*
25-30	*Introduction to program. The message is stored as Y$, then string-splitting is used to sequentially print this on the screen + background machine code noise & beeps. The speed is changeable by altering PAUSE 3.*
35-40	*Menu of choices. Input and go to line number.*
50-55	*Continue subroutine.*
60-65	*Start test subroutine.*
70-75	*Choose question number subroutine.*
80-86	*Check to see if question has already been answered.*
90-96	*Prevent a repeat of the same question.*
100-190	*Update the real-time clock (based on the FRAMES system variable), and print the time on the screen (00.00 to 30.00 minutes/seconds).*
200-250	*Initialise variables eg IQ,SCORE,MIN, SEC etc... plus start clock and test.*
300-360	*Double number answer. (X,Y)*
400-420	*Single number answer (X)*
500-520	*Single word answer.*
600-640	*Personality test (P.TEST) no. 1. Answer and score (A$,PT1A)*
650-690	*Personality test (P.TEST) no.2. Answer and score (A$,PT2B)*
700-730	*Load UDG. Graphic "A" = " "*
800-815	*P.TEST no.1 results/comments.*
900-915	*P.TEST no.2 results/comments.*
1000-1140	*IQ Test. Introductory message (Y$), as above. Start test, choose question number, update clock, check to see if question already answered, go to question line number (1000 + (16 * Question number)). The main question loop.*
1116-1910	*The 50 IQ test questions. Print question, GOSUB for an answer, check to see if question was passed on (answer = 0), remove question from screen, return to main loop (line 1040).*
1950-1990	*Time-up/test completed, GOSUB for results, return to menu.*
2000-2310	*P.TEST no.1. Set-up screen, and print intro message (Y$). Start test, print questions in sequence, check each answer, increase score. End test, GOSUB for result, print result/comments and return to menu.*
3000-3540	*P.TEST no.2. As per P.TEST no.1 except no message and there are 30 questions.*
4000-4230	*IQ test results. Get IQ value from score, print score, print IQ value and comments, return to menu.*
5000-5130	*IQ test examples.*
9990-9999	*Auto-save/verify routine: By GOTO 9900, and REM for program size: 34.6K.*

THE PRACTICE ELEMENT

It must be noted that IQ tests cannot be regarded as totally reliable, and that the test can only be used once on a given individual because of the practice element. However, an examination of the listing should provide some very useful routines which you could incorporate into your own programs. For example: The named size/free variables; the message routines and the real-time clock (based on the FRAMES system variable); and finally, the auto-save/verify routine which saves time during the development of any program.


```
  1 REM I.Q. TEST.
  2 POKE 23609,250: POKE 23561,
6: POKE 23562,3: LET SIZE=(PEEK
23627+256*PEEK 23628-23755)/1000
: LET FREE=(65536-USR 7962)/1000
  3 REM
  4 REM ************************
  5 REM ***I.Q. TEST PROGRAM***
  6 REM **** By G.Turnbull.****
  7 REM **For 48K ZX Spectrum**
  8 REM ************************
  9 REM
 10 REM INITIAL SCREEN.
 11 LET A$="  ■■■   ■■■■   ■■■
■■■ ■■■ ■■■  "
 12 LET B$="   ■   ■ ■   ■
■   ■   ■   "
 13 LET C$="   ■   ■ ■   ■
■■■ ■■■■  ■   "
 14 LET D$="   ■   ■ ■■   ■
■     ■  ■   "
 15 LET E$="  ■■■ ■ ■■■■ ■  ■
■■■ ■■■■  ■ ■ "
 16 LET F$=A$+B$+C$+D$+E$
 17 BORDER 6: PAPER 6: INK 2: C
LS : PRINT AT 7,0;F$;AT 12,11;"■
■": BEEP .6,35: PAUSE 75
 20 REM LOAD M.C.
 21 CLEAR 65110: RESTORE : FOR
A=65110 TO 65128: READ B: POKE A
,B: NEXT A: READ C: LET C=C+6512
8: FOR A=65129 TO C: READ B: POK
E A,B: NEXT A
 22 DATA 62,9,237,71,237,94,201
,0,0,0,62,62,237,71,237,86,201,0
,0
 23 DATA 21,229,213,197,245,17,
16,0,33,16,0,205,181,03,241,193,
209,225,243,195,56,0
 25 REM INTRO.
 26 PAPER 4: BORDER 4: INK 0: C
LS : PRINT "INTELLIGENCE QUOTIEN
T (I.Q) TEST"; OVER 1;AT 0,0;"__
_____"''
'
 27 LET Y$=" THIS PROGRAM CONTA
INS ONE I.Q. TEST AND TWO PERSON
ALITY TESTS. THE TEST CONSISTS O
F 50 Q's,    AND HAS A TIME LIMI
T OF 30 MINS (BY A REAL TIME CLO
CK). IT MUST BE NOTED THAT I.Q T
ESTS CANNOT  BE REGARED AS TOTAL
LY ACCURATE.  THE TESTS ONLY SHO
W POTENTIAL  AND CAN ONLY BE USE
D ONCE.      INTELLIGENCE IS A
VERY VARIED  INDIVIDUAL FACTOR A
ND IS RELATEDTO A GENERAL LEVEL
OF ABILITY,  AS SHOWN IN THE PER
FORMANCE OF AWIDE RANGE OF DIFFE
RENT TASKS."
 28 RANDOMIZE USR 65110: FOR Z=
1 TO LEN Y$: IF Y$(Z)=" " THEN
PRINT " ";: GO TO 30
 29 PRINT Y$(Z);: BEEP .05,25:
PAUSE 3
 30 NEXT Z: RANDOMIZE USR 65120
: PAUSE 25: GO SUB 50
 35 REM TEST MENU.
 36 PAPER 0: INK 7: BORDER 0: B
RIGHT 1: CLS : PRINT "I.Q. TEST
MENU:"; OVER 1;AT 0,0;"_________
_____"''
 37 PRINT "PRESS 1 FOR I.Q. TES
T."''"2 FOR PERSONALITY TEST No.
1.","3 FOR PERSONALITY TEST No.2
.","4 FOR I.Q. TEST EXAMPLES."''
 38 INPUT "TEST No.";X: IF X=4
THEN  GO TO 5000
 39 IF X<1 OR X>4 THEN  GO TO 3
8
 40 GO TO X*1000
 50 REM CONT. ROUTINE.
 55 PRINT #0;"PRESS ANY KEY TO
CONTINUE:": PAUSE 0: PRINT #0: R
ETURN
 60 REM START ROUTINE.
 65 PRINT #0;"PRESS ANY KEY TO
START CLOCK:": PAUSE 0: PRINT #0
: RETURN
 70 REM Q. No. ROUTINE.
 75 INPUT "ENTER THE Q.No. YOU
WISH TO TRY:";B: IF B<0 OR B>50
THEN  GO TO 75
 76 IF B=0 THEN  GO TO 1955
 77 RETURN
 80 REM CHECK SCORE VALUE.
 85 IF S(B)=1 THEN  PRINT AT 2,
0; FLASH 1;"QUESTION ALREADY ANS
WERED!"; FLASH 0: GO TO 1040
 86 RETURN
 90 REM STOP REPEAT Q.
 91 LET BB=BB+1
 95 LET S(B)=1
```

```
 96 RETURN
 100 REM UPDATE CLOCK.
 110 DEF FN Z()=INT ((65536*PEEK
 23674+256*PEEK 23673+PEEK 23672
)/50): LET SEC2=FN Z(): LET SEC3
=SEC2-SEC1
 120 IF SEC3<60 THEN  LET SEC=SE
C3: GO TO 140
 130 LET MIN=INT SEC3/60: LET SE
C=(MIN-INT MIN)*60: LET MIN=INT
MIN: LET SEC=INT (SEC+.5)
 140 PRINT AT 0,0;"TIME=";AT 0,7
;"."
 150 IF MIN<=9 AND SEC<=9 THEN
PRINT AT 0,5;"0";MIN;AT 0,8;"0";
SEC
 160 IF MIN>9 AND SEC<=9 THEN  P
RINT AT 0,5;MIN;AT 0,8;"0";SEC
 170 IF MIN<=9 AND SEC>9 THEN  P
RINT AT 0,5;"0";MIN;AT 0,8;SEC
 180 IF MIN>9 AND SEC>9 THEN  PR
INT AT 0,5;MIN;AT 0,8;SEC
 190 RETURN
 200 REM INITIALISE.
 210 LET BB=0: DIM S(50): LET IQ
=0: LET W$="
             ": LET SCORE=0: LET
MIN=0: LET SEC=0
 220 DEF FN Z()=INT ((65536*PEEK
 23674+256*PEEK 23673+PEEK 23672
)/50)
 230 GO SUB 60: REM START TEST.
 240 LET SEC1=FN Z()
 250 RETURN
 300 REM 2*NO. ANSWER.
 310 INPUT "ENTER No. OF 1st. IT
EM (X):";X
 320 IF X<0 OR X>6 THEN  GO TO 3
10
 330 INPUT "ENTER No. OF 2nd. IT
EM (Y):";Y
 340 IF Y<0 OR Y>6 THEN  GO TO 3
20
 350 IF x<>0 AND X=Y THEN  PRINT
 AT 2,0;"TRY AGAIN!": GO TO 1040
 360 RETURN
 400 REM SINGLE NO. ANSWER.
 410 INPUT "ENTER YOUR VALUE FOR
 A:";X
 420 RETURN
 500 REM ONE WORD ANSWER.
 510 INPUT "ENTER THE WORD/PART-
WORD ANSWER:";A$
 520 RETURN
 600 REM P.T.1 A. & SCORE.
 610 INPUT "ENTER ANSWER TO Q. (
A/B):";A$
 620 IF A$<>"a" AND A$<>"b" THEN
  GO TO 610
 630 IF A$="a" THEN  LET PT1A=PT
1A+1
 640 CLS : RETURN
 650 REM P.T.2 A. & SCORE.
 660 INPUT "ENTER ANSWER TO Q. (
A/B):";A$
 670 IF A$<>"a" AND A$<>"b" THEN
  GO TO 660
 680 IF A$="b" THEN  LET PT2B=PT
2B+1
 690 CLS : RETURN
 700 REM LOAD U.D.G.
 710 RESTORE 720: FOR B=0 TO 7:
READ C: POKE USR CHR$ 144+B,C: N
EXT B
 720 DATA 0,24,0,126,126,0,24,0
 730 RETURN
 800 REM P.TEST NO.1 RESULT.
 801 IF PT1A=20 THEN  PRINT "EXT
REMELY INTROVERTED!"
 802 IF PT1A=19 THEN  PRINT "VER
Y INTROVERTED"
 803 IF PT1A=18 THEN  PRINT "QUI
TE INTROVERTED"
 804 IF PT1A=17 OR PT1A=16 THEN
 PRINT "SOMEWHAT INTROVERTED"
 805 IF PT1A=15 OR PT1A=14 THEN
 PRINT "SLIGHTLY INTROVERTED"
 806 IF PT1A=13 OR PT1A=12 THEN
 PRINT "A SHADE INTROVERTED"
 807 IF PT1A=11 OR PT1A=10 OR PT
1A=9 THEN  PRINT "AVERAGE"
 808 IF PT1A=8 OR PT1A=7 THEN  P
RINT "A SHADE EXTROVERTED"
 809 IF PT1A=5 OR PT1A=6 THEN  P
RINT "SLIGHTLY EXTROVERTED"
 810 IF PT1A=3 OR PT1A=4 THEN  P
RINT "SOMEWHAT EXTROVERTED"
 811 IF PT1A=2 THEN  PRINT "QUIT
E EXTROVERTED"
 812 IF PT1A=1 THEN  PRINT "VERY
 EXTROVERTED"
 813 IF PT1A=0 THEN  PRINT "EXTR
EMELY EXTROVERTED!"
 815 RETURN
 900 REM P.TEST NO.2 RESULT.
 901 IF PT2B=30 OR PT2B=29 THEN
 PRINT "UNSHAKABLE!"
 902 IF PT2B=28 OR PT2B=27 THEN
 PRINT "IMPERTURBABLE"
 903 IF PT2B=26 OR PT2B=25 THEN
 PRINT "UNFLAPPABLE"
 904 IF PT2B=24 OR PT2B=23 THEN
 PRINT "CALM"
 905 IF PT2B=22 OR PT2B=21 THEN
 PRINT "BALANCED"
 906 IF PT2B=20 OR PT2B=19 OR PT
2B=18 THEN  PRINT "STEADY"
 907 IF PT2B=17 OR PT2B=16 OR PT
2B=15 OR PT2B=14 THEN  PRINT "AV
ERAGE"
```

```
 908 IF PT2B=13 OR PT2B=12 OR PT
2B=11 THEN  PRINT "SYMPATHETIC"
 909 IF PT2B=10 OR PT2B=9 THEN
PRINT "SUGGESTIBLE"
 910 IF PT2B=8 OR PT2B=7 THEN  P
RINT "EMOTIONAL"
 911 IF PT2B=6 OR PT2B=5 THEN  P
RINT "SENSITIVE"
 912 IF PT2B=4 OR PT2B=3 THEN  P
RINT "OVERSENSITIVE"
 913 IF PT2B=2 OR PT2B=1 THEN  P
RINT "NERVOUS"
 914 IF PT2B=1 OR PT2B=0 THEN  P
RINT "NEUROTIC!"
 915 RETURN
1000 REM I.Q. TEST.
1005 PAPER 6: BORDER 6: BRIGHT 0
: INK 2: CLS : GO SUB 700
1010 PRINT "I.Q.TEST.:"; OVER 1;
AT 0,0;"_________"''
1015 LET Y$=" THIS IS A VERBAL/N
UMERICAL TYPEOF TEST. NO CALCULA
TORS ARE      ALLOWED. YOU HAVE 3
0 MINS. SO    USE THE TIME WELL,
DON'T GET     STUCK ON ANY ONE Q.
 TO PASS      ENTER 0 TO ANY ANSW
ER(S).         ONCE STARTED YOU M
UST ENTER THEQ. No. YOU WISH TO
TRY, THIS     ALLOWS YOU TO RETUR
N TO A PASSEDQ. IF YOU HAVE ANY
TIME LEFT.     ENTER 0 IF YOU ARE
 SURE YOU AREFINISHED BEFORE THE
 TIME IS UP. THE CLOCK IS UPDATE
D FOR EACH Q."
1016 RANDOMIZE USR 65110: FOR Z=
1 TO LEN Y$: IF Y$(Z)=" " THEN
PRINT " ";: GO TO 1018
1017 PRINT Y$(Z);: BEEP .05,25:
PAUSE 2
1018 NEXT Z: RANDOMIZE USR 65120
1020 GO SUB 50
1025 CLS : PRINT AT 0,16;"I.Q. T
EST:"; OVER 1;AT 0,16;"_________
"
1030 GO SUB 200: REM START.
1040 GO SUB 70: REM CHOOSE Q.
1045 IF BB=50 THEN  PRINT  FLASH
 1;AT 2,0;"ALL QUESTIONS ANSWERE
D!"; FLASH 0: PAUSE 50: GO TO 19
51
1050 GO SUB 100: REM GO CLOCK.
1055 IF MIN=30 THEN  GO TO 1950
1060 GO SUB 80: REM CHECK Q.
1065 GO TO 1100+(B*16): REM GO
QUESTION.
1100 REM I.Q TEST Q's.
1116 PRINT AT 2,0;"Q's. 1-6:ANAL
OGIES: Q.1."; OVER 1;AT 2,0;"___
_____________  ___";AT 4,0;"EN
TER THE No.s OF THE EXACT TWO WO
RDS NEEDED TO COMPLETE THESE:"
1117 PRINT AT 7,0;"SITTER IS TO
CHAIR AS X IS TO Y:"
1118 PRINT AT 9,0;"(CUP, SAUCER,
 PLATE, LEG)";AT 10,2;"1";TAB 8;
"2";TAB 16;"3";TAB 22;"4"
1119 GO SUB 300: REM ANSWER.
1120 IF X=1 AND Y=2 THEN  LET SC
ORE=SCORE+1
1121 IF X=0 AND Y=0 THEN  GO TO
1124: REM PASS.
1123 GO SUB 90: REM NO REPEAT Q.
1124 FOR A=2 TO 10: PRINT AT A,0
;W$: NEXT A
1125 GO TO 1040: REM RETURN.
1132 PRINT AT 2,0;"Q. No.2:"; OV
ER 1;AT 2,0;"_______"
1133 PRINT AT 4,0;"NEEDLE IS TO
THREAD AS X IS TO Y"
1134 PRINT AT 6,0;"(COTTON, SEW,
 LEADER, FOLLOWER)";AT 7,3;"1";T
AB 10;"2";TAB 16;"3";TAB 26;"4"
1135 GO SUB 300
1136 IF X=3 AND Y=4 THEN  LET SC
ORE=SCORE+1
1137 IF X=0 AND Y=0 THEN  GO TO
1139
1138 GO SUB 90
1139 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1140 GO TO 1040
1148 PRINT AT 2,0;"Q. No.3:"; OV
ER 1;AT 2,0;"_______"
1149 PRINT AT 4,0;"BETTER IS TO
WORSE AS X IS TO Y:"
1150 PRINT AT 6,0;"(REJOICE, CHO
ICE, BAD, MOURN)";AT 7,4;"1";TAB
 12;"2";TAB 19;"3";TAB 25;"4"
1151 GO SUB 300
1152 IF X=1 AND Y=4 THEN  LET SC
ORE=SCORE+1
1153 IF X=0 AND Y=0 THEN  GO TO
1155
1154 GO SUB 90
1155 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1156 GO TO 1040
1164 PRINT AT 2,0;"Q. No.4:"; OV
ER 1;AT 2,0;"_______"
1165 PRINT AT 4,0;"FLOOR IS TO S
UPPORT AS X IS TO Y"
1166 PRINT AT 6,0;"(WINDOW, GLAS
S, VIEW, BRICK)";AT 7,3;"1";TAB
11;"2";TAB 17;"3";TAB 24;"4"
1167 GO SUB 300
1168 IF X=1 AND Y=3 THEN  LET SC
ORE=SCORE+1
1169 IF X=0 AND Y=0 THEN  GO TO
1171
```

```
1170 GO SUB 90
1171 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1172 GO TO 1040
1180 PRINT AT 2,0;"Q. No.5:"; OV
ER 1;AT 2,0;"________"
1181 PRINT AT 4,0;"ISLAND IS TO
WATER AS X IS TO Y:"
1182 PRINT AT 6,0;"(WITHOUT, CEN
TRE, DIAGONAL,";AT 8,0;"PERIMETE
R)";AT 7,4;"1";TAB 12;"2";TAB 21
;"3";AT 9,4;"4"
1183 GO SUB 300
1184 IF X=2 AND Y=4 THEN  LET SC
ORE=SCORE+1
1185 IF X=0 AND Y=0 THEN  GO TO
1187
1186 GO SUB 90
1187 FOR A=2 TO 9: PRINT AT A,0;
W$: NEXT A
1188 GO TO 1040
1196 PRINT AT 2,0;"Q. No.6:"; OV
ER 1;AT 2,0;"________"
1197 PRINT AT 4,0;"VEIL IS TO CU
RTAIN AS X IS TO Y:"
1198 PRINT AT 6,0;"(EYES, SEE, W
INDOW, HEAR)";AT 7,2;"1";TAB 8;"
2";TAB 14;"3";TAB 21;"4"
1199 GO SUB 300
1200 IF X=1 AND Y=3 THEN  LET SC
ORE=SCORE+1
1201 IF X=0 AND Y=0 THEN  GO TO
1203
1202 GO SUB 90
1203 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1204 GO TO 1040
1212 PRINT AT 2,0;"Q's. 7-12:SIM
ILARITIES: Q.7.";  OVER 1;AT 2,0;
"_____  ___";AT
 4,0;"ENTER THE No.s OF TWO WORD
S WITHTHE MOST SIMILAR MEANINGS.
"
1213 PRINT AT 7,0;"DIVULGE, DIVE
R, REVEAL, REVERT";AT 8,3;"1";TA
B 11;"2";TAB 18;"3";TAB 26;"4"
1215 GO SUB 300
1216 IF X=1 OR X=3 AND Y=1 OR Y=
3 THEN  LET SCORE=SCORE+1
1217 IF X=0 AND Y=0 THEN  GO TO
1219
1218 GO SUB 90
1219 FOR A=2 TO 8: PRINT AT A,0;
W$: NEXT A
1220 GO TO 1040
1228 PRINT AT 2,0;"Q. No.8:"; OV
ER 1;AT 2,0;"________"
1229 PRINT AT 4,0;"BLESSING, BLE
SS, BENEDICTION,";AT 6,0;"BLESSE
D";AT 5,3;"1";TAB 12;"2";TAB 21;
"3";AT 7,3;"4"
1231 GO SUB 300
1232 IF X=1 OR X=3 AND Y=1 OR Y=
3 THEN  LET SCORE=SCORE+1
1233 IF X=0 AND Y=0 THEN  GO TO
1235
1234 GO SUB 90
1235 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1236 GO TO 1040
1244 PRINT AT 2,0;"Q. No.9:"; OV
ER 1;AT 2,0;"________"
1245 PRINT AT 4,0;"INTELLIGENCE,
 SPEEDINESS,";AT 6,0;"CURRENTS,
TIDINGS";AT 5,5;"1";TAB 18;"2";A
T 7,3;"3";TAB 13;"4"
1247 GO SUB 300
1248 IF X=1 OR X=4 AND Y=1 OR Y=
4 THEN  LET SCORE=SCORE+1
1249 IF X=0 AND Y=0 THEN  GO TO
1251
1250 GO SUB 90
1251 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1252 GO TO 1040
1260 PRINT AT 2,0;"Q. No.10:"; O
VER 1;AT 2,0;"_________"
1261 PRINT AT 4,0;"TALE, NOVEL,
VOLUME, STORY";AT 5,1;"1";TAB 8;
"2";TAB 15;"3";TAB 23;"4"
1263 GO SUB 300
1264 IF X=1 OR X=4 AND Y=1 OR Y=
4 THEN  LET SCORE=SCORE+1
1265 IF X=0 AND Y=0 THEN  GO TO
1267
1266 GO SUB 90
1267 FOR A=2 TO 5: PRINT AT A,0;
W$: NEXT A
1268 GO TO 1040
1276 PRINT AT 2,0;"Q. No.11:"; O
VER 1;AT 2,0;"_________"
1277 PRINT AT 4,0;"INCARCERATE,
PUNISH, CANE,";AT 6,0;"CHASTISE"
;AT 5,5;"1";TAB 15;"2";TAB 22;"3
";AT 7,4;"4"
1279 GO SUB 300
1280 IF X=2 OR X=4 AND Y=2 OR Y=
4 THEN  LET SCORE=SCORE+1
1281 IF X=0 AND Y=0 THEN  GO TO
1283
1282 GO SUB 90
1283 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1284 GO TO 1040
1292 PRINT AT 2,0;"Q. No.12:"; O
VER 1;AT 2,0;"_________"
1293 PRINT AT 4,0;"LUMP, WOOD, R
AY, BEAM";AT 5,1;"1";TAB 7;"2";T
AB 13;"3";TAB 18;"4"
1295 GO SUB 300
```

```
1296 IF X=3 OR X=4 AND Y=3 OR Y=
4 THEN  LET SCORE=SCORE+1
1297 IF X=Ø AND Y=Ø THEN  GO TO
1299
1298 GO SUB 9Ø
1299 FOR A=2 TO 5: PRINT AT A,Ø;
W$: NEXT A
13ØØ GO TO 1Ø4Ø
13Ø8 PRINT AT 2,Ø;"Q's.13-19:MAT
HS EQUATIONS. Q.13."; OVER 1;AT
2,Ø;"_____
____";AT 4,Ø;"WORK OUT THE VALUE
 OF A IN THESEEQUATIONS, ON PAPE
R AND THEN    ENTER IT WHEN PROM
PTED."
13Ø9 PRINT AT 8,Ø;" 42=A*(A+1)"
1311 GO SUB 4ØØ
1312 IF X=6 THEN  LET SCORE=SCOR
E+1
1313 IF X=Ø THEN  GO TO 1315
1314 GO SUB 9Ø
1315 FOR A=2 TO 8: PRINT AT A,Ø;
W$: NEXT A
1316 GO TO 1Ø4Ø
1324 PRINT AT 2,Ø;"Q. No.14:"; O
VER 1;AT 2,Ø;"_________"
1325 PRINT AT 4,Ø;" 8*7=2*A+A+2"
1327 GO SUB 4ØØ
1328 IF X=18 THEN  LET SCORE=SCO
RE+1
1329 IF X=Ø THEN  GO TO 1331
133Ø GO SUB 9Ø
1331 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1332 GO TO 1Ø4Ø
134Ø PRINT AT 2,Ø;"Q. No.15:"; O
VER 1;AT 2,Ø;"_________"
1341 PRINT AT 4,Ø;" 2+(9*6)=14*A
"
1343 GO SUB 4ØØ
1344 IF X=4 THEN  LET SCORE=SCOR
E+1
1345 IF X=Ø THEN  GO TO 1347
1346 GO SUB 9Ø
1347 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1348 GO TO 1Ø4Ø
1356 PRINT AT 2,Ø;"Q. No.16:"; O
VER 1;AT 2,Ø;"_________"
1357 PRINT AT 4,Ø;" 12+8-21=16+A
"
1359 GO SUB 4ØØ
136Ø IF X=-17 THEN  LET SCORE=SC
ORE+1
1361 IF X=Ø THEN  GO TO 1363
1362 GO SUB 9Ø
1363 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1364 GO TO 1Ø4Ø
1372 PRINT AT 2,Ø;"Q. No.17:"; O
VER 1;AT 2,Ø;"_________"
1373 PRINT AT 4,Ø;" 5*9=15*A"
1375 GO SUB 4ØØ
1376 IF X=3 THEN  LET SCORE=SCOR
E+1
1377 IF X=Ø THEN  GO TO 1379
1378 GO SUB 9Ø
1379 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
138Ø GO TO 1Ø4Ø
1388 PRINT AT 2,Ø;"Q. No.18:"; O
VER 1;AT 2,Ø;"_________"
1389 PRINT AT 4,Ø;" Ø.21*Ø.25=Ø.
6*Ø.7*A"
1391 GO SUB 4ØØ
1392 IF X=2 THEN  LET SCORE=SCOR
E+1
1393 IF X=Ø THEN  GO TO 1395
1394 GO SUB 9Ø
1395 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1396 GO TO 1Ø4Ø
14Ø4 PRINT AT 2,Ø;"Q. No.19:"; O
VER 1;AT 2,Ø;"_________"
14Ø5 PRINT AT 4,Ø;" Ø.28*Ø.35=Ø.
5*Ø.4*A"
14Ø7 GO SUB 4ØØ
14Ø8 IF X=4 THEN  LET SCORE=SCOR
E+1
14Ø9 IF X=Ø THEN  GO TO 1411
141Ø GO SUB 9Ø
1411 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1412 GO TO 1Ø4Ø
142Ø PRINT AT 2,Ø;"Q's.2Ø-25: LI
NKS. Q.2Ø."; OVER 1;AT 2,Ø;"____
_____  _____";AT 4,Ø;"ENTE
R THE FULL WORD OR THE      MISS
ING PART OF THE WORD IN THE BRAC
KETS, WHICH MEANS THE SAME  IN O
NE SENSE AS BOTH OF THE     SURR
OUNDING WORDS."
1421 PRINT AT 1Ø,Ø;"WHIP (L**H)
TIE"
1423 GO SUB 5ØØ
1424 IF A$="as" OR A$="lash" THE
N  LET SCORE=SCORE+1
1425 IF A$="Ø" THEN  GO TO 1427
1426 GO SUB 9Ø
1427 FOR A=2 TO 1Ø: PRINT AT A,Ø
;W$: NEXT A
1428 GO TO 1Ø4Ø
1436 PRINT AT 2,Ø;"Q. No. 21:";
OVER 1;AT 2,Ø;"_________"
1437 PRINT AT 4,Ø;"DASH (D**T) M
ISSILE"
1439 GO SUB 5ØØ
144Ø IF A$="ar" OR A$="dart" THE
N  LET SCORE=SCORE+1
1441 IF A$="Ø" THEN  GO TO 1443
```

```
1442 GO SUB 9Ø
1443 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1444 GO TO 1Ø4Ø
1452 PRINT AT 2,Ø;"Q. No. 22:";
OVER 1;AT 2,Ø;"__________"
1453 PRINT AT 4,Ø;"MOULD (F**M)
CLASS"
1455 GO SUB 5ØØ
1456 IF A$="or" OR A$="form" THE
N  LET SCORE=SCORE+1
1457 IF A$="Ø" THEN  GO TO 1459
1458 GO SUB 9Ø
1459 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1460 GO TO 1Ø4Ø
1468 PRINT AT 2,Ø;"Q. No. 23:";
OVER 1;AT 2,Ø;"__________"
1469 PRINT AT 4,Ø;"SQUASH (P***S
) CROWD"
1471 GO SUB 5ØØ
1472 IF A$="res" OR A$="press" T
HEN  LET SCORE=SCORE+1
1473 IF A$="Ø" THEN  GO TO 1475
1474 GO SUB 9Ø
1475 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1476 GO TO 1Ø4Ø
1484 PRINT AT 2,Ø;"Q. No. 24:";
OVER 1;AT 2,Ø;"__________"
1485 PRINT AT 4,Ø;"THIN (F**E) G
OOD"
1486 GO SUB 5ØØ
1488 IF A$="in" OR A$="fine" THE
N  LET SCORE=SCORE+1
1489 IF A$="Ø" THEN  GO TO 1491
149Ø GO SUB 9Ø
1491 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1492 GO TO 1Ø4Ø
15ØØ PRINT AT 2,Ø;"Q. No. 25:";
OVER 1;AT 2,Ø;"__________"
1501 PRINT AT 4,Ø;"IGNITE (F**E)
 SHOOT"
1503 GO SUB 5ØØ
1504 IF A$="ir" OR A$="fire" THE
N  LET SCORE=SCORE+1
1505 IF A$="Ø" THEN  GO TO 1507
1506 GO SUB 9Ø
1507 FOR A=2 TO 4: PRINT AT A,Ø;
W$: NEXT A
1508 GO TO 1Ø4Ø
1516 PRINT AT 2,Ø;"Q's. 26-31: O
PPOSITES: Q.26."; OVER 1;AT 2,Ø;
"_____ ____";AT
 4,Ø;"ENTER THE No.s OF TWO WORD
S WITHALMOST OPPOSITE MEANINGS."
1517 PRINT AT 7,Ø;"HEAVY, LARGE,
 LIGHT, BIG, WEIGHT";AT 8,2;"1";
TAB 9;"2";TAB 16;"3";TAB 22;"4";
TAB 29;"5"
1519 GO SUB 3ØØ
152Ø IF X=1 OR X=3 AND Y=1 OR Y=
3 THEN  LET SCORE=SCORE+1
1521 IF X=Ø AND Y=Ø THEN  GO TO
1523
1522 GO SUB 9Ø
1523 FOR A=2 TO 8: PRINT AT A,Ø;
W$: NEXT A
1524 GO TO 1Ø4Ø
1532 PRINT AT 2,Ø;"Q. No. 27:";
OVER 1;AT 2,Ø;"__________"
1533 PRINT AT 4,Ø;"INSULT, DENY,
 DENIGRATE, FIRM,";AT 6,Ø;"AFFIR
M"
1534 PRINT AT 5,2;"1";TAB 9;"2";
TAB 18;"3";TAB 26;"4";AT 7,2;"5"
1535 GO SUB 3ØØ
1536 IF X=2 OR X=5 AND Y=2 OR Y=
5 THEN  LET SCORE=SCORE+1
1537 IF X=Ø AND Y=Ø THEN  GO TO
1539
1538 GO SUB 9Ø
1539 FOR A=2 TO 7: PRINT AT A,Ø;
W$: NEXT A
154Ø GO TO 1Ø4Ø
1548 PRINT AT 2,Ø;"Q. No. 28:";
OVER 1;AT 2,Ø;"__________"
1549 PRINT AT 4,Ø;"MISSED, VEIL,
 CONFUSE, SECRET,";AT 6,Ø;"EXPOS
E"
155Ø PRINT AT 5,2;"1";TAB 9;"2";
TAB 17;"3";TAB 25;"4";AT 7,2;"5"
1551 GO SUB 3ØØ
1552 IF X=2 OR X=5 AND Y=2 OR Y=
5 THEN  LET SCORE=SCORE+1
1553 IF X=Ø AND Y=Ø THEN  GO TO
1555
1554 GO SUB 9Ø
1555 FOR A=2 TO 7: PRINT AT A,Ø;
W$: NEXT A
1556 GO TO 1Ø4Ø
1564 PRINT AT 2,Ø;"Q. No. 29:";
OVER 1;AT 2,Ø;"__________"
1565 PRINT AT 4,Ø;"FRANK, OVERT,
 PLAIN, SIMPLE,";AT 6,Ø;"SECRETI
VE"
1566 PRINT AT 5,2;"1";TAB 9;"2";
TAB 16;"3";TAB 24;"4";AT 7,3;"5"
1567 GO SUB 3ØØ
1568 IF X=1 OR X=5 AND Y=1 OR Y=
5 THEN  LET SCORE=SCORE+1
1569 IF X=Ø AND Y=Ø THEN  GO TO
1571
157Ø GO SUB 9Ø
1571 FOR A=2 TO 7: PRINT AT A,Ø;
W$: NEXT A
1572 GO TO 1Ø4Ø
158Ø PRINT AT 2,Ø;"Q. No. 3Ø:";
```

```
OVER 1;AT 2,0;"__________"
1581 PRINT AT 4,0;"AGGRAVATE, PL
EASE, ENJOY,";AT 6,0;"IMPROVE, L
IKE"
1582 PRINT AT 5,3;"1";TAB 14;"2"
;TAB 21;"3";AT 7,3;"4";TAB 10;"5
"
1583 GO SUB 300
1584 IF X=1 OR X=4 AND Y=1 OR Y=
4 THEN  LET SCORE=SCORE+1
1585 IF X=0 AND Y=0 THEN  GO TO
1587
1586 GO SUB 90
1587 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1588 GO TO 1040
1596 PRINT AT 2,0;"Q. No. 31:";
OVER 1;AT 2,0;"__________"
1597 PRINT AT 4,0;"ANTEDATE, PRI
MITIVE, PRIMORDIAL";AT 6,0;"PRIM
ATE, ULTIMATE"
1598 PRINT AT 5,3;"1";TAB 13;"2"
;TAB 25;"3";AT 7,3;"4";TAB 13;"5
"
1599 GO SUB 300
1600 IF X=3 OR X=5 AND Y=3 OR Y=
5 THEN  LET SCORE=SCORE+1
1601 IF X=0 AND Y=0 THEN  GO TO
1603
1602 GO SUB 90
1603 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1604 GO TO 1040
1612 PRINT AT 2,0;"Q's. 32-36: M
ATHS SERIES: Q.32."; OVER 1;AT 2
,0;"_____  __
__";AT 4,0;"ENTER THE VALUE OF T
HE NO. WHICHLOGICALLY FOLLOWS TH
E OTHERS."
1613 PRINT AT 7,0;" 3,6,12,24,  (
A)"
1615 GO SUB 400
1616 IF X=48 THEN  LET SCORE=SCO
RE+1
1617 IF X=0 THEN  GO TO 1619
1618 GO SUB 90
1619 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1620 GO TO 1040
1628 PRINT AT 2,0;"Q. No. 33:";
OVER 1;AT 2,0;"__________"
1629 PRINT AT 4,0;" 81,54,36,24
(A)"
1631 GO SUB 400
1632 IF X=16 THEN  LET SCORE=SCO
RE+1
1633 IF X=0 THEN  GO TO 1635
1634 GO SUB 90
1635 FOR A=2 TO 4: PRINT AT A,0;
W$: NEXT A
1636 GO TO 1040
1644 PRINT AT 2,0;"Q. No. 34:";
OVER 1;AT 2,0;"__________"
1645 PRINT AT 4,0;" 2,3,5,9,17 (
A)"
1647 GO SUB 400
1648 IF X=33 THEN  LET SCORE=SCO
RE+1
1649 IF X=0 THEN  GO TO 1651
1650 GO SUB 90
1651 FOR A=2 TO 4: PRINT AT A,0;
W$: NEXT A
1652 GO TO 1040
1660 PRINT AT 2,0;"Q. No. 35:";
OVER 1;AT 2,0;"__________"
1661 PRINT AT 4,0;" 7,13,19,25 (
A)"
1663 GO SUB 400
1664 IF X=31 THEN  LET SCORE=SCO
RE+1
1665 IF X=0 THEN  GO TO 1667
1666 GO SUB 90
1667 FOR A=2 TO 4: PRINT AT A,0;
W$: NEXT A
1668 GO TO 1040
1676 PRINT AT 2,0;"Q. No. 36:";
OVER 1;AT 2,0;"__________"
1677 PRINT AT 4,0;" 9,16,25,36 (
A)"
1679 GO SUB 400
1680 IF X=49 THEN  LET SCORE=SCO
RE+1
1681 IF X=0 THEN  GO TO 1683
1682 GO SUB 90
1683 FOR A=2 TO 4: PRINT AT A,0;
W$: NEXT A
1684 GO TO 1040
1692 PRINT AT 2,0;"Q's. 37-42: M
ID-TERMS: Q.37."; OVER 1;AT 2,0;
"_____  _____";AT
 4,0;"IN EACH Q. THE THREE UPPER
 TERMSCORRESPOND TO THOSE BELOW,
      INSERT THE MISSING LETTERS
."
1693 PRINT AT 8,0;"FACE (BODY) L
EGS :","NOSE (N***) KNEES"
1695 GO SUB 500
1696 IF A$="avel" OR A$="navel"
THEN  LET SCORE=SCORE+1
1697 IF A$="0" THEN  GO TO 1699
1698 GO SUB 90
1699 FOR A=2 TO 9: PRINT AT A,0;
W$: NEXT A
1700 GO TO 1040
1708 PRINT AT 2,0;"Q. No. 38:";
OVER 1;AT 2,0;"__________"
1709 PRINT AT 4,0;"PAST (PRESENT
) FUTURE :","WAS (I***) WILL BE"
1711 GO SUB 500
```

```
1712 IF A$="s" OR A$="is" THEN
LET SCORE=SCORE+1
1713 IF A$="Ø" THEN  GO TO 1715
1714 GO SUB 9Ø
1715 FOR A=2 TO 5: PRINT AT A,Ø;
W$: NEXT A
1716 GO TO 1Ø4Ø
1724 PRINT AT 2,Ø;"Q. No. 39:";
OVER 1;AT 2,Ø;"_________"
1725 PRINT AT 4,Ø;"COMPLETE (INC
OMPLETE) BLANK :","ALWAYS (S***)
 NEVER"
1727 GO SUB 5ØØ
1728 IF A$="ometimes" OR A$="som
etimes" THEN  LET SCORE=SCORE+1
1729 IF A$="Ø" THEN  GO TO 1731
1730 GO SUB 9Ø
1731 FOR A=2 TO 5: PRINT AT A,Ø;
W$: NEXT A
1732 GO TO 1Ø4Ø
1740 PRINT AT 2,Ø;"Q. No. 4Ø:";
OVER 1;AT 2,Ø;"_________"
1741 PRINT AT 4,Ø;"GLUT (SCARCIT
Y) FAMINE :","MANY (F***) NONE"
1743 GO SUB 5ØØ
1744 IF A$="ew" OR A$="few" THEN
  LET SCORE=SCORE+1
1745 IF A$="Ø" THEN  GO TO 1747
1746 GO SUB 9Ø
1747 FOR A=2 TO 5: PRINT AT A,Ø;
W$: NEXT A
1748 GO TO 1Ø4Ø
1756 PRINT AT 2,Ø;"Q. No. 41:";
OVER 1;AT 2,Ø;"_________"
1757 PRINT AT 4,Ø;"RUSHING (PASS
ING) ENDURING :","EVANESCENT (T*
**T) ETERNAL"
1759 GO SUB 5ØØ
1760 IF A$="ransien" OR A$="tran
sient" THEN  LET SCORE=SCORE+1
1761 IF A$="Ø" THEN  GO TO 1763
1762 GO SUB 9Ø
1763 FOR A=2 TO 5: PRINT AT A,Ø;
W$: NEXT A
1764 GO TO 1Ø4Ø
1770 PRINT AT 2,Ø;"Q. No. 42:";
OVER 1;AT 2,Ø;"_________"
1773 PRINT AT 4,Ø;"NASCENT (MATU
RE) SENILE :","GREEN (R***) DECA
YED"
1775 GO SUB 5ØØ
1776 IF A$="ipe" OR A$="ripe" TH
EN  LET SCORE=SCORE+1
1777 IF A$="Ø" THEN  GO TO 1779
1778 GO SUB 9Ø
1779 FOR A=2 TO 5: PRINT AT A,Ø;
W$: NEXT A
1780 GO TO 1Ø4Ø
1788 PRINT AT 2,Ø;"Q's.43-47:SIM
ILAR/OPPOSITE: Q.26"; OVER 1;AT
2,Ø;"_____
 ____";AT 4,Ø;"ENTER THE No.s OF
 TWO WORDS WITHEITHER NEARLY EQU
AL MEANINGS OR ALMOST OPPOSITE M
EANINGS."
1789 PRINT AT 8,Ø;"RAPPORT, MERC
URIAL, HAPPY,";AT 1Ø,Ø;"RAPACIOU
S, PHLEGMATIC"
1790 PRINT AT 9,3;"1";TAB 13;"2"
;TAB 22;"3";AT 11,3;"4";TAB 15;"
5"
1791 GO SUB 3ØØ
1792 IF X=2 OR X=5 AND Y=2 OR Y=
5 THEN  LET SCORE=SCORE+1
1793 IF X=Ø AND Y=Ø THEN  GO TO
1795
1794 GO SUB 9Ø
1795 FOR A=2 TO 11: PRINT AT A,Ø
;W$: NEXT A
1796 GO TO 1Ø4Ø
1804 PRINT AT 2,Ø;"Q. No. 44:";
OVER 1;AT 2,Ø;"_________"
1805 PRINT AT 4,Ø;"TENACIOUS, RE
SOLVE, IRRESOLUTE,";AT 6,Ø;"SOLU
TION, TENACITY"
1806 PRINT AT 5,4;"1";TAB 13;"2"
;TAB 25;"3";AT 7,3;"4";TAB 13;"5
"
1807 GO SUB 3ØØ
1808 IF X=1 OR X=3 AND Y=1 OR Y=
3 THEN  LET SCORE=SCORE+1
1809 IF X=Ø AND Y=Ø THEN  GO TO
1811
1810 GO SUB 9Ø
1811 FOR A=2 TO 7: PRINT AT A,Ø;
W$: NEXT A
1812 GO TO 1Ø4Ø
1820 PRINT AT 2,Ø;"Q. No. 45:";
OVER 1;AT 2,Ø;"_________"
1821 PRINT AT 4,Ø;"REAL, RENAL,
LITERALLY,";AT 6,Ø;"SIMILARLY, V
ERITABLY"
1822 PRINT AT 5,1;"1";TAB 8;"2";
TAB 16;"3";AT 7,3;"4";TAB 15;"5"
1823 GO SUB 3ØØ
1824 IF X=5 OR X=3 AND Y=5 OR Y=
3 THEN  LET SCORE=SCORE+1
1825 IF X=Ø AND Y=Ø THEN  GO TO
1827
1826 GO SUB 9Ø
1827 FOR A=2 TO 7: PRINT AT A,Ø;
W$: NEXT A
1828 GO TO 1Ø4Ø
1836 PRINT AT 2,Ø;"Q. No. 46:";
OVER 1;AT 2,Ø;"_________"
1837 PRINT AT 4,Ø;"TOPOGRAPHY, H
EAP, PRIME, PLATEAU";AT 6,Ø;"HOL
E"
```

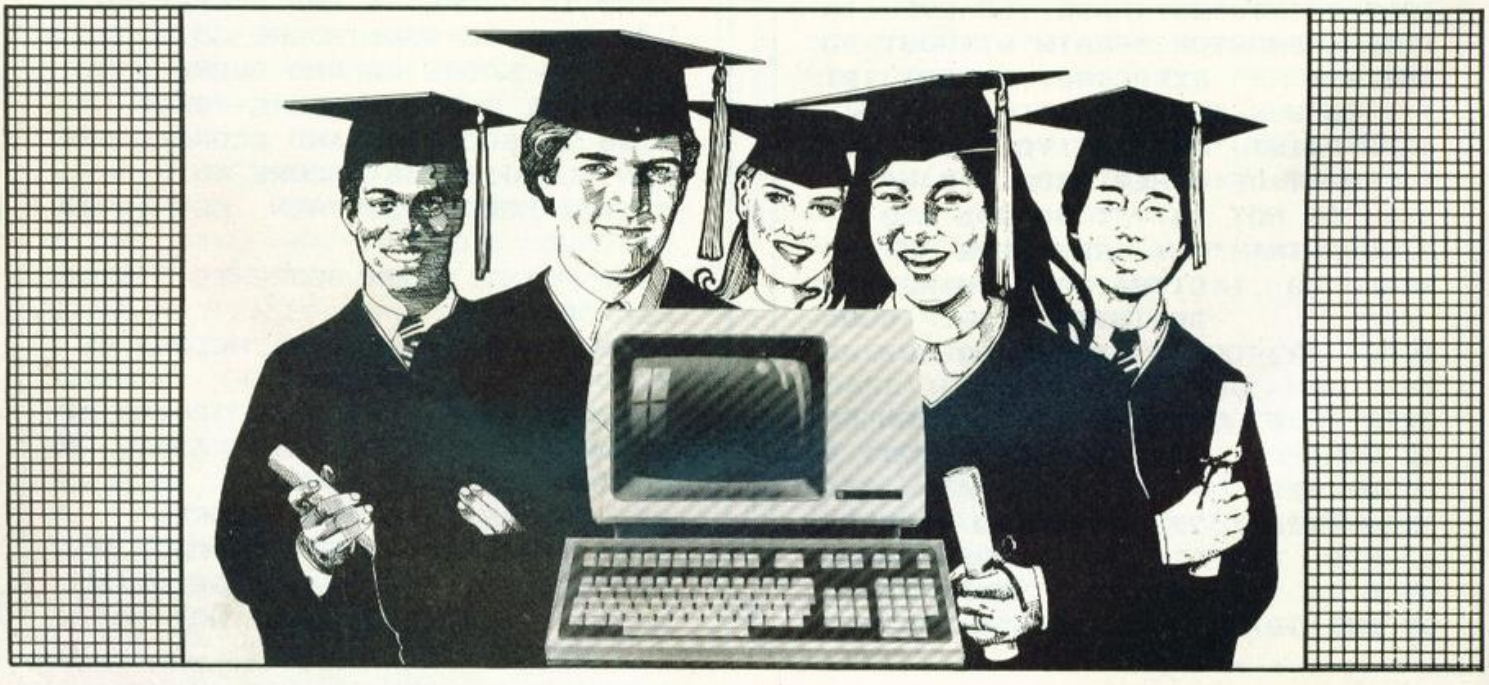
```
1838 PRINT AT 5,4;"1";TAB 13;"2"
;TAB 20;"3";TAB 28;"4";AT 7,1;"5
"
1839 GO SUB 300
1840 IF X=5 OR X=2 AND Y=5 OR Y=
2 THEN  LET SCORE=SCORE+1
1841 IF X=0 AND Y=0 THEN  GO TO
1843
1842 GO SUB 90
1843 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1844 GO TO 1040
1852 PRINT AT 2,0;"Q. No. 47:";
OVER 1;AT 2,0;"__________"
1853 PRINT AT 4,0;"HATE, AFFECTI
ON, AFFLICTION,";AT 6,0;"LOVE, P
ASSION"
1854 PRINT AT 5,1;"1";TAB 10;"2"
;TAB 22;"3";AT 7,1;"4";TAB 9;"5"
1855 GO SUB 300
1856 IF X=1 OR X=4 AND Y=1 OR Y=
4 THEN  LET SCORE=SCORE+1
1857 IF X=0 AND Y=0 THEN  GO TO
1859
1858 GO SUB 90
1859 FOR A=2 TO 7: PRINT AT A,0;
W$: NEXT A
1860 GO TO 1040
1868 PRINT AT 2,0;"Q's.48-50: MA
THS MID-TERMS: Q.26"; OVER 1;AT
2,0;"_____
 ____";AT 4,0;"IN EACH Q. THE 3
No.s ON THE    LEFT ARE RELATED
TO THOSE ON THERIGHT, ENTER THE
MISSING VALUE."
1869 PRINT AT 8,0;" 7 (12) 5 : 8
 (A) 3"
1871 GO SUB 400
1872 IF X=11 THEN  LET SCORE=SCO
RE+1
1873 IF X=0 THEN  GO TO 1875
1874 GO SUB 90
1875 FOR A=2 TO 8: PRINT AT A,0;
W$: NEXT A
1876 GO TO 1040
1884 PRINT AT 2,0;"Q. No. 49:";
OVER 1;AT 2,0;"__________"
1885 PRINT AT 4,0;" 3 (6) 2 : 3
(A) 3"
1887 GO SUB 400
1888 IF X=9 THEN  LET SCORE=SCOR
E+1
1889 IF X=0 THEN  GO TO 1891
1890 GO SUB 90
1891 FOR A=2 TO 4: PRINT AT A,0;
W$: NEXT A
1892 GO TO 1040
1900 PRINT AT 2,0;"Q. No. 50:";
OVER 1;AT 2,0;"__________"
1901 PRINT AT 4,0;" 49 (15) 64 :
 16 (A) 144"
1903 GO SUB 400
1904 IF X=16 THEN  LET SCORE=SCO
RE+1
1905 IF X=0 THEN  GO TO 1907
1906 GO SUB 90
1907 FOR A=2 TO 4: PRINT AT A,0;
W$: NEXT A
1908 GO TO 1040
1910 REM TIME UP/FINISHED.
1950 PRINT AT 2,0; FLASH 1;"TIME
 UP!"; FLASH 0: PAUSE 75
1955 PRINT AT 2,0;"YOU HAVE FINI
SHED ALL THE Q's.": PAUSE 75
1960 PAPER 6: BORDER 6: INK 0: C
LS
1970 GO SUB 4000: REM IQ RESULT
1990 GO TO 35
2000 REM PERSON. TEST NO.1.
2010 PAPER 6: BORDER 6: INK 0: B
RIGHT 0: CLS : LET PT1A=0
2020 PRINT "PERSONALITY TESTS:";
 OVER 1;AT 0,0;"_________________
_"''
2030 LET Y$=" THESE ARE TWO TEST
S WHICH HAVE NO RIGHT OR WRONG A
NSWERS, THEY ARE UNRELATED TO TH
E I.Q. TEST. YOU MUST ANSWER THE
 Q's. WITH A OR B AS QUICKLY AND
 AS HONESTLY AS POSSIBLE, DO NOT
 THINK ABOUT THEM FOR TOO LONG A
S EMOTIONS   ARE IMPORTANT. THE
RESULTS WILL BE GIVEN SOON AFTER
 THE END     OF EACH TEST."
2040 RANDOMIZE USR 65110: FOR Z=
1 TO LEN Y$: IF Y$(Z)=" " THEN
PRINT " ";: GO TO 2060
2050 PRINT Y$(Z);: BEEP .05,25:
PAUSE 2
2060 NEXT Z: RANDOMIZE USR 65120
: PAUSE 25: GO SUB 50: CLS
2065 PRINT "P.TEST NO.1:"; OVER
1;AT 0,0;"____________"''
2070 PRINT "1) WHICH WOULD YOU P
REFER TO BE A SCIENTIST (A) OR A
 POLITICIAN (B)?": GO SUB 600
2080 PRINT "2) DO YOU THINK THAT
 SOME WELL- KNOW, 'HONEST' PROFE
SSIONS DO   MORE HARM (A) THAN G
OOD (B) FOR THIS COUNTRY?": GO S
UB 600
2090 PRINT "3) WHICH IS MORE IMP
ORTANT TO A CRITIC, DISCRIMINATI
ON (B) OR   TOLERANCE (A)?": GO
SUB 600
2100 PRINT "4) WOULD YOU RATHER
BE YOUR OWN BOSS (A), OR A RECEP
TIONIST (B)?": GO SUB 600
```

```
2110 PRINT "5) SHOULD A DOCTOR A
LLOW HIMSELFT0 BE EMOTIONAL (B)
IN TREATING PATIENTS, OR NOT (A)
?": GO SUB 600
2120 PRINT "6) DO YOU FIND IT HA
RD (A) TO   MODIFY BEHAVIOUR REL
ATED TO     EVERYDAY RELATIONSHI
PS OR NOT   (B)?": GO SUB 600
2130 PRINT "7) ON HOLIDAY DO YOU
 PREFER TO  SPEND TIME READING &
 WALKING (A)OR MEETING PEOPLE (B
)?": GO SUB 600
2140 PRINT "8) WOULD YOU FIND BE
ING A HERMITEASY (A), OR HARD (B
)?": GO SUB 600
2150 PRINT "9) WOULD YOU PREFER
TO MARRY A  THOUGHTFUL (A), OR S
OCIABLE     PERSON (B)?": GO SUB
 600
2160 PRINT "10) ARE MOST PEOPLE
GENERALLY   TRUSTWORTHY (B), OR
NOT (A)?": GO SUB 600
2170 PRINT "11) DO YOU LIKE ORGA
NISING      PARTIES (A), OR NOT
(B)?": GO SUB 600
2180 PRINT "12) WOULD YOU PREFER
 TO BE A    LIBRARIAN (A), OR A
 SALESPERSON(B)?": GO SUB 600
2190 PRINT "13) WOULD YOU DESCRI
BE YOURSELF AS CAUTIOUS (A) OR O
UT-GOING (B)": GO SUB 600
2200 PRINT "14) WOULD YOU LIKE T
O BE A CIVILSERVANT (A) OR IN TH
E GOVERNMENT(B)?": GO SUB 600
2210 PRINT "15) DO YOU ENJOY BIG
, NOISY     PARTIES (B), OR NOT
(A)?": GO SUB 600
2220 PRINT "16) WOULD YOU FIND I
T DIFFICULT TO MAKE A PUBLIC SPE
ECH (A), OR EASY (B)?": GO SUB 6
00
2230 PRINT "17) IN A THEATRE WOU
LD YOU LIKE TO BE A STAGE-HAND (
A), OR AN   ACTOR (B)?": GO SUB
600
2240 PRINT "18) DO YOU HAVE A RE
ADY REPLY   FOR MOST CONVERSATIO
N (B), OR   ARE YOU MORE RESERVE
D (A)?": GO SUB 600
2250 PRINT "19) ARE YOU SLOW (A)
, OR QUICK  (B) AT MAKING NEW FR
IENDS?": GO SUB 600
2260 PRINT "20) WOULD YOU DESCRI
BE YOURSELF AS BEING FULL OF ENE
RGY (B), OR NOT (A)?": GO SUB 60
0
2270 PRINT  FLASH 1;AT 10,7;"TES
T NO.1 OVER!"; FLASH 0: GO SUB 5
0
```

```
2280 CLS : PRINT "RESULTS OF P.T
EST NO.1:"; OVER 1;AT 0,0;"_____
_____"''
2285 PRINT "YOU ARE ";
2290 GO SUB 800
2300 GO SUB 50
2310 GO TO 35
3000 REM PERSON. TEST NO.2.
3010 PAPER 6: BORDER 6: INK 0: B
RIGHT 0: CLS : LET PT2B=0
3020 PRINT "PERSONALITY TEST NO.
2:"; OVER 1;AT 0,0;"_____
_____"''
3030 GO SUB 50
3040 PRINT "1) AS FAR AS YOU KNO
W HAVE YOU  EVER (A) WALKED IN Y
OUR SLEEP,  OR NOT (B)?": GO SUB
 650
3050 PRINT "2) HAVE YOU BEEN OFF
 WORK DUE TOILLNESS FOR A TIME P
ERIOD LONGERTHAN MOST PEOPLE (A)
, OR NOT (B)": GO SUB 650
3060 PRINT "3) DO YOU HAVE A TEN
DENCY TO    FEEL CONFUSED IF INT
ERRUPTED    WHILST WORKING (A),
OR NOT (B)?": GO SUB 650
3070 PRINT "4) DO YOU ENJOY SOME
 HARD       EXERCISE EVERY DAY (
A), OR NOT  (B)?": GO SUB 650
3080 PRINT "5) THE LAST TIME YOU
 BEGAN TO   LEARN A NEW SKILL DI
D YOU FEEL  CONFIDENT (B), OR NO
T (A)?": GO SUB 650
3090 PRINT "6) HAVE YOU FELT STR
ONGLY ABOUT EVERYDAY IRRITATIONS
 (A) , OR   NOT (B)?": GO SUB 65
0
3100 PRINT "7) HAVE YOU EVER WOR
RIED FOR    HOURS AFTER A SITUAT
ION WHERE   YOU FELT HUMILIATED
(A), OR NOT (B)?": GO SUB 650
3110 PRINT "8) WOULD PEOPLE REGA
RD YOU AS A SENSITIVE PERSON (A)
, OR NOT (B)": GO SUB 650
3120 PRINT "9) DO YOU USUALLY GE
T TO SLEEP  EASILY (B), OR NOT (
A)?": GO SUB 650
3130 PRINT "10) WOULD MANY PEOPL
E CONSIDER  YOU SHY (A), OR NOT
(B)?": GO SUB 650
3140 PRINT "11) DO YOU FEEL DIST
URBED IF    SOMEONE YOU KNOW FAI
LS TO GREET YOU (A), OR NOT (B)?
": GO SUB 650
3150 PRINT "12) DO YOU (A) SOMET
IMES FEEL   HAPPY OR SAD WITHOUT
 ANY REAL   CAUSE, OR NOT (B)?":
 GO SUB 650
3160 PRINT "13) AT WORK DO YOU O
```

```
FTERN FIND  YOURSELF DAY-DREAMIN
G (A), OR   NOT (B)?": GO SUB 65
Ø
3170 PRINT "14) CAN YOU (A) REME
MBER HAVING ANY NIGHTMARES IN TH
E LAST FIVE YEARS, OR NOT (B)?":
 GO SUB 650
3180 PRINT "15) HAVE YOU A REAL
FEAR OF     HEIGHTS/TUNNELS OR O
UT-DOORS    (A), OR NOT (B)?": G
O SUB 650
3190 PRINT "16) DO YOU USUALLY B
EHAVE CARMLYAND EFFICIENTLY IN A
N EMERGENCY (B), OR NOT (A)?": G
O SUB 650
3200 PRINT "17) ARE YOU A VERY E
MOTIONAL    PERSON DURING NORMAL
 SITUATIONS (A), OR ARE YOU NOT
(B)?": GO SUB 650
3210 PRINT "18) DO YOU (A) FREQU
ENTLY WORRY ABOUT YOUR HEALTH, O
R NOT (B)?": GO SUB 650
3220 PRINT "19) CAN YOU REMEMBER
 DEFINITELY ANNOYING SOMEONE THI
S YEAR (A), OR NOT (B)?": GO SUB
 650
3230 PRINT "20) DO YOU SWEAT WIT
HOUT DOING  MUCH EXERCISE (A), O
R NOT (B)?": GO SUB 650
3240 PRINT "21) CAN YOU REMEMBER
 YOUR MIND  GOING BLANK WHILST D
OING A JOB  DURING THE LAST FIVE
 YEARS (A), OR NOT (B)?": GO SUB
 650
3250 PRINT "22) WITHIN THE LAST
YEAR CAN YOUREMEMBER MEETING AT
LEAST THREE PEOPLE THAT YOU THOU
GHT WERE    DEFINITELY UNFRIENDL
Y TOWARDS   YOU (A), OR NOT (B)?
": GO SUB 650
3260 PRINT "23) HAVE YOU EVER (A
) BEEN SHORTOF BREATH WITHOUT DO
ING ANY     EXERCISE, OR NOT (B)
?": GO SUB 650
3270 PRINT "24) ARE YOU USUALLY
TOLERANT OF OTHER PEOPLE'S WAYS
(B), OR NOT (A)?": GO SUB 650
3280 PRINT "25) ARE THERE ANY NO
RMAL        SITUATIONS WHERE YOU
 FEEL       DEFINITELY SELF-CONC
IOUS (A), ORNOT (B)?": GO SUB 65
Ø
3290 PRINT "26) DO YOU OFTERN FE
EL UNHAPPY  (A), OR NOT (B)?": G
O SUB 650
3300 PRINT "27) HAVE YOU SUFFERE
D FROM      DIARRHOEA MORE THAN
ONCE IN THE LAST TWO YEARS (A),
OR NOT (B)?": GO SUB 650
3310 PRINT "28) ARE YOU USUALLY
SELF-       CONFIDENT (B), OR NO
T (A)?": GO SUB 650
3320 PRINT "29) DO YOU BELIEVE Y
OU CAN COPE WITH EVERYDAY SITUAT
IONS AS WELLAS ANYONE ELSE (B),
OR NOT (A)?": GO SUB 650
3330 PRINT "30) DO YOU USE ASPIR
IN/SLEEPING-TABLETS OR TRANQUILI
ZERS MORE   THAN ONCE A MONTH (A
), NO:(B)?": GO SUB 650
3500 PRINT  FLASH 1;AT 10,7;"TES
T NO.2 OVER!"; FLASH Ø: GO SUB 5
Ø
3510 CLS : PRINT "RESULTS OF P.T
EST NO.2:"; OVER 1;AT Ø,Ø;"_____
_____________________"''
3515 PRINT "YOU ARE ";
3520 GO SUB 900
3530 GO SUB 50
3540 GO TO 35
4000 REM IQ. RESULT.
4010 PRINT "I.Q. TEST RESULT:";
OVER 1;AT Ø,Ø;"____________________"
''
4015 IF SCORE=Ø THEN  LET IQ=80
4020 IF SCORE>Ø AND SCORE<=5 THE
N  LET IQ=85+(2*SCORE)
4030 IF SCORE>5 AND SCORE<=12 TH
EN  LET IQ=INT (95+(2*(SCORE-6))
)
4040 IF SCORE>12 AND SCORE<=18 T
HEN  LET IQ=INT (104+(2*(SCORE-1
2)))
4050 IF SCORE>18 AND SCORE<=25 T
HEN  LET IQ=114+(SCORE-18)
4060 IF SCORE>25 AND SCORE<=30 T
HEN  LET IQ=120+(SCORE-25)
4070 IF SCORE>30 AND SCORE<=35 T
HEN  LET IQ=125+(SCORE-30)
4080 IF SCORE>35 AND SCORE<=40 T
HEN  LET IQ=130+(SCORE-35)
4090 IF SCORE>40 AND SCORE<=45 T
HEN  LET IQ=135+(SCORE-40)-1
4100 IF SCORE>45 AND SCORE<=49 T
HEN  LET IQ=138+(SCORE-45)-1
4110 IF SCORE=50 THEN  LET IQ=14
2
4120 PRINT "YOUR SCORE IS ";SCOR
E;"/50"''
4125 PRINT "YOUR I.Q. RESULT IS
";IQ''
4130 IF SCORE>=40 THEN  PRINT "P
ERHAPS YOU SHOULD THINK ABOUT  A
PPLYING TO JOIN MENSA!          T
HAT WAS AN EXCELLENT SCORE."
4140 IF SCORE<40 AND SCORE>=30 T
HEN  PRINT "THAT WAS A GREAT SCO
RE,         WELL DONE! NOT QUITE
```

```
 UP TO MENSASTANDARDS BUT IN THE
 UPPER 10%  AREA OF THE POPULATI
ON."
4150 IF SCORE<30 AND SCORE>=25 T
HEN  PRINT "THAT WAS A VERY GOOD
 SCORE. IN  THE UPPER 15% OF THE
 POPULATION."
4160 IF SCORE<25 AND SCORE>=20 T
HEN  PRINT "GOOD SCORE. WELL ABO
VE THE      POPULATION AVERAGE."
4170 IF SCORE<20 AND SCORE>=15 T
HEN  PRINT "FAIR SCORE, JUST ABO
VE THE      POPULATION AVERAGE."
4180 IF SCORE<15 AND SCORE>=8 TH
EN  PRINT "AVERAGE SCORE, WITHIN
 THE 68% OFTHE POPULATION BRACKE
T."
4190 IF SCORE<8 AND SCORE>=5 THE
N  PRINT "POOR SCORE. BELOW THE
POPULATIONAVERAGE."
4200 IF SCORE<5 AND SCORE>=1 THE
N  PRINT "VERY POOR. WITHIN THE
LOWER 16% OF THE POPULATION RANG
E!"
4210 IF SCORE=0 THEN  PRINT "SUP
ER CRETIN! YOU GOT THEM ALL  WRO
NG."
4220 GO SUB 50
4230 RETURN
5000 REM IQ. TEST EXAMPLES.
5010 INK 0: PAPER 6: BORDER 6: B
RIGHT 0: CLS
5020 PRINT "I.Q. TEST EXAMPLES:"
; OVER 1;AT 0,0;"_______________
___"''
5030 PRINT "A) ANALOGIES:""DARK
 IS TO LIGHT AS X IS TO Y:  (";
FLASH 1;"BLACK"; FLASH 0;", TREE
, PLANT, "; FLASH 1;"WHITE"; FLA
SH 0;")"''
5040 PRINT "B) SIMILARITIES:"';
FLASH 1;"ENTIRE"; FLASH 0;", WID
E, EMPTY, "; FLASH 1;"WHOLE"; FL
ASH 0: PRINT
5050 PRINT "C) EQUATIONS:""21-6
=3*(A) "; FLASH 1;"5"; FLASH 0:
PRINT
5060 PRINT "D) LINKS:""INVOICE
(B**L) BEAK "; FLASH 1;"IL"; FLA
SH 0: PRINT
5070 PRINT "E) OPPOSITES:"'; FLA
SH 1;"TENSE"; FLASH 0;", TERSE,
SERIOUS, "; FLASH 1;"RELAXED"; F
LASH 0: PRINT
5080 PRINT "F) MID-TERMS:""FIRS
T (SECOND) THIRD : ONE (T**)THRE
E "; FLASH 1;"WO"; FLASH 0: PRIN
T
5090 GO SUB 50: CLS
5100 PRINT "G) SIMILAR/OPPOSITES
:"'; FLASH 1;"PUNISH"; FLASH 0;"
, REPUTE, REPLY, "; FLASH 1;"REW
ARD"; FLASH 0: PRINT
5110 PRINT "H) MATHS MID-TERMS:"
'" 11 (12) 13 : 4 (A) 6 "; FLASH
 1;"5"; FLASH 0: PRINT
5120 GO SUB 50
5130 GO TO 35
9990 REM SAVE ROUTINE.
9998 CLS : PRINT "SAVE:": SAVE "
I.Q. TEST." LINE 1: CLS : PRINT
"VERIFY:": VERIFY "I.Q. TEST.":
CLS : PRINT "O.K.": STOP
9999 REM LISTING OCCUPIES 34.5K.
```



Block Delete

Faster than many professional toolkits, this short utility allows easy deletion of blocks.

Most deletion routines only allow you to delete one line at a time; this one automatically deletes blocks as well as being quite short.

It is set up in the printer buffer and is therefore suitable for both 16K and 48K machines.

MONITOR ROUTINE

It calls the monitor routine at 6510h to get the address of the first line to be deleted and then again to get the address of the line following the last line to be deleted. These addresses are then passed to the monitor routine at 6629h, which performs the necessary deletion.

This produces a block deletion routine that is much, much faster than many professional toolkit programs!

IN FACTS

A few words on the IN function might be useful. When using this (except on issue 3 Spectrums) the value of BIT 7 is 1, which means the value returned as natural will vary between 255 and 191. Only the 5 least significant bits are relevant when testing for a key so:
LET X=IN 65022: LET X=X−32*INT(X/32).

Then test for lower values which will be the same regardless of whichever machine is being used by:
IF X=23 THEN...

HOW TO USE IT

The routine shown provides a simple, compact, easy to use way of deleting blocks of source lines from a BASIC program. To use it, key in the program and SAVE it. Lines 4 to 6 comprise the loader and data necessary to create the machine code. Key in 'GOTO 4 ENTER' and the machine code is POKEd into memory locations 23300 onwards (this is in the PRINTER buffer). If the flashing message 'Checksum error' is displayed, you have probably made a mistake in the DATA statement at line 6. Check it carefully, correct the error and key in 'GOTO 4 ENTER' again. When correct, the message "DELETE loaded" comes.

Now you can test the routine. Key in 'GOTO 1 ENTER' and the routine requests the line number at which you want to start to deleting. Key in '4 ENTER' and the routine will ask for the line number up to which you want to delete. Key in '6 ENTER' and the routine will delete lines 4 to 6.

You can now save the BASIC and m-c with the commands:

SAVE "delete"

and

SAVE "deletcode" CODE mem,20

Now you have a routine which occupies lines 1 to 3 of a BASIC program. To use it, make sure there are no lines 1, 2 or 3 in the program you want to edit. Key in:

MERGE "delete"

and the three lines of BASIC routine are merged with your program. Now key in 'GOTO 3 ENTER' to load the machine code. The routine will ask you which lines are to be deleted as already described.

There is no restriction on the line numbers which you enter, except that they must satisfy the tests in line 1. If you give a non-existent line as the start line, deleting will commence with the next valid line. If the last line number you give does not exist, deletion will include the last line before this number. This is the most compact way of using the routine. However, if you don't mind using more than three lines, you can do the following. Delete line 3. Change line 5 to 'GOTO 1'. Now the routine will initialise itself when you MERGE it and key in 'GOTO 4 ENTER'.

```
   1 INPUT "Delete from line ";f
;" to line ";t: IF f<3 OR t>9999
 OR f>t THEN  GO TO 1
   2 POKE 23296,f-256*INT (f/256
): POKE 23297,INT (f/256): POKE
23298,t-256*INT (t/256): POKE 23
299,INT (t/256): RANDOMIZE USR m
em: STOP
   3 LET mem=23300: LOAD "delete
code"CODE mem,20: GO TO 1
   4 LET mem=23300: LET cs=0: RE
STORE 6: FOR i=0 TO 18: READ v:
POKE (mem+i),v: LET cs=cs+v: NEX
T i: READ csv: IF csv<>cs THEN
PRINT  FLASH 1;"Checksum error":
 STOP
   5 PRINT "Delete is loaded": S
TOP
   6 DATA 42,0,91,205,110,25,229
,42,2,91,35,205,110,25,209,205,2
29,25,201,2081
```

```
5B04  2A005B   LD HL,(5B00)
5B07  CD6E19   CALL 196E
5B0A  E5       PUSH HL
5B0B  2A025B   LD HL,(5B02)
5B0E  23       INC HL
5B0F  CD6E19   CALL 196E
5B12  D1       POP DE
5B13  CDE519   CALL 19E5
5B16  C9       RET
```

D. ROBINSON

Redefinition

Use graphics that you've redefined in your own programs.

This program allows you to redefine 1152 graphics in blocks of 96 for use in your own programs. The data can be stored on tape and re-loaded whenever you require them. The graphics can be called up by POKEing the CHARS. system variable 23607.

POKEing RAM

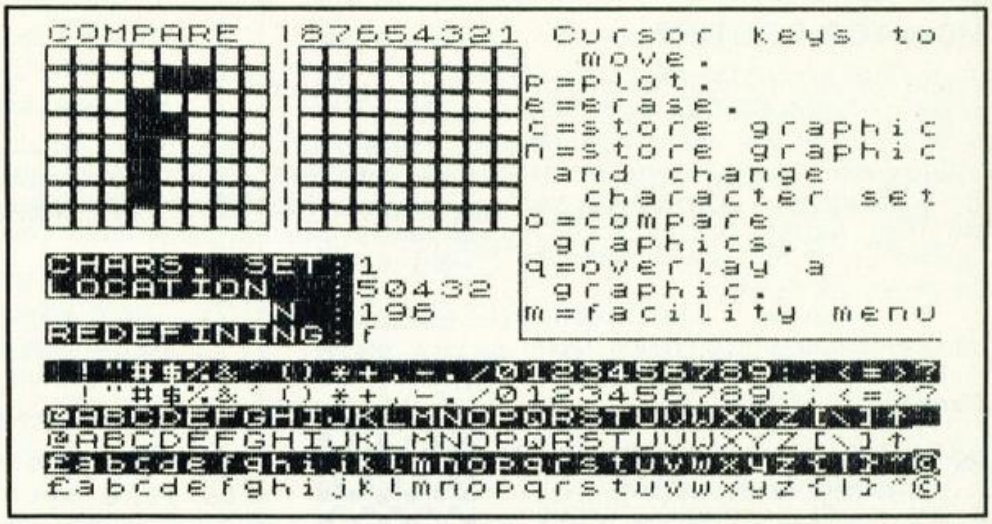
The data for the standard character set are POKEd into RAM and are then read from ROM locations 15616 to 16383. The character set can then be redefined in RAM and called into the program by POKE 23607, N where N is the start address of the newly defined set minus 256 and divided by 256. The first address of the graphics is 50432 leaving more than 27K of RAM for your own program plus the user defined graphics area.

FACILITIES

There are four facilities: character redefiner, with complete instructions and two grids so that one graphic can be compared with another; loader; saver and set call up. In the latter the redefined set appears in the program itself. Instructions are given in the program.

HOW IT RUNS

1-200	*Explanation*	*1000-1730*	*Define mode*
200-230	*Character set read from ROM to RAM*	*2000-2070*	*Load mode*
		3000-3050	*Save mode*
240-260	*Facility menu*	*4000-4030*	*Set call-up*




```
   1 REM XYGESOFT Jan-Mar '84              D C R
 100 POKE 23658,0: POKE 23607,60: CLEAR 5e4: BORDER 0: PAPER 0: INK 7: OVER 0: C
LS
 102 LET j$=" !""#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_#
abcdefghijklmnopqrstuvwxyz{|}"
 105 INPUT "Do you want an explanation ? "; LINE y$: IF y$="" THEN  GO TO 105
 107 IF y$(1)<>"Y" AND y$(1)<>"y" THEN  GO TO 210
 110 LET a$="   This program allows you to   design 1152 graphics in blocks  of
96 at a time for use in your own programs.                    This is achieved
 by firstly  POKEing the data for the charac-ter set from ROM(locations 15616 to
 16383)into RAM.Any amount ofsymbols,letters & numbers can   then be redefined i
n RAM."
 120 GO SUB 200
 130 LET a$="  This new character set can    then be adopted by:
                              POKE 23607,N
             N is the start address of the new character set minus 256 and div
ided by 256"
 132 LET a$=a$+                "                    23607 is the CHARS system var-i
able which is POKEd to point tothe new character sets in RAM."
 135 GO SUB 200
 140 LET a$="  For example,if 50432 was the  start address,then N would be:
  (50432-256)/256 = 196         Therefore,to use this new set located at 50432, P
OKE 23607,196  Any redefined graphics in thisset would then appear in the    pro
gram listing.                 "
 142 LET a$=a$+"  50432 is in fact the first    address where the new character
sets are to be stored.This meansyou still have 27K+ for the restof your own prog
ram."
 145 GO SUB 200
 150 LET a$="  If you wanted to return to thenormal set(located at 15616),   the
n N would be :                 (15616-256)/256 = 60        So,to return to t
he normal    character set, POKE 23607,60"
 155 GO SUB 200
 160 LET a$="  <NOTES ON THE DEFINE MODE>                                    I
f you choose the character   definer mode,four statements in INVERSE VIDEO will
appear:         CHARS. SET;this is the char-  acter set you are defining.      L
OCATION;this is the address  of the first byte in this char- set.
             "
```

```
 165 LET a$=a$+"  N;this is the number to be    POKEd into 23607 when you want
to use your set.                    REDEFINING;this is the charac-ter you are rede
fining."
 170 GO SUB 200
 180 LET a$="  <NOTES ON THE DEFINE MODE>                                      F
inally-                    "
 184 LET a$=a$+"2 character sets,one in INVERSE VIDEO,will appear at the bottom
of the screen.The inverse one isto be used for reference.In the other set,any re
defined chars.  will appear."
 190 GO SUB 200
 199 GO TO 210
 200 CLS : PRINT AT 0,5; PAPER 1; INK 6;"Character Redefinition"'': FOR f=1 TO L
EN a$: PRINT a$(f);: IF a$(f)<>" " THEN  BEEP .01,40
 202 NEXT f: PRINT #0;"PRESS A KEY...": PAUSE 0: BEEP .5,20: RETURN
 210 CLS : INPUT "How many character sets do you  want to redefine ? "; LINE b$:
 IF b$="" OR CODE b$<48 OR CODE b$>57 THEN  GO TO 210
 211 LET b=VAL b$: IF b<1 OR b>12 THEN  GO TO 210
 215 LET po=60: BEEP .5,22
 220 PRINT AT 0,0; FLASH 1; BRIGHT 1;"Please wait...";''''';"I'm POKEing at the m
oment..."
 225 DIM c(12): DIM h(8)
 226 LET h(1)=128: FOR f=2 TO 8: LET h(f)=h(f-1)/2: NEXT f
 230 FOR f=196 TO 196+(b-1)*4 STEP 4: LET c(f/4-48)=1: FOR g=15616 TO 16383: POK
E f*256+256+g-15616,PEEK g: NEXT g: NEXT f
 240: BORDER 0: PAPER 0: INK 7: CLS : PRINT TAB 9;"Facility Menu-": PRINT AT 2,0
;"1 > Define a new set";AT 4,0;"2 > Load a set";AT 6,0;"3 > Save a set";AT 8,0;"
4 > Call up a set"
 250 LET i$=INKEY$: IF i$<"1" OR i$>"4" THEN  GO TO 250
 260 GO TO 1000*VAL I$
1000 BEEP .5,20: BEEP .3,24: PAPER 6: OVER 0: BORDER 7: INK 1: CLS : PRINT AT 0,
0;"1 > DEFINER"
1002 FOR f=1 TO 6: POKE USR "a"+f,129: NEXT f: POKE USR "a",255: POKE USR "a"+7,
255
1005 POKE 23607,60: PRINT AT 15,0; INVERSE 1;j$(1 TO 32);''j$(33 TO 64)''j$(65 T
O 96)
1010 INPUT "Which set to be defined ?"; LINE z$: IF z$="" OR CODE z$<48 OR CODE
z$>57 THEN  GO TO 1010
1011 LET ch=VAL z$: IF ch>12 OR ch<1 THEN  BEEP .5,-30: GO TO 1010
1012 IF NOT c(ch) THEN  BEEP .5,-30: GO TO 1010
1015 PRINT AT 10,0;"CHARS. SET:";ch
1016 LET ch=196+(ch-1)*4
1018 PRINT AT 11,0;"LOCATION  :";ch*256+256
1022 POKE 23607,ch: PRINT AT 16,0;j$(1 TO 32)''j$(33 TO 64)''j$(65 TO 96)
1033 POKE 23607,po: INPUT AT 0,0;"Character to be redefined ? "; LINE c$: IF COD
E c$>127 OR CODE c$<32 OR LEN c$>1 THEN  BEEP .5,-20: GO TO 1033
1034 PRINT AT 13,0;"REDEFINING:";c$;AT 12,8;"N :";ch
1035 FOR f=1 TO 8: PRINT AT f,0;"□□□□□□□□";□□□□□□□□": NEXT f
1036 PRINT AT 0,0; INK 3; PAPER 7;"COMPARE 87654321"
1037 LET x=1: LET y=9: DIM a(8)
1038 DIM b$(8,8): LET k=(CODE c$-32)*8
1040 PRINT AT 0,18;"Cursor keys to";AT 1,19;"move.";AT 2,17;"p=plot.";AT 3,17;"e
=erase.";AT 4,17;"c=store graphic"
1041 PRINT AT 5,17;"n=store graphic";AT 6,18;"and change";AT 7,19;"character set
";AT 8,17;"o=compare";AT 9,18;"graphics.";AT 10,17;"q=overlay a";AT 11,18;"graph
ic.";AT 12,17;"m=facility menu"
1042 PLOT 135,104: DRAW 0,-38: DRAW 120,0
1045 OVER 1: INK 1: PRINT AT x,y;"■": BEEP .003,x+y: PRINT AT x,y;"■"
1046 LET i$=INKEY$
1050 LET y=y+(i$="8" AND y<16)-(i$="5" AND y>9): LET x=x-(i$="7" AND x>1)+(i$="6
" AND x<8)
1060 IF i$="p" AND b$(x,y-8)=" " THEN  PRINT AT x,y; OVER 0;"■": LET b$(x,y-8)=
"
■": LET a(x)=a(x)+2^16-y)
1065 IF i$="e" AND b$(x,y-8)="■" THEN  PRINT AT x,y; OVER 0;"□": LET b$(x,y-8)=
"
 ": LET a(x)=a(x)-2^(16-y)
1070 IF i$="o" THEN  GO SUB 1500
1075 IF i$="c" THEN  GO SUB 1200: GO SUB 1700: GO TO 1030
1080 IF i$="n" THEN  GO SUB 1200: GO SUB 1700: GO TO 1010
```

```
1085 IF i$="q" THEN  GO SUB 1600
1090 IF i$="m" THEN  CLS : BORDER 0: PAPER 0: OVER 0: INK 7: GO TO 240
1100 GO TO 1045
1200 OVER 0: FOR f=1 TO 8: POKE ch*256+256+k+f-1,a(f): NEXT f: RETURN
1500 INPUT "Graphic to be compared ?"; LINE g$: IF CODE g$<32 OR CODE g$>127 OR
LEN g$>1 THEN  BEEP .5,-20: GO TO 1500
1505 FOR f=1 TO 8: PRINT AT f,0; OVER 0;"□□□□□□□□": NEXT f
1510 FOR f=0 TO 7: LET gr=PEEK (ch*256+256+((CODE g$-32)*8)+f)
1515 FOR g=1 TO 8
1520 IF gr>=h(g) THEN  PRINT AT 1+f,g-1; OVER 0;"■": LET gr=gr-h(g)
1530 NEXT g: NEXT f
1540 RETURN
1600 INPUT "Graphic to be overlaid ?"; LINE g$: IF CODE g$<32 OR CODE g$>127 OR
LEN g$>1 THEN  BEEP .5,-20: GO TO 1600
1605 FOR f=1 TO 8: PRINT AT f,9; OVER 0;"□□□□□□□□": NEXT f
1607 DIM a(8)
1610 FOR f=0 TO 7: LET gr=PEEK (ch*256+256+((CODE g$-32)*8)+f)
1615 FOR g=1 TO 8
1620 IF gr>=h(g) THEN  PRINT AT 1+f,g+8; OVER 0;"■": LET gr=gr-h(g): LET a(f+1)
=
a(f+1)+h(g): LET b$(f+1,g)="■"
1630 NEXT g: NEXT f: RETURN
1700 LET k=CODE c$-32: IF k<32 THEN  LET xx=16: GO TO 1730
1710 IF k<64 THEN  LET xx=18: LET k=k-32: GO TO 1730
1720 IF k<96 THEN  LET xx=20: LET k=k-64
1730 POKE 23607,ch: PRINT AT xx,k;c$: POKE 23607,po: RETURN
2000 BEEP 1,10: BORDER 6: PAPER 6: INK 0: OVER 0: CLS : PRINT AT 0,0;"2 > LOADER
"
2010 PRINT AT 4,1;"What is the set's name ?          (ENTER if name not known)"
2020 INPUT d$
2030 PRINT AT 7,0;: LOAD d$CODE
2040 FOR f=196 TO 244 STEP 4: IF NOT PEEK (f*256+255) THEN  NEXT f
2045 POKE f*256+255,0
2050 PRINT AT 8,0;"You have LOADed character set ";f/4-48;AT 10,0;"  To call thi
s up in your own   program,use POKE 23607,";f;AT 13,0;"  This set is located at
";f*256+256
2055 LET c(f/4-48)=1
2060 PRINT AT 16,0;"CHARACTER SET:";f/4-48: POKE 23607,f: PRINT j$: POKE 23607,p
o
2070 PRINT #0;"ANY KEY FOR FACILITY MENU...": PAUSE 0: BEEP .2,13: BEEP .3,11: G
O TO 240
3000 BEEP 1,12: BORDER 4: PAPER 6: INK 0: OVER 0: CLS : PRINT AT 0,0;"3 > SAVE A
 SET"
3010 INPUT "Which character set do you want to save ? "; LINE c$: IF CODE c$<48
OR CODE c$>57 THEN  BEEP .5,-30: GO TO 3010
3012 LET c=VAL c$: IF c<1 OR c>12 THEN  BEEP .5,-40: GO TO 3010
3013 IF NOT c(c) THEN  BEEP .6,-22: GO TO 3010
3020 PRINT AT 2,0;"Saving set ";c
3025 LET s$="Chars. "+STR$ c
3027 LET c=(c+48)*4
3030 POKE c*256+255,1: SAVE s$CODE c*256+255,769:
3040 BEEP .6,2: PRINT AT 4,0;"Verifying...": PRINT AT 6,0;: VERIFY s$CODE : BEEP
 .8,5: PRINT AT 7,0;"Code has verified    "
3045 PRINT AT 10,0;"Do you want to save it again ?": INPUT y$: IF y$(1)="y" THEN
  GO TO 3027
3048 PRINT AT 10,0;"Do you want to SAVE another     character set ?": INPUT y$:
IF y$(1)="y" THEN  GO TO 3e3
3050 GO TO 2070
4000 BEEP 1,22: PAPER 5: BORDER 6: INK 0: OVER 0: CLS : PRINT AT 0,0;"4 > SET CA
LL-UP"
4010 INPUT "Which set do you want to work   with ?"; LINE c$: IF CODE c$<48 OR C
ODE c$>57 THEN  BEEP .5,-35: GO TO 4010
4012 LET b=VAL c$: IF b=0 THEN  LET po=60: POKE 23607,po: GO TO 2070
4015 IF b<1 OR b>12 THEN  BEEP .5,-22: GO TO 4010
4016 IF NOT c(b) THEN  GO TO 4010
4020 PRINT AT 2,0;"  If at any time you want to    return to the standard charac
terset,choose this facility and askto work with set 0.                Don't forge
t redefined chars. will appear in the program      listing !!!"
4030 LET po=(b+48)*4: POKE 23607,po: GO TO 2070
```

DAVID NOWOTNIK

Slogo

A BASIC simulation to introduce you to the world of Logo.

The aim of this article is to introduce the programming language of LOGO. I'm sure many readers will want to try LOGO on their Spectrums. But, if you've tried to find some commercial software which allows you to use LOGO then you might well have been disappointed; there isn't much available. As a remedy to that situation, I shall be presenting, in three parts, a BASIC program which simulates LOGO.

PART THE FIRST

In each part, you'll get a BASIC program to type into your Spectrum, and instructions on how to use the program. And, in using the program, you'll be using LOGO. So, by the time the series is complete, you should have a good understanding of the language. The program I'll be presenting later must be added or MERGed to program one, then the third added to the other two. So, while this first program will fit a 16K Spectrum, you'll need a 48K Spectrum to accommodate the final version.

Ideally, a LOGO translation program should be written in machine code for a fast operating speed. I've written my version in BASIC as it's easier for me to write it, and easier for you to type in. But in using BASIC, speed is lost — that's why I've called my version of the language SLOGO!

Once you have entered the first listing and SAVEd the program on tape, you are ready to start. But before we begin typing in our LOGO commands, it might be useful to consider the background and principles of this language.

LOGO THEORY

LOGO was designed to provide children with an early introduction to computer programming and to develop their abilities in logical thinking. The commands of LOGO are simple to understand, yet LOGO encourages a good programming style by virtue of its structure. LOGO is best known as a graphics language, enabling still (and animated) pictures and patterns to be drawn. More powerful versions for text handling as well as graphics but my version, in common with most, deals with graphics only.

You'll find LOGO a very useful language if you have young children, if you want a versatile graphics routine or if you want to move on from BASIC, LOGO is an easy language to learn, yet it introduces you to structured programming, as used by more powerful languages such as FORTH.

TURNING TURTLE

Drawing with LOGO makes use of a turtle — but before you rush off with complaints to the RSPCA, these are only imaginary animals! Imagine you had precise control over the movement of your turtle; instructing it to go forward or back, turn left or right, all by specific amounts. Your turtle carries a pen which you can instruct to be lowered onto a sheet of paper, so that, as the turtle moves, a line is drawn. Using combinations of the four movement commands (forward, back, left and right) drawings and patterns can be created.

Some versions of LOGO actually make use of a robotic 'turtle' which is interfaced to a computer. LOGO commands are transmitted through to the turtle, creating shapes on a sheet of paper placed on the floor. More often, the monitor or TV screen forms the paper and an electronic turtle is drawn and moved, on the screen. This is how my version works.

LISTING 1

Enter Listing 1 and then RUN the program and you'll see the turtle on the screen. It is represented by the "^"; the point is its head and this indicates the direction of the turtle. You'll see that the turtle's starting point is at the centre of the screen, heading directly up the screen.

At the bottom of the screen you'll see "W;" and a flashing cursor. This tells you that the computer is waiting for you to enter a command. At this stage, this LOGO program will accept only 8 commands as shown in Fig.2. Note that all commands are entered in capital letters (CAPS LOCK mode is automatically set by the program) and for some commands, a two letter abbreviation can be used instead of the full word. Before you use any command, I'll describe what each one will do.

LET'S GO

Now let's try a few examples. Type in:

FORWARD 40 (or FD 40), and press enter. You'll see the turtle disappear, a line 40 pixels long will be drawn, and then the turtle reappears. To enable the program to work as quickly as possible, our turtle will always disappear when all instructions are complete and the "W;" symbol comes back on the screen. Also, notice that when the program starts, the pen is in the down position, enabling a line to be drawn.

Now, lets turn the turtle 90° to the right. Type in:

RIGHT 90 (or RT 90) and press enter.

Then move forward again:

FD 40.

If you continue entering alternatively RT 90 and FD 40, you would end up with a square. As it is cumbersome to type in one command at a time, LOGO allows you to string commands together. Let's see how. First, clear the screen, and reset the turtle using the command DRAW, then enter:

FD 40 RT 90 FD 40 RT 90 FD 40 RT 90 FD 40 RT 90 and enter.

Leave a single space between each command and number. This, again, draws a square. You may have noticed that we are repeating the same two commands four times. Fortunately, LOGO allows repetition of commands to be simplified and it's one of the things I'll be dealing with later.

OFF SCREEN

What should happen if the turtle goes off the screen? Let's find out. ▷

Clear the screen with DRAW, then enter:

RT 10 FD 2000.

The turtle is turned slightly to the right, then moves forward 2000 pixels. As it disappears off the edge of the screen, our turtle reappears on the other side. This feature is called wrap. It is useful to draw special patterns and means that you won't get an error message if you accidentally go off the edge of the screen.

Another feature of this LOGO program is that the specified number used by FORWARD, BACK, RIGHT and LEFT can be replaced by a random number. As an example, try:

FD RANDOM 50

The turtle will move forward by an amount somewhere between 0 and 50.

That's enough of the theory of LOGO for now. Try making up some drawings, patterns, or shapes for yourself. The program has an error trapping routine, so if you make a mistake, you should get a reasonably friendly error message. In the next part, you'll notice a quantum leap in the power of this LOGO translater.

PART THE SECOND

I'll now expand the number of instructions you can give to the turtle, turning this program into a quite powerful and versatile version of LOGO.

Take a look at the table, which gives you a list of all the extra LOGO commands that will be introduced to you in this section. But before you can get the program in part one to accept any of these new commands, you'll have to add the second program lines to those of the original program. Here is the best way of doing that:

ADDITIONAL LOGO

First type into your Spectrum the program lines. Then, save this routine on tape. Load into your Spectrum the program I gave you earlier and MERGE in the new routine just saved on tape. This may seem a complicated way of adding these new lines, but it will help if you have made any mistakes in typing in.

If you find your expanded program does not work smoothly, then check through your second program — it should match exactly the listing. As I warned earlier, the merged programs will only fit a 48K Spectrum, so I'm afraid you'll only be wasting your time if you have a 16K machine.

EXTRA STUFF

When you RUN the program, you should get the same response as before; a "W;" prompt message at the base of the screen, and the turtle ("A") at the centre of the screen. If you can recall, we have already used the commands FORWARD, BACK, LEFT and RIGHT to move the turtle, and DRAW, HOME, PENUP and PENDOWN as additional control instructions. Let's have a look at the first six of the new instructions, which also happen to be the simplest.

So far, all our drawings have been made using black 'ink' on white 'paper'. The command PENCOLOUR (or PC) allows you to change coours at anytime. To complete the PENCOLOUR command you have to add a number, 0 to 7, according to the number keys on your Spectrum. For example, PENCOLOUR 1 will change the foreground colour to blue for subsequent movements of the turtle.

I have already introduced the concept of wrap. This is when an instruction to the turtle causes it to 'fall off' the edge of the screen. with wrap, the turtle will re-appear on the opposite side of the screen, and continue on its way. You can forbid the turtle wrap round using the command NOWRAP. After using NOWRAP, if you accidentally command the turtle to go off the edge of the screen, you will get an error message. To reinstate the wrapping facility, use the command WRAP.

SET IN YOUR WAYS

The last three instructions of the first group are very sraightforward. COPY will produce a copy of your screen display on the printer, SAVE allows you to save the program (I will deal with creating a LOGO program soon) on tape, and STOP simply stops the LOGO program, and hands you back to BASIC. Remember with all LOGO commands to type them out in full (do not use the equivalent BASIC keywords — LOGO won't recognise them), or use the two letter abbreviation, if one is available.

The next group of instructions all start with SET, and they move the turtle in a precise way. If you look in your Spectrum handbook (on page 121) you'll find out how high resolution graphics are created. There are 176 dots from top to bottom of the screen, and 256 from left to right. Each one of these dots can be defined by a co-ordinate (like a place can be fixed by a map co-ordinate). The dot at the bottom left of the screen has the co-ordinate 0,0 and the one at top right 255,175. If you move vertically up or down the screen, you are moving horizontally left and right, you move along the x axis.

SETX APPEAL

All that may seem horribly mathematical, but I hope it will be clearer when we use the SET instructions to move the turtle from wherever it is SETX will move the turtle along the x-axis (horizontally left or right) to a specified point. Thus, SETX 10 will move the turtle from where ever it is on the screen horizontally to a point which is 11 dots (remember the first has the number 0) from the left hand edge of the screen. Similarly SETY 10 will move the turtle vertically to a point 11 dots from the bottom edge of the screen, not counting the command line. If you move the turtle with the pen down, a line will be drawn. Try this example. Reset the turtle with the DRAW instruction, then enter the command:

SX 60 SY 140 SX 128 SY 88 (and press ENTER)

This will draw horizontal and vertical lines to create a rectangle.

The command SETXY will move the turtle to the coordinates specified after the command. You'll need two numbers after SETXY, the first is the x-coordinate, the second, the y-coordinate. For example, with the rectangle from the above routine still on the screen, enter:

SETXY 60 140 (or XY 60 140) and press ENTER

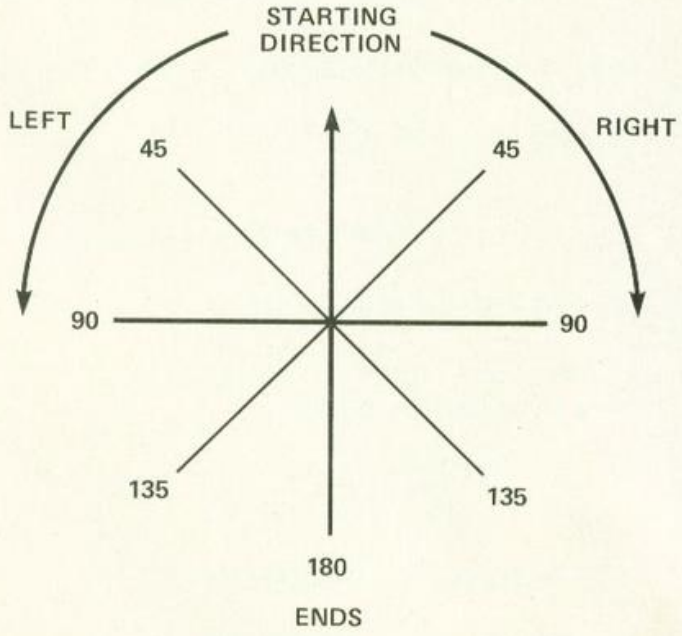
This will draw a diagonal to the rectangle, taking the turtle to the coordinates 60, 140 which is one corner (the top left corner) of the box.

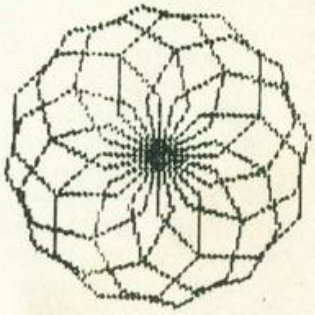
Try a number of SETX, SETY and SETXY instructions for yourself. The X number must be between 0 and 255, and the Y number between 0 and 175.

DIRECTION

You may have noticed that, in using these SET instructions that the turtle's direction (or heading) is not changed from that before the SET instruction was used. There is an additional instruction to alter the heading of the turtle to a definite direction; this is the SETHEADING command. SETHEADING 0 will direct the turtle to point vertically up the screen; SETHEADING 270

```
 110 DATA 18,24,1
 140 DATA "LI",6100,"BG",6500
 240 DATA "LIST",6100,"BACKGROUN
D",6500
6100 REM   LIST
6105 IF def<1 THEN  RETURN
6110 PRINT #1;"LIST - To Screen
or Printer?"
6115 LET z$=INKEY$: IF z$="" THE
N  GO TO 6115
6120 IF z$="S" THEN  LET li=1: G
O TO 6200
6125 IF z$="P" THEN  LET li=2: G
O TO 6140
6130 GO TO 6115
6140 CLS : PRINT "Printer listin
g - please wait"
6145 OPEN #2,"p"
6150 FOR j=1 TO def: GO SUB 6300
6155 PRINT : NEXT j
6160 OPEN #2,"s": RETURN
6200 FOR j=1 TO def
6205 CLS : PRINT #1;"Please Wait
"
6210 GO SUB 6300
6215 IF INKEY$="" THEN  GO TO 62
15
6220 NEXT j: RETURN
6300 LET ed=j: GO SUB 5805
6305 LET y$=w$(j+n)
6310 PRINT "Definition - ";y$''
6315 GO SUB 5955: RETURN
6500 REM   BACKGROUND
6505 GO SUB 1200: IF err>0 THEN
 RETURN
6510 IF a<0 OR a>31 THEN  LET er
r=2: RETURN
6515 LET col=a
6520 GO SUB 1200: IF err>0 THEN
 RETURN
6525 IF a<0 OR a>21 THEN  LET er
r=2: RETURN
6530 LET row=a
6535 GO SUB 1200: IF err>0 THEN
 RETURN
6540 IF a<0 OR (a+col)>31 THEN
LET err=2: RETURN
6545 LET width=a
6550 GO SUB 1200: IF err>0 THEN
 RETURN
6555 IF a<0 OR (a+row)>21 THEN
LET err=2: RETURN
6560 LET height=a
6565 GO SUB 1200: IF err>0 THEN
 RETURN
6570 IF a<0 OR a>7 THEN  LET err
=2: RETURN
6575 FOR i=row TO row+height
6580 FOR j=col TO col+width
6585 LET at=22528+32*i+j
6590 LET ll=PEEK at
6595 LET lk=INT (ll/8): LET ll=l
l-8*lk
6600 POKE at,ll+8*a
6605 NEXT j: NEXT i
6710 RETURN
```





will point the turtle horizontally towards the left edge of the screen. The heading you give to the turtle must be between 0 and 360.

If you are getting confused between the SETHEADING and LEFT and RIGHT commands then remember, SETHEADING gives you *absolute* command of the heading whereas LEFT and RIGHT turn the turtle *relative* to the turtle's direction. ▷

One final piece of theory for this part before we try a few more examples. If you remember we created a square with the instructions:

FD 40 RT 90 FD 40 RT 90 FD 40 RT 90 FD 40 RT 90

You have probably noticed that the two commands FD 40 and RT 90 are repeated 4 times. There is a command in LOGO which helps in repetition. Quite logically that command is REPEAT (RP for short), Here's how it is used in drawing a box. Clear the screen with the DRAW command, then enter the command:

RP 4 (FD 40 RT 90) and press ENTER

The number after REPEAT (or RP) is the number of times the commands within the square brackets are to be repeated. Remember, always leave a single space between commands, numbers, and square brackets. To be complete, the REPEAT command, *must* be followed by a number, then an 'open' square bracket. A 'close' square bracket indicates the end of the repeat loop.

NESTED LOOPS

In the same way that FOR-NEXT loops can be 'nested' in BASIC, so can REPEAT loops be nested. As an example, the following LOGO instruction will produce a symmetrical pattern.

RP 12 (RT 30 RP 8 (RT 45 FD 20))

Nesting of loops is easier in LOGO than in BASIC, and as long as you have the same number of "("as")", then you shouldn't go too far wrong!

The above command can be quite simply varied to produce a variety of symmetrical shapes. If you look at the command, I've linked together (under the command) two sets of two numbers. If you multiply the first two together you should get 360, and if you multiply the second you'll also get 360. And that is the trick in getting symmetrical patterns. You can replace any pair of numbers with another pair, so long as the new numbers in the pair, when multiplied together, produce 360. Why not try a few, and see what shapes you can produce.

Here are 3 other pattern drawing routines you may like to try:

DRAW RT 20
RP 5 (FD 40 RT 135 FD 40 LT 63)

DRAW PU XY 80 60 PD
RP 8 (FD 40 RT 45) RT 45
RP 8 (FD 96 RT 135)

DRAW PC 1 RP 3 (FD 40 RT 120)
PC 2 RP 3 (BK 40 LT 60)
LT 90 PC 3 RP 3 (FD 40 RT 120)
PC 4 RP 3 (BK 40 LT 60)

Type in one line at a time, pressing ENTER when you get to the end. Try and follow what is happening on the screen, and relate it to the instruction you have just entered. You'll find that you should soon understand most of the LOGO commands I've introduced so far.

STORAGE

Up till now, we have simply entered commands to the computer, and the Spectrum has obeyed them immediately, then the command is forgotten. It would be far more useful for the computer to store instructions and allow us to recall them at will. Our LOGO translater will then be operating on a LOGO program.

PART THE LAST

To incorporate the program in this section with the version you completed last time, first enter the listing into your computer, then store it on tape. Then, load into your Spectrum the program completed last time (from parts 1 and 2), then MERGE this program into it. Finally, save a copy of the completed program, then you're ready to RUN.

NEW COMMANDS

There are two additional commands

```
Definition - SKY

Ø  BG Ø Ø 31 5 5

Definition - SUN

Ø  PU XY 4Ø 16Ø PC 6 PD

1  SX 42 PU XY 38 159 PD

2  SX 44 PU XY 37 158 PD

3  SX 45 PU XY 37 157 PD

4  SX 45 PU XY 38 156 PD

5  SX 44 PU XY 4Ø 155 PD

6  SX 42 PU


Definition - GROUND

Ø  BG Ø 6 31 14 4


Definition - HOUSE

Ø  BG 2Ø 1Ø 5 5 2


Definition - DOOR

Ø  BG 22 13 1 2 1


Definition - WINDOWS

Ø  BG 21 11 Ø Ø 7
```

created by the extra routines in this part. The first is BACKGROUND (BG), which allows you to define the paper colour in specified areas of the screen. Five numbers are required after BACKGROUND to complete the command. Here is an example which you can try:

BACKGROUND 0 0 31 6 5

The first four numbers define the area, a rectangle, to be "painted", and the last number is the colour (0 to 7, as shown on the Spectrum keyboard). The first two numbers are the x and y co-ordinates of the top left point of the area to be coloured. In the example, 0 0 refers to the top left of the screen, unlike the co-ordinates for SET which start at the bottom of the screen. In fact the first two BACKGROUND co-ordinates are the same as the BASIC PRINT AT. The third and fourth numbers of the BACKGROUND command are width and height of the square to be coloured. In the example the number 31 is the full screen width, and the number 6 means six character squares down. The number 5 is the colour cyan, and so, the example given will "paint" a cyan block at the top of the screen. In an example later on, the same command will create the sky in a scene which will be created with LOGO commands.

The other new command in this part is LIST. It will list to screen or printer a LOGO program. But, as I haven't yet explained how to create a LOGO program, you won't yet have anything to list!

```
1  BG 24 11 Ø Ø 7

Definition - ROOF

Ø  PU XY 16Ø

1  96 PC Ø

2  PD SH 45 FD 33 RT 9Ø FD 33

Definition - SCENE

Ø  SKY SUN

1  GROUND TREE

2  HOUSE ROOF DOOR WINDOWS

3  PU XY 2ØØ 16Ø SH 18Ø

4  PD

Definition - TREE

Ø  PU XY 8Ø 1ØØ PC Ø PD

1  SH 9Ø RP 8 [ FD 7 LT 45 ]

2  FD 2

3  RT 9Ø FD 3Ø LT 9Ø FD 1

4  LT 9Ø FD 3Ø RT 9Ø FD 1

5  RT 9Ø FD 3Ø
```

DEFINING LOGO COMMANDS

LOGO programs are made by creating new commands from the commands that LOGO already understands. You create a new command using DEFINE (DF). Most versions of LOGO use TO instead of DEFINE, but I've used the latter as it is more explanatory. You complete the DEFINE command with the new command name, which must be different from all the commands currently available to LOGO. As an example, let's tell the computer how to draw a box; we'll define a command called BOX. First enter the command:

DEFINE BOX (or DF BOX)

The screen will clear after the computer has checked that the command BOX doesn't already exist. You'll get the message 'DEFINE BOX' at the top of the screen, and the usual 'W;' at the base of the screen.

There is no standard way of defining new commands in LOGO; there are probably as many different ways as there are versions of LOGO. So the instructions that follow just happen to be the way I have decided to allow the definition of new commands.

To define our box, enter the command:

RP 4 (FD 40 RT 90)
and press ENTER.

That line will appear at the top of the screen with a line number of zero. The line number is not used by LOGO; I've just added it to identify lines in case we want to make any changes. In any one definition, you can enter up to 10 lines, but no line can be longer than 20 characters. For our BOX definition, all we need is the one line already entered, so we tell the computer we've finished by entering END. You'll get a message to tell you that the new command BOX has been stored, then you're back to a clean sheet of paper, and the 'W;' symbol. The LOGO program will now accept BOX as a command; try it!

You can include in your definitions already defined commands. As an example, define another command PATTERN, as follows:

DEFINE PATTERN
RP 8 (BOX RT 45)
END

When complete, enter PATTERN as a direct command; you'll get a pattern based on the BOX routine you defined. ▷

Now define MOVE:

```
DEFINE MOVE
PU XY 50 PD BOX
PU SX 120 PD BOX
PU SX 190 PD BOX
END
```

Define the command as before, then enter MOVE as a direct command.

EDITING COMMANDS

The pattern created by PATTERN wasn't particularly exciting; we could change PATTERN to improve it. To do this we use the command EDIT. The syntax is EDIT PATTERN. Once you have entered this command, the computer will spend a few moments searching for PATTERN, then reformatting, ready for changes to be made. The definition will reappear on the screen as you entered it (apart from END). At the bottom of the screen is a menu of editing options.

- Option 1 is EDIT. It will allow you to change a line. Enter the line number of the line you wish to change, and re-enter the line as you want it.
- Option 2, INSERT allows you to insert another line between two lines of the screen. Remember, you cannot exceed a total of ten lines, so you can insert a line if you have 9 lines or less in your definition. If you opt to insert a line, say number two, then the previous line two becomes number 3, 3 becomes 4, and so on.
- DELETE (option 3) allows you to delete a line.
- REMOVE (option 4) will remove the whole definition. Pressing 5 will return you to normal command ('W:').

So, to EDIT the command PATTERN, you will want to change line zero. Press 1 to get the EDIT option, and press 0 to indicate that it is line 0 which you want to replace. Then enter:

```
PR 12 (BOX RT 30)
```

Press 5 to exit the edit routine, enter the direct command DRAW to clear the screen and reset the turtle, then try PATTERN again.

LOGO STRUCTURES

By now you should already have an ideal of how programs are built up in LOGO. Each definition should be quite independent; you should check it out before moving on to the next definition. This is structured programming. It has the advantage of being easier to follow what is happening (than, for example, 'unstructured' BASIC), so it should be easier to correct any mistakes. Programs written in this way are also much easier for others to understand. The BBC machine's PROCEDURE and the QL's DEF PRO also allow programs to be structured in a similar way.

To start you off in LOGO programming look at the listing, obtained by the LIST command. When entering a LOGO program, remember to enter one definition at a time (end each with END, which is not shown in the listing), then test it and edit as necessary before moving onto the next. In the example program, notice that the command SCENE is the command which uses all the other commands in its definition. It is the command which is central to the operation of the program; you operate the whole program (when it's all in the computer) by entering the direct command SCENE. In this way LOGO differs from BASIC. Its programs will not start with a single RUN command; they start with a defined command which is the 'core' of the program.

There is more to LOGO than the aspects I have covered in this series.; LOGO also uses variables; it allows decision making with IF...THEN ...ELSE structures; and it permits text handling. If you want to know more about LOGO there are several good books available (for example LOGO Programming by Peter Boss). Despite the limitations of my program, I hope you have gained an insight into the fascinating possibilities that LOGO is able to offer as an introduction to programming to young and old, or simply, as an easy to use graphics creation package.

COMMANDS

HOME	*This moves the turtle from anywhere on the screen back to its starting position at the centre of the screen, heading directly up the screen.*
DRAW	*This clears the screen, then moves the turtle HOME.*
FORWARD	*The turtle is moved forward a specific amount. To complete the command, you have to tell the turtle how far to move forward, eg FORWARD 20 (or FD 20). The unit of distance is one pixel; remember the screen is made up of 256 pixels across, and 176 down.*
BACK	*The turtle is turned head-to-tail, then moved in the same way as FORWARD (eg BACK 25).*
LEFT	*The turtle's head remains in the same place but its body will swivel so that its direction rotates to the left. You have to tell the turtle how far to turn. As do FORWARD and BACK, the command LEFT requires a number added to it to complete the instruction, to tell the turtle how much to turn. This number is the angle of turn in degrees. If you've forgotten that 90 degrees makes a right angle, then using Fig 3 as a guide when using the left (or RIGHT) command. Angles should be whole numbers between 0 and 360.*
RIGHT	*This is the same principle as LEFT, but, of course, the turtle turns right instead of left.*
PENUP	*This command raises the pen from the paper so that the turtle can be moved without a line being drawn.*
PENDOWN	*The pen is placed on the paper.*

COMMANDS AND ABBREVIATIONS

SECTION 1:			
DRAW	—	*NOWRAP*	*NW*
HOME	—	*COPY*	—
FORWARD	*FD*	*SAVE*	—
BACK	*BK*	*STOP*	—
LEFT	*LT*	*SETX*	*SX*
RIGHT	*RT*	*SETY*	*SY*
PENUP	*PU*	*SETHEADING*	*SH*
PENDOWN	*PD*	*REPEAT*	*RP*
SECTION 2:		**SECTION 3:**	
PENCOLOUR	*PC*	*BACKGROUND*	*BG*
WRAP	*WR*	*LIST*	*LI*

```
  5 REM ************************
 10 REM *                      *
 15 REM *        SLOGO         *
 20 REM *                      *
 25 REM *  by David Nowotnik   *
 30 REM *    January, 1984     *
 35 REM *                      *
 40 REM ************************
 45 REM
 50 REM      Initialise
 55 REM
 60 READ m: READ n: READ o
 65 DIM x$(m,2): DIM w$(n,12):
DIM f$(o,12)
 70 DIM u(m): DIM v(n): DIM g(o
)
 75 LET pp=1: LET wr=1: LET tur
t=1
 80 FOR i=1 TO m: READ x$(i)
 85 READ u(i): NEXT i
 90 FOR i=1 TO n: READ w$(i)
 95 READ v(i): NEXT i
100 FOR i=1 TO o: READ f$(i)
105 READ g(i): NEXT i
110 DATA 6,8,1
115 DATA "FD",3000,"BK",3200,"L
T",3400,"RT",3600
120 DATA "PU",4000,"PD",4050
200 DATA "FORWARD",3000,"BACK",
3200,"LEFT",3400
205 DATA "RIGHT",3600,"DRAW",38
00,"HOME",3850
210 DATA "PENUP",4000,"PENDOWN"
,4050
300 DATA "RANDOM",8000
400 POKE 23658,8
480 GO SUB 3800: GO TO 2000
490 REM
500 REM   Error subroutines
510 REM
520 GO SUB (690+err*10)
530 PRINT #1;a$
540 PAUSE 250
550 RETURN
700 LET a$="Command error - re-
enter the    line.": RETURN
710 LET a$="Number error - re-e
nter the line": RETURN
720 LET a$="No wrap - the line
cannot be    drawn": RETURN
990 REM
1000 REM   Subroutines
1010 REM
1020 REM   Entry check
1030 LET t=0: LET y$=""
1040 LET s=s+1: IF s>LEN z$ THEN
  LET t1=1: RETURN
1050 IF (z$(s)=" " OR z$(s)=CHR$
 0) AND t=0 THEN  GO TO 1040
1060 IF (z$(s)=" " OR z$(s)=CHR$
 0) AND t=1 THEN  RETURN
1070 LET y$=y$+z$(s): LET t=1: G
O TO 1040
1080 REM
1100 REM  Draw/Erase turtle
1110 IF x<3 OR x>252 OR y<3 OR y
>172 THEN  RETURN
1120 FOR j=160 TO 200 STEP 40
1130 LET q=dir+j: IF q<0 THEN  L
ET q=360-j
1140 IF q>360 THEN  LET q=q-360
1150 LET q=q*PI/180
1160 LET x1=5*SIN q: LET y1=5*CO
S q
1170 PLOT  INVERSE turt;x,y
1180 DRAW  INVERSE turt;x1,y1
1190 NEXT j: RETURN
1200 REM
1210 REM  A number or a function
1220 GO SUB 1020: IF err=1 THEN
 LET err=2: RETURN
1230 IF CODE y$>57 THEN  GO TO 1
300
1240 IF y$="" THEN  LET err=2: R
ETURN
1250 FOR k=1 TO LEN y$
1260 IF CODE y$(k)<48 OR CODE y$
(k)>57 THEN  LET err=2
1270 NEXT k
1280 IF err=0 THEN  LET a=VAL y$
1290 LET t=0: RETURN
1300 REM  Evaluate a function
1310 LET y$=(y$+"              ")(
 TO 12)
1320 FOR i=1 TO o: IF y$=f$(i) T
HEN  GO TO 1340
1330 NEXT i: LET err=2: RETURN
1340 GO SUB g(i): RETURN
1400 REM  X Wrap
1410 LET cr=1: IF q<2*PI AND q>P
I THEN  LET cr=-1
1420 IF cr=1 THEN  LET x1=255-x:
 IF q>1.57 THEN  LET y1=0: LET x
2=0: GO TO 1430
1425 IF cr=1 THEN  LET y1=x1/TAN
 q: LET x2=0
1430 IF cr=-1 THEN  LET x1=-x: L
ET y1=(x1/TAN (q-PI)): LET x2=25
5
1440 LET y2=y+y1: LET a=a-INT SQ
R (ABS (x1)^2+ABS (y1)^2)
```

```
1450 RETURN
1470 REM
1500 REM Y Wrap
1510 LET cr=1: IF q>PI/2 AND q<3
*PI/2 THEN  LET cr=-1
1520 IF cr=1 THEN  LET y1=175-y:
 LET x1=y1*TAN q: LET y2=0
1530 IF cr=-1 THEN  LET y1=y: LE
T x1=-(y1*TAN (q-PI)): LET y2=17
5: LET y1=-y
1540 LET x2=x+x1: LET a=a-INT SQ
R (ABS (x1)^2+ABS (y1)^2)
1550 RETURN
1600 REM X and/or Y WRAP
1605 IF q>PI/2 THEN  GO TO 1630
1610 LET x3=x+(175-y)*TAN q
1615 IF x3>255 THEN  GO TO 1400
1620 IF x3<255 THEN  GO TO 1500
1625 GO TO 1680
1630 IF q>PI THEN  GO TO 1645
1635 LET x3=x+y*TAN (PI-q)
1640 GO TO 1615
1645 IF q>3*PI/2 THEN  GO TO 167
0
1650 LET x3=x-y*TAN (q-PI)
1655 IF x3<0 THEN  GO TO 1400
1660 IF x3>0 THEN  GO TO 1500
1665 GO TO 1680
1670 LET x3=x-(175-y)*TAN (2*PI-
q)
1675 GO TO 1655
1680 IF y1>=x1 THEN  LET a1=INT
(y1/COS q+1.5)
1685 IF y1<x1 THEN  LET a1=INT (
x1/SIN q+1.5)
1690 LET a=a-a1: GO SUB 1450: IF
 y2<0 THEN  LET y2=175
1695 IF y2>175 THEN  LET y2=0
1700 GO SUB 1500: IF x2<0 THEN
LET x2=255
1705 IF x2>255 THEN  LET x2=0
1710 RETURN
1990 REM
2000 REM   Input routine
2010 REM
2020 INPUT "W:"; LINE z$: LET s=
0
2025 LET count=0: LET rc=0: GO S
UB 2040: GO TO 2020
2030 REM
2040 REM   Command check
2050 REM
2060 LET t=0: LET t1=0: LET err=
0
2070 GO SUB 1020
2080 IF t1=1 AND y$="" AND count
=0 THEN  LET turt=0: GO SUB 1100
2085 IF t1=1 AND y$="" THEN  RET
URN
2090 IF LEN y$<>2 THEN  GO TO 21
30
2100 FOR i=1 TO m
2110 IF y$=x$(i) THEN  GO TO 220
0
2120 NEXT i
2130 LET y$=(y$+"              ")(
 TO 12)
2140 FOR i=1 TO n
2150 IF y$=w$(i) THEN  GO TO 221
0
2160 NEXT i
2170 REM  Command error
2180 LET err=1: IF turt=1 THEN
LET turt=0: GO SUB 1100
2190 GO TO 520
2200 GO SUB u(i): GO TO 2220
2210 GO SUB v(i)
2220 IF err>0 THEN  GO TO 520
2230 GO TO 2070
2990 REM
3000 REM   FORWARD routine
3010 REM
3020 GO SUB 1200: IF err>0 THEN
 RETURN
3030 IF turt=0 THEN  LET turt=1:
 GO SUB 1100
3040 LET q=dir*PI/180
3050 LET x1=INT (.5+a*SIN q): LE
T y1=INT (.5+a*COS q)
3060 LET tr=0: LET x2=x+x1: LET
y2=y+y1
3070 IF x2<0 OR x2>255 THEN  LET
 tr=1
3080 IF y2<0 OR y2>175 THEN  LET
 tr=tr+2
3090 IF tr>0 AND wr=0 THEN  LET
err=3: RETURN
3100 IF tr=0 THEN  GO TO 3120
3110 GO SUB (1300+tr*100)
3120 IF pp=1 THEN  PLOT x,y: DRA
W x1,y1
3130 IF pp=0 THEN  PLOT  INVERSE
 1; OVER 1;x,y: DRAW  INVERSE 1;
 OVER 1;x1,y1
3140 LET x=x2: LET y=y2: IF tr>0
 THEN  GO TO 3050
3150 LET t=0: RETURN
3190 REM
3200 REM  BACK
3210 REM
3220 IF turt=0 THEN  LET turt=1:
 GO SUB 1100
3230 LET dir=dir+180: IF dir>360
 THEN  LET dir=dir-360
3240 GO TO 3020
3390 REM
3400 REM  LEFT
3410 REM
```

```
3420 IF turt=0 THEN  LET turt=1:
 GO SUB 1100
3430 GO SUB 1200: IF err>0 THEN
 RETURN
3440 LET dir=dir-a
3450 IF dir<0 THEN  LET dir=360+
dir: GO TO 3450
3460 RETURN
3590 REM
3600 REM  RIGHT
3610 REM
3620 IF turt=0 THEN  LET turt=1:
 GO SUB 1100
3630 GO SUB 1200: IF err>0 THEN
 RETURN
3640 LET dir=dir+a
3650 IF dir>360 THEN  LET dir=di
r-360: GO TO 3650
3660 RETURN
3790 REM
3800 REM  DRAW
3810 REM
3820 CLS
3840 REM
3850 REM  HOME
3860 REM
3870 IF turt=0 THEN  LET turt=1:
 GO SUB 1100
3880 LET x=128: LET y=88
3890 LET dir=0: LET turt=0
3900 GO SUB 1100: RETURN
3990 REM
4000 REM  PENUP
4010 REM
4020 LET pp=0: RETURN
4030 REM
4050 REM  PENDOWN
4060 REM
4070 LET pp=1: RETURN
4080 REM
8000 REM  RANDOM
8010 GO SUB 1210: IF err=2 THEN
 RETURN
8020 LET a=INT (RND*a): RETURN
```

```
  65 DIM x$(m,2): DIM w$(n+40,12
): DIM f$(o,12)
 110 DATA 16,22,1
 125 DATA "RP",4100,"PC",4300,"W
R",4400,"NW",4600
 130 DATA "SX",4600,"SY",4630,"X
Y",4660,"SH",4700
 135 DATA "DF",5000,"ED",5200
 215 DATA "REPEAT",4100,"]",4200
,"PENCOLOUR",4300,"WRAP",4400,"N
OWRAP",4450
 220 DATA "SAVE",4500,"COPY",452
0,"STOP",4550
 225 DATA "SETX",4600,"SETY",463
0,"SETXY",4660,"SETHEADING",4700
 230 DATA "DEFINE",5000,"EDIT",5
200
 450 DIM r(20,2)
 465 LET def=0: DIM k$(40,200):
DIM l(40)
 470 DIM b$(10,28): DIM c(10): D
IM m(40): DIM n(40)
 730 LET a$="Too many REPEATs":
RETURN
 740 LET a$="DEFINE name error":
 RETURN
 750 LET a$="No room for further
 commands": RETURN
 760 LET a$="Incorrect Command n
ame": RETURN
2140 FOR i=1 TO n+def
2210 IF i<=n THEN  GO SUB v(i):
GO TO 2220
2215 GO SUB 6000
4100 REM   REPEAT
4105 REM
4110 GO SUB 1200: IF err>0 THEN
 RETURN
4115 LET s=s+1: IF s>LEN z$ THEN
  LET err=1: RETURN
4120 IF z$(s)=" " THEN  GO TO 41
15
4125 IF z$(s)="[" THEN  GO TO 41
35
4130 LET err=1: RETURN
4135 LET rc=rc+1: LET s=s+1
4140 IF rc>20 THEN  LET err=4: R
ETURN
4145 LET r(rc,1)=s: LET r(rc,2)=
a
4150 RETURN
4155 REM
4200 REM   ] - Repeat loop
4205 REM
4210 LET r(rc,2)=r(rc,2)-1
4215 IF r(rc,2)>0 THEN  LET s=r(
rc,1): RETURN
4220 LET rc=rc-1: RETURN
4300 REM   PENCOLOUR
4305 REM
4310 GO SUB 1200: IF err>0 THEN
 RETURN
4315 IF a>7 THEN  LET err=2: RET
URN
4320 INK a: RETURN
4325 REM
4400 REM   WRAP
4405 REM
```

```
4410 LET wr=1: RETURN
4415 REM
4450 REM   NOWRAP
4455 REM
4460 LET wr=0: RETURN
4465 REM
4500 REM   SAVE
4505 INPUT "SAVE - Enter file na
me ";n$
4510 SAVE n$ LINE 2000: RETURN
4515 REM
4520 REM   COPY
4525 COPY : RETURN
4530 REM
4550 REM   STOP
4555 CLS : STOP
4600 REM   SETX
4605 GO SUB 1200: IF err>0 THEN
 RETURN
4610 IF turt=0 THEN  LET turt=1:
 GO SUB 1100
4615 IF a<0 OR a>255 THEN  LET e
rr=2: RETURN
4620 LET y2=y: LET tr=0: LET y1=
0: LET x1=a-x: LET x2=a: GO TO 3
120
4625 REM
4630 REM   SETY
4635 GO SUB 1200: IF err>0 THEN
 RETURN
4640 IF turt=0 THEN  LET turt=1:
 GO SUB 1100
4645 IF a<0 OR a>175 THEN  LET e
rr=2: RETURN
4650 LET x2=x: LET tr=0: LET x1=
0: LET y1=a-y: LET y2=a: GO TO 3
120
4655 REM
4660 REM   SETXY
4665 IF turt=0 THEN  LET turt=1:
 GO SUB 1100
4670 GO SUB 1200: IF err>0 THEN
 RETURN
4675 IF a<0 OR a>255 THEN  LET e
rr=2: RETURN
4680 LET x1=a-x: LET x2=a: GO SU
B 1200: IF err>0 THEN  RETURN
4685 IF a<0 OR a>175 THEN  LET e
rr=2: RETURN
4690 LET tr=0: LET y1=a-y: LET y
2=a: GO TO 3120
4695 REM
4700 REM   SETHEADING
4705 GO SUB 1200: IF err>0 THEN
 RETURN
4710 IF a<0 OR a>359 THEN  LET e
rr=2: RETURN
4715 LET dir=a: RETURN
5000 REM   DEFINE
5005 GO SUB 1020: IF t1=1 AND LE
N y$<2 THEN  LET err=5: RETURN
5010 IF LEN y$<>2 THEN  GO TO 50
25
5015 FOR i=1 TO m: IF y$=x$(i) T
HEN  LET err=5: RETURN
5020 NEXT i
5025 LET y$=(y$+"              ")(
 TO 12)
5030 FOR i=1 TO def+n: IF y$=w$(
i) THEN  LET err=5: RETURN
5035 NEXT i: IF def>39 THEN  LET
 err=6: RETURN
5040 CLS : PRINT "DEFINE ";y$''
5050 LET def=def+1: LET no=0
5055 INPUT "W:"; LINE z$: IF LEN
 z$>28 THEN  PRINT #1;"Too Long!
": PAUSE 200: GO TO 5055
5060 IF LEN z$<2 THEN  PRINT #1;
"Nonsense!": PAUSE 200: GO TO 50
55
5065 PRINT no;TAB 3;z$'': LET no
=no+1: LET len=LEN z$
5070 IF len<3 THEN  GO TO 5085
5075 FOR i=1 TO len-2: IF z$(i T
O i+2)="END" THEN  GO TO 5095
5080 NEXT i
5085 IF no<11 THEN  LET b$(no)=z
$: LET c(no)=len: GO TO 5055
5090 PRINT #1;"No more space is
available for  ";y$: PAUSE 200:
GO TO 5100
5095 LET len=len-3: LET b$(no)=z
$( TO len): LET c(no)=len: IF no
<10 THEN  LET c(no+1)=0
5100 LET w$(n+def)=y$: LET c$=""
: LET no=1: LET len=0
5105 IF c(no)=0 THEN  GO TO 5120
5110 LET c$=c$+b$(no, TO c(no))+
CHR$ 0: LET len=len+c(no)+1
5115 LET no=no+1: IF no<10 THEN
 GO TO 5105
5120 LET k$(def)=c$: LET l(def)=
len
5125 CLS : PRINT "STORED - ";y$
5130 PAUSE 200: CLS : RETURN
5200 REM   EDIT
5205 GO SUB 1020: IF t1=1 AND LE
N y$<2 THEN  LET err=6: RETURN
5210 LET y$=(y$+"              ")(
 TO 12)
5215 IF def=0 THEN  LET err=4: R
ETURN
5220 FOR i=n+1 TO n+def: IF y$=w
$(i) THEN  GO TO 5230
5225 NEXT i: LET err=4: RETURN
5230 GO SUB 5800
5240 GO SUB 5950
5250 PRINT #1;"1:EDIT    2:INSER
```

```
T    3:DELETE  4:REMOVE  5:RETUR
N"
5255 LET z$=INKEY$: IF z$="" THE
N  GO TO 5255
5260 IF z$<"1" OR z$>"5" THEN  G
O TO 5255
5265 LET z=VAL z$: GO TO 5200+10
0*z
5275 GO TO 5235
5300 GO SUB 5950: PRINT #1;"EDIT
 - Enter the line"
5305 LET z$=INKEY$: IF z$="" THE
N  GO TO 5305
5310 IF z$<"0" OR z$>"9" THEN  G
O TO 5305
5315 LET lin=VAL z$
5320 INPUT (lin);" W:"; LINE z$:
 IF LEN z$>28 THEN  PRINT #1;"To
o Long": PAUSE 200: GO TO 5320
5325 LET b$(lin+1)=z$: LET c(lin
+1)=LEN z$
5330 GO TO 5235
5400 IF k>9 THEN  PRINT #1;"No s
pace for another line": PAUSE 20
0: RETURN
5405 GO SUB 5950: PRINT #1;"INSE
RT - Enter line number"
5410 LET z$=INKEY$: IF z$="" THE
N  GO TO 5410
5415 IF z$<"0" AND z$>STR$ (k-1)
 THEN  GO TO 5410
5420 LET ins=VAL z$
5425 INPUT (ins);" W:"; LINE z$:
 IF LEN z$>28 THEN  PRINT #1;"To
o Long": PAUSE 200: GO TO 5425
5430 LET ins=ins+1
5435 FOR i=k TO ins STEP -1
5440 LET b$(i+1)=b$(i): LET c(i+
1)=c(i): NEXT i
5445 LET b$(ins)=z$: LET c(ins)=
LEN z$
5450 LET k=k+1: GO TO 5235
5500 LET z$="DELETE ": GO SUB 59
50: GO SUB 5900
5505 IF a$="N" THEN  GO TO 5235
5510 GO SUB 5950: PRINT #1;z$;"-
 Enter the line number"
5515 LET z$=INKEY$: IF z$="" THE
N  GO TO 5515
5520 IF z$<"0" OR z$>STR$ (k-1)
THEN  GO TO 5915
5525 LET k=k-1
5530 FOR i=1+VAL z$ TO k
5535 LET b$(i)=b$(i+1): LET c(i)
=c(i+1): NEXT i
5540 GO TO 5235
5600 LET z$="REMOVE ": GO SUB 59
50: GO SUB 5900
5605 IF a$="N" THEN  GO TO 5235
5610 LET def=def-1: IF def=0 OR
ed=def+1 THEN  RETURN
5620 FOR i=n+ed TO def+n
5625 LET w$(i)=w$(i+1): NEXT i
5630 FOR i=ed TO def
5635 LET k$(i)=k$(i+1): LET l(i)
=l(i+1): NEXT i
5640 CLS : RETURN
5700 LET c$="": LET no=0
5705 FOR i=1 TO k
5710 LET c$=c$+b$(i, TO c(i))+CH
R$ 0: LET no=no+c(i)+1
5715 NEXT i
5720 LET k$(ed)=c$: LET l(ed)=no
5725 CLS : PRINT "STORED - ";y$:
 PAUSE 200
5730 CLS : RETURN
5800 LET ed=i-n: CLS : PRINT "ED
IT ";y$''"Please wait"
5805 LET len=1: LET k=0: LET z$=
k$(ed, TO l(ed))
5810 IF len>l(ed) THEN  RETURN
5815 LET c$="": LET no=0: LET k=
k+1
5820 IF z$(len)<>CHR$ 0 THEN  LE
T c$=c$+z$(len): LET no=no+1: LE
T len=len+1: GO TO 5820
5825 LET len=len+1: LET b$(k)=c$
: LET c(k)=no: GO TO 5810
5830 RETURN
5900 PRINT #1;z$;"- Are you sure
? (y/n)"
5905 LET a$=INKEY$: IF a$="" THE
N  GO TO 5905
5910 IF NOT (a$="Y" OR a$="N") T
HEN  GO TO 5905
5915 RETURN
5950 CLS : PRINT "EDIT ";y$''
5955 FOR i=1 TO k
5960 PRINT i-1;TAB 3;b$(i)'': NE
XT i
5965 RETURN
6000 REM  Use a defined command
6005 IF count=0 THEN  LET p$=z$:
 LET s1=s
6010 IF count>0 THEN  LET m(coun
t)=comm: LET n(count)=s
6015 LET comm=i-n: LET z$=k$(com
m, TO l(comm))
6020 LET count=count+1
6025 LET s=0: GO SUB 2040
6030 LET count=count-1
6035 IF count>0 THEN  LET comm=m
(count): LET z$=k$(comm, TO l(co
mm)): LET s=n(count)
6040 IF count=0 THEN  LET z$=p$:
 LET s=s1
6045 IF count<0 THEN  LET err=1
6050 RETURN
```

GAVIN SMYTH

Galactoids

A fast moving machine code game — can you blast the random coloured spider-like aliens before it is too late?

At last! A program that overcomes the problem of the speed (or lack of it) of Sinclair BASIC. This machine code space invaders-type game occupies just over 1K and so will run on either the 16K or 48K Spectrum. This is not an attempt to teach machine code programming, but with the instructions given, anyone can run the program without having to understand it.

The object of the game is to destroy the eight coloured spiderlike aliens randomly moving about the screen dropping bombs all the time. Naturally you have to hit them dead centre, but if a bomb touches your gun at all, bang! If you are invaded, you only lose a life, not the whole game. To begin with, the game is easy since the gun moves at eight times the speed of an invader but after you have bumped off the first three without trying, the remaining aliens move at a fifth of the guns speed — to get the last one, you have either to be pretty quick or very lucky! After this, you get another sheet with an extra life, but games do not normally last for more than three screenfuls of aliens.

THE AIM OF THE PROGAM

There are several reasons for writing this program:

1 To make a game less predictable than the familiar legions marching across the screen in neat rows. This randomness adds to the difficulty, keeping games short and defeating those show-offs who hog the space invaders machine all evening.

2 To keep kids (and adults) out of the way by making the games colourful, noisy, simple to operate and totally uncrashable (famous last words). The controls are: Z for left, X for right and space to fire. There is no return to BASIC in the program (to prevent cries of "Hey! They've stopped moving!"), so the only way to dispel the menace forever is to pull the plug out — but beware, the game is addictive.

3 To cause endless frustration with the aliens' continuous chirping and high scores not even in three figures. (My best so far is a mere quarter century and the score counter can clock up to 9999.)

4 Finally, to show how the Spectrum ROM can be used to simplify programming be performing the more fundamental duties such as printing.

INTERESTING ROUTINES

Here are a few of the techniques used in the program to get a lot out of the machine without typing too much in:

RANDOM NUMBERS: The subroutine RND (address 32056d) returns a fairly random number in the A register. What it does is step a pointer through the ROM and return the value of that location (random enough for this purpose although there is a distinctly non-random region) an unused region of over 1K starting at 14446d, where all the memory locations contain 255d. This provides an interesting short period every three sheets or so when the aliens move uniformly without dropping bombs and with a distinct sound, but they are still just as frustratingly tricky to hit.

SOUND: The subroutine BLEEP (31677d) is used to emit a random short tone. It is there simply to slow the program a little, it can be slowed still more by increasing the value of DE in this routine (equivalent to POKEing a larger number of locations 31683d). This routine calls the ROM subroutine BEEPER (969d) which controls the loudspeaker. This is one of the few routines that corrupts the IX register pair, hence the PUSH IX and POP IX near it in the program in places.

To use BEEPER, DE should contain a number equal to INT(frequency*duration) and HL should contain INT(437500d/frequency- 30.125d). For example, to play the note A 440Hz for 2.5 seconds:

DE contains INT(440*2.5)=1100d
HL contains INT(437500/440-30.125)=962d

INVADER CONTROL: The IX register pair is used to hold the start of the individual invader database, greatly simplifying code since, for example, the invader column is always (IX+1) regardless of the invader being used.

GRAPHICS GALORE

The remainder of the routines deal with graphics:

CLS (3435d): This routine simply clears the screen.

BORD1 (8859d): This sets the border to the colour contained in the A register. For example, to set a red (code 2) border:

```
LD A,2
CALL BORD1
```

CHANOPEN (5633d): Before writing to the screen, it must be opened for printing. The upper screen is channel "S" with code 2, so to prepare for printing:

```
LD A,2
CALL CHANOPEN
```

PRSTRING (8252d): This prints a string of characters of length BC which starts at the address contained in DE. For example, to print the word "Sinclair" which is held in the ROM at 5440d:

```
LE DE,5440d
LD BC,8; length of word
CALL PRSTRING
```

OUTNUM1 (6683d): This routine prints out the integer contained in BC, it is used to print line numbers. The number must be positive and less than 10000.

(No check has been made for overflow of this type since it is unlikely that such a score will be reached.)

CLSET (3545d): There are two ways to move the print position. One is to use the AT code (22d) followed by position with PRSTRING or RST 10. This method is used in the PRINT routine, address 32034d. The other method is to use CLSET: for this, B contains 24d-row; C contains 33d-column. For example:

```
PRINT AT 5,6;is equivalent to:
LD B,19d ;(24-5)
LD C,27d ;(33-6)
CALL CLSET
```

You will notice that RST 10 occurs frequently in the program — this calls the main printing routine which displays the character with code held in A. It will cope with any character, including tokens, paper and ink controls, and position controls. For example, print blue "X":

```
LDA A,16d;INK control code
RST 10
LD, A,1 ;colour code for blue
RST 10
LA,88d ;code for "X"
RST 10
```

The subroutine MAININIT (31206d) takes care of setting up some system pointers:

UDG: Rather than move all the data for the graphics characters into the user defined area, the program changes the system variable UDG (23675d) to point to the graphics data at the end of the program. (If Mohammed can't come to the mountain. . .)

ATTRP: Loading location 23693d with 7 is equivalent to PAPER 0: INK 7. In general, this system variable contains:

FLASH*128d + BRIGHT*64d + PAPER*8 + INK

MASKP: Loading 23694d with 255d is equivalent to INK 8: PAPER 8: FLASH 8:

BRIGHT 8 — setting ink and paper to transparent so that screen colours will not be affected by anything printed — the alternative to this would be complex colour controls before each print.

There is also a general listing of the whole program (see listing 4).

Most of the information on ROM routines was found in Dr Ian Logan's book, *'Understanding your Spectrum'*, which contains a lot of valuable information on other routines and machine code for the Z80.

TYPING IT IN

We received the listing of this program in full HEX, Mnemonics and REM form-unfortunately this ran to 16 pages! The present form was obtained by a program which created the lines in listing 1 by PEEKing at the code from the tape of the program supplied.

Each line of the listing consists of ten DATA bytes plus a checksum to help ensure that errors are elminated. The method of entering these codes are slightly different for the 16 and 48K Spectrums:

48K: Enter both lisitings 1 and 2a and RUN the program, correcting any errors that are reported. Should you get an 'integer out of range' report, type PRINT i-50990 and press Enter.

PROGRAM STRUCTURE

START	*Call MAININIT*	*:Set system variables, 'random number pointer.*
NEWG	*Call NEWGAME*	*:Reset score(0), lives(4); print instructions; wait until ENTER is pressed.*
SCREEN	*Call SETUP*	*:Print score, hiscore, lives; set screen attributes (in bands of different inks).*
MAINLOOP	*For each invader — Test lives; if none, GOTO ENDGAME. If invader pointed to is dead, try next; GOTO MOVE8.*	
Call UPDATE	*:Call MOVINVDR*	*:1 in 16 chance of row change (¼ up, ¾ move down). Check limits; if on edge or top, restore old position. If on bottom, return from MOVINVDR. If on edge, reverse direction. Random 1 in 16 chance of reverse. If bomb not dropped, ½ chance of drop; copy invader position into bomb.*
	Call INVDRPRNT	*Erase old invader, print new one.*
	If bomb dropped, Call MOVBOMB	*:Erase old bomb; move down a row; print new bomb.*
	Call KEYBOARD	*:Call BLEEP. If SPC pressed and bullet not fired, copy gun position into bullet. If Z pressed, move left. If X pressed, move right. If new position out of range, restore old co-ordinates.*
	Call GUNPRINT	*Erase old gun, print new one.*
	If bullet fired, Call MOVBULLET	*:Erase old bullet; move up a row; If at top row, reset bullet else print new bullet.*
For each invader		
CHECK	*If alive, Call LASTROW*	*:If invader on bottom line, blow up gun and invader.*
.Call HITINVDR	*:If bullet and invader co-incident, Call BLOWNINVDR*	*:Explosion (flashing, beeping), Reset bullet; signal dead invader; decrease number :invaders; increase score; update hiscore if necessary; print score and hiscore.*
Call HITGUN	*:If bomb has hit gun, Call BLOWGUN*	*:Explosion; reset bullet; decrease lives; print lives.*
	If bomb is on bottom line, Call ERASEBOMB	*:Erase bomb; reset its co-ordinates.*

Loop back to CHECK for next invader.
Loop back to MOVE8 until all 8 invaders updated.
Go back to MAINLOOP to repeat sequence.

ENDSCREEN	*Award extra life; beep low tone; start another screen by jumping to SCREEN.*
ENDGAME	*:Call DISPLAY print end message; flash screen and beep; beep very low tone.*

Start another game by jumping to NEWG.

Correct any errors then NEW the program and enter:

```
10 CLEAR 30999: FOR i=31000 TO
32417: POKE i, PEEK
(i+20000):NEXT i
```

Now RUN it and enter Listing 3.
16K: Do not enter listing 1 but enter listing 2b and RUN the program, enter each of the 11 numbers in each DATA line one number at a time. If you make a mistake the program will tell you and you will need to re-enter that line.

SAVING

Before testing the program it MUST be saved on tape. Even though we have taken great pains to foolproof the entering system, it is still possible for an error to slip through!

First, remove the BASIC listing (it you entered it in the 48K mode make a copy on tape just in case by using NEW — the machine code is quite safe) then enter listing 3. To save both the loader and m/c enter GOTO 40, start the tape and press any key. After a few seconds the "start tape" message appears again, press another key and the m/c will be saved. Verify both sections by VERIFY"" and then VERIFY""Code.

To start the game either reload it from tape or enter RANDOMISE USR 31000. If all is well the instructions will be displayed and look out!

If it crashes then switch off, enter listing 4, RUN, reload the program and check each number until you find the mistake (not forgetting to ignore the last number of each DATA line), POKE the address given with the correct number. Now follow the saving procedure again.

```
  10 DATA 205,230,121,205,7,122
,205,64,122,33,1314
  20 DATA 81,125,6,8,58,78,125,
167,40,82,770
  30 DATA 58,75,125,167,40,61,9
4,35,86,35,776
  40 DATA 229,197,213,221,225,2
21,126,1,167,40,1640
  50 DATA 40,205,203,121,221,22
9,6,8,33,81,1147
  60 DATA 125,94,35,86,35,229,1
97,213,221,225,1460
  70 DATA 221,126,1,167,40,9,20
5,131,121,205,1226
  80 DATA 151,121,205,168,121,1
93,225,16,228,221,1649
  90 DATA 225,193,225,16,185,24
,178,33,78,125,1282
 100 DATA 52,17,64,0,33,0,32,20
5,181,3,587
 110 DATA 24,160,205,196,122,24
,152,221,126,0,1230
 120 DATA 254,21,192,205,27,124
,42,73,125,43,1106
 130 DATA 34,73,125,205,208,123
,201,221,126,0,1316
 140 DATA 33,79,125,190,192,221
,126,1,35,190,1192
 150 DATA 204,208,123,201,221,1
26,5,254,21,192,1555
 160 DATA 221,70,6,58,76,125,20
5,187,121,205,1274
 170 DATA 186,124,201,61,184,20
4,27,124,60,184,1355
 180 DATA 204,27,124,60,184,204
,27,124,201,205,1360
 190 DATA 1,123,205,249,124,221
,126,6,167,196,1418
 200 DATA 118,124,205,129,123,2
05,228,124,58,80,1394
 210 DATA 125,167,196,150,124,2
01,33,74,126,34,1230
 220 DATA 123,92,62,7,33,141,92
,119,62,255,986
 230 DATA 35,119,175,205,155,34
,58,120,92,111,1104
 240 DATA 237,95,230,31,103,34,
118,92,201,175,1316
 250 DATA 50,73,125,62,4,50,78,
125,205,107,879
 260 DATA 13,62,2,205,1,22,17,1
53,125,1,601
 270 DATA 32,0,205,60,32,1,5,24
,205,217,781
 280 DATA 13,237,75,71,125,205,
27,26,17,197,993
 290 DATA 125,1,133,0,205,60,32
,62,191,219,1028
 300 DATA 254,230,1,32,248,201,
205,107,13,62,1353
 310 DATA 2,205,1,22,17,153,125
,1,32,0,558
 320 DATA 205,60,32,1,5,24,205,
217,13,237,999
 330 DATA 75,71,125,205,27,26,1
,26,24,205,785
 340 DATA 217,13,237,75,73,125,
205,27,26,1,999
 350 DATA 15,24,205,217,13,237,
75,78,125,6,995
 360 DATA 0,205,27,26,33,0,88,6
2,7,6,454
 370 DATA 128,119,35,16,252,61,
254,2,32,245,1144
 380 DATA 62,8,50,75,125,33,81,
125,6,8,573
 390 DATA 94,35,86,35,213,221,2
25,175,221,119,1424
 400 DATA 6,221,112,0,221,52,0,
205,56,125,998
 410 DATA 230,2,61,221,119,4,20
5,56,125,230,1253
 420 DATA 15,198,8,221,119,1,16
,218,62,7,865
 430 DATA 50,76,125,175,50,80,1
25,201,17,185,1084
 440 DATA 125,1,12,0,205,60,32,
33,32,88,588
 450 DATA 17,33,88,6,82,229,213
,197,205,56,1126
 460 DATA 125,1,127,2,119,237,1
76,241,245,47,1320
```

```
  470 DATA 230,7,60,103,205,56,1
25,111,17,16,930
  480 DATA 0,205,181,3,193,209,2
25,16,222,33,1287
  490 DATA 0,48,17,80,0,205,181,
3,201,221,956
  500 DATA 126,0,221,119,2,221,1
26,1,221,119,1156
  510 DATA 3,205,56,125,230,15,3
2,15,205,56,942
  520 DATA 125,230,3,32,5,221,53
,0,24,3,696
  530 DATA 221,52,0,221,126,4,22
1,134,1,221,1201
  540 DATA 119,1,254,1,40,15,254
,30,40,11,765
  550 DATA 221,126,0,167,40,5,25
4,21,200,24,1058
  560 DATA 12,221,126,2,221,119,
0,221,126,3,1051
  570 DATA 221,119,1,221,126,1,2
54,2,204,120,1269
  580 DATA 123,254,29,204,120,12
3,205,56,125,230,1469
  590 DATA 15,204,120,123,221,12
6,6,167,192,205,1379
  600 DATA 56,125,230,1,192,221,
126,0,221,119,1291
  610 DATA 5,221,126,1,221,119,6
,201,221,126,1247
  620 DATA 4,237,68,221,119,4,20
1,205,189,123,1371
  630 DATA 33,76,125,62,127,219,
254,230,1,32,1159
  640 DATA 15,58,80,125,167,32,9
,126,50,80,742
  650 DATA 125,62,21,50,79,125,1
26,50,77,125,840
  660 DATA 62,254,219,254,203,87
,32,1,52,230,1394
  670 DATA 2,32,1,53,126,254,4,4
0,3,254,769
  680 DATA 27,192,58,77,125,119,
201,205,56,125,1185
  690 DATA 246,15,17,16,0,38,1,1
11,221,229,894
  700 DATA 205,181,3,221,225,201
,33,75,125,53,1322
  710 DATA 62,147,221,70,0,221,7
8,1,205,73,1078
  720 DATA 124,175,221,119,1,50,
80,125,42,73,1010
  730 DATA 125,35,34,73,125,58,7
2,125,188,40,875
  740 DATA 7,56,11,34,71,125,24,
6,58,71,463
  750 DATA 125,189,56,245,1,26,2
4,205,217,13,1101
  760 DATA 237,75,73,125,205,27,
26,1,5,24,798
  770 DATA 205,217,13,237,75,71,
125,205,27,26,1201
  780 DATA 201,58,76,125,6,21,79
,62,144,205,977
  790 DATA 73,124,1,15,24,205,21
7,13,33,78,783
  800 DATA 125,53,6,0,78,205,27,
26,62,22,604
  810 DATA 215,58,79,125,215,58,
80,125,215,62,1232
  820 DATA 32,215,175,50,80,125,
201,221,229,245,1573
  830 DATA 197,6,21,120,50,77,12
5,193,197,62,1048
  840 DATA 150,205,205,124,193,2
41,245,197,205,205,1970
  850 DATA 124,58,77,125,71,16,2
32,193,205,14,1115
  860 DATA 125,241,221,225,175,2
05,155,34,205,186,1772
  870 DATA 124,201,62,22,215,221
,126,5,215,221,1412
  880 DATA 126,6,215,62,32,215,2
21,52,5,62,996
  890 DATA 22,215,221,126,5,215,
221,126,6,215,1372
  900 DATA 62,154,215,201,33,79,
125,62,22,215,1168
  910 DATA 126,215,58,80,125,215
,62,32,215,53,1181
  920 DATA 40,13,62,22,215,126,2
15,58,80,125,956
  930 DATA 215,62,153,215,201,17
5,50,80,125,201,1477
  940 DATA 62,22,215,221,126,5,2
15,221,126,6,1219
  950 DATA 215,62,32,215,175,221
,119,6,201,205,1451
  960 DATA 34,125,38,2,22,0,205,
56,125,230,837
  970 DATA 31,95,205,181,3,205,5
6,125,205,155,1261
  980 DATA 34,201,58,77,125,6,21
,79,205,14,820
  990 DATA 125,58,76,125,6,21,79
,62,144,205,901
 1000 DATA 34,125,201,221,70,2,2
21,78,3,205,1160
 1010 DATA 14,125,221,70,0,221,7
8,1,62,147,939
 1020 DATA 205,34,125,201,62,22,
215,120,215,121,1320
 1030 DATA 215,62,8,215,6,4,62,3
2,245,215,1064
 1040 DATA 241,16,251,201,245,62
,22,215,120,215,1588
 1050 DATA 121,215,62,8,215,241,
245,215,241,60,1623
 1060 DATA 245,215,241,60,215,20
```

GAME

```
1,229,42,118,92,1658
 1070 DATA 35,62,63,164,103,34,1
18,92,126,225,1022
 1080 DATA 201,0,0,0,0,8,0,0,3,0
,212
 1090 DATA 0,97,125,104,125,111,
125,118,125,125,1055
 1100 DATA 125,132,125,139,125,1
46,125,0,0,0,917
 1110 DATA 0,0,0,0,0,0,0,0,0,0,0
 1120 DATA 0,0,0,0,0,0,0,0,0,0,0
 1130 DATA 0,0,0,0,0,0,0,0,0,0,0
 1140 DATA 0,0,0,0,0,0,0,0,0,0,0
 1150 DATA 0,0,0,0,0,0,0,0,0,0,0
 1160 DATA 0,0,0,22,0,0,32,83,67
,79,283
 1170 DATA 82,69,32,48,32,32,32,
32,76,105,540
 1180 DATA 118,101,115,32,52,32,
72,73,83,67,745
 1190 DATA 79,82,69,32,48,22,11,
11,71,65,490
 1200 DATA 77,69,32,79,86,69,82,
22,4,11,531
 1210 DATA 73,78,86,65,68,69,82,
83,13,13,630
 1220 DATA 13,32,32,32,32,32,60,
90,62,32,417
 1230 DATA 109,111,118,101,115,3
2,108,101,102,116,1013
 1240 DATA 13,32,32,32,32,32,60,
88,62,32,415
 1250 DATA 109,111,118,101,115,3
2,114,105,103,104,1012
 1260 DATA 116,13,32,32,32,32,60
,83,80,67,547
 1270 DATA 62,32,102,105,114,101
,115,13,13,32,689
 1280 DATA 32,80,114,101,115,115
,32,60,69,78,796
 1290 DATA 84,69,82,62,32,116,11
1,32,98,101,787
 1300 DATA 103,105,110,13,13,13,
13,13,147,148,678
 1310 DATA 149,32,32,153,32,154,
32,153,32,32,801
 1320 DATA 150,151,152,32,32,150
,151,152,32,32,1034
 1330 DATA 153,32,154,32,153,32,
32,147,148,149,1032
 1340 DATA 0,0,1,0,4,31,51,254,6
0,255,656
 1350 DATA 90,36,126,255,189,24,
0,0,128,0,848
 1360 DATA 32,248,204,127,8,20,3
4,65,1,102,841
 1370 DATA 24,0,0,126,219,153,25
5,195,126,60,1158
 1380 DATA 24,36,67,128,130,76,4
8,0,224,30,763
 1390 DATA 0,31,224,7,56,192,129
,66,36,129,870
 1400 DATA 129,36,66,129,3,28,22
4,7,248,0,870
 1410 DATA 120,7,0,24,60,60,60,2
4,60,0,415
 1420 DATA 102,60,24,24,60,126,6
0,0,0,0,456
```

LISTING 2a

```
1999 REM ** 48K LOADER **
2000 CLEAR 30999: RESTORE
2010 FOR i=31000 TO 32417 STEP 1
0: LET c=0
2020 FOR j=0 TO 9: READ a: POKE
(i+j),a: LET c=c+a: NEXT j
2030 READ a: IF a <> c THEN  CLS
 : PRINT  FLASH 1;"ERROR AT LINE
 ";i-30990: FLASH 0: STOP
2040 NEXT i
```

LISTING 2b

```
1999 REM ** 16K LOADER **
2000 CLEAR 30999: RESTORE
2010 FOR i=31000 TO 32417 STEP 1
0
2020 LET c=0: FOR j=0 TO 9: INPU
T a: POKE (i+j),a: LET c=c+a: NE
XT j
2030 INPUT a: IF a <> c THEN  CL
S : PRINT  FLASH 1;"ERROR AT LIN
E ";i-30990: FLASH 0: GO TO 2020
2040 NEXT i
```

LISTING 3

```
  9 REM ** INITIAL PROGRAM **
 10 CLEAR 30999
 20 LOAD "" CODE
 30 RANDOMIZE  USR 31000
 40 SAVE "galacts" LINE 10
 50 SAVE "mc" CODE 31000,1418
```

LISTING 4

```
  1 REM ** CHECKING PROGRAM **
  5 CLEAR 30999: LOAD "" CODE
 10 FOR i=31000 TO 32417
 20 PRINT "Address-";i;"="; PEE
K i
 30 NEXT i
```

Datafile

A multi-purpose file handling program with, er, multi-purpose file handling applications.

Keeping records can be a laborious job — bits of paper, alterations, amendments etc etc. But now in this comprehensive and versatile program you have all the facilities that most filing applications will require.

Datafile impressed us in both operation and in programming techniques, it is liberally REMmed so that the operation of each section is explained and has been well documented by Nigel for the operator to use.

Whatever your interest, job, or need, Datafile will be of use. Type it in and then read the following instructions on using it.

RECORD LIMIT

Datafile is a utility program which sets up a file in A$ in accordance with parameters supplied by the user. The only limit to the number of records held is the size of memory of your Spectrum. A 16K machine can only manage 28 records 100 characters long but fields within each record can be any length and each field can be given a name up to 6 characters long.

TOOLKIT

Toolkit is a set of subroutines designed to enable you to use files generated by Datafile in your own programs. The sample application program Letters gives an example of one such application.

The uses of these routines are manifold. You might use them to set up a directory of your records and tapes or books. You could equally use Datafile to keep track of all your business clients. The example which follows guides you through every stage of datafile and explains the use of Toolkit routines.

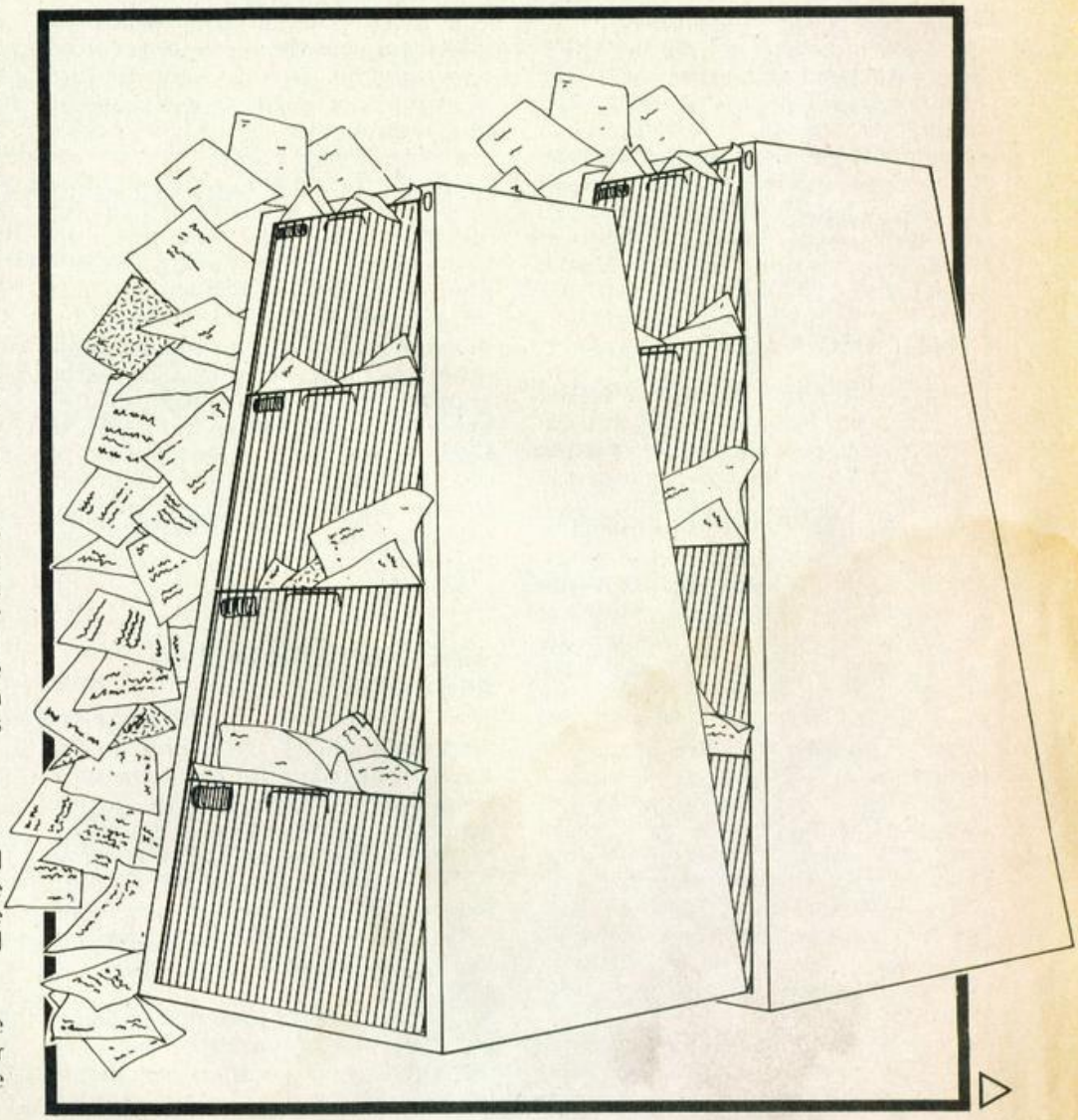
USING THE PROGRAM

On using the LOAD "" command and playing your tape you will be greeted with the main menu. Before you can use a file it must be defined in the following ways:

1. Select option from the menu. The program first asks you to name your file. You can call it anything you like but for this example enter CLIENT.
2. Number of records. Once you have named your file the program asks you how many records will be on this file. It is always a good idea to specify more than you think will be required to allow for future growth. For this example enter 10. This will mean that only 9 clients can appear on the file because Datafile uses the first record to store information about the file size and structure.
3. Record size. Each record can be any size you wish. Obviously it must be equal to or greater than the maximum number of characters of data what you wish to store on each record. For the example enter 120.
4. Number of fields. What you enter here requires some thought. It is necessary to define one field for every item of data which you want to hold on each client. The difficult part may be deciding what constitutes an "item of data".

Say you wanted to store the address of each client. You could specify just one field say, ADDRESS. This would not be very useful because then you would be unable to access individual lines of the address. A better solution would be to define one field for each line of the address and call them something like ADLIN1, ADLIN2 etc to refer to address line 1 and so on. For this example enter 10.

5. Field definition. If you have followed the example instructions then the program will now tell you that the CLIENT file has 10 records of 120 characters each.



MEAN IT

The program will then ask for a name and size for each field in turn. You can call the fields anything you like but any name more then 6 characters long will cut short automatically. I recommend that you give meaningful names to your fields to make future use of your files easier. There is no restriction on the length which you specify for each field providing that the sum of the field lengths is less than the record length which you entered at 3 (record size) above. For the example I suggest that you enter the following field names and lengths:

Name of field	Length of field
INITS	*6*
NAME	*15*
ADLIN1	*20*
ADLIN2	*15*
ADLIN3	*15*
ADLIN4	*10*
PHONE	*15*
ACCNO	*6*
INVDAT	*8*
INVNO	*6*
	116

The total length of these fields is 116 characters which is less than the 120 character size which we entered at 3 above, When you have entered the data above you will have defined all of the 10 fields which you specified for each client record at 4 above, You will now be returned to the main menu.

ENTERING DATA

First select option 2 from the menu. If you have been following the example then you will be asked for the value of INITS in this record. For reasons which will become plain when you use LETTERS, enter your own initials separated by full stops. Your entry will now be displayed next to the field name. If you made a mistake don't worry, you will be given an opportunity to put it right when you have entered all the data for this record. Continue to enter your details as prompted. If you just press "ENTER" when asked for a value then that field will not be left empty. Clearly you will not have an account number and you will not have invoices outstanding yourself. Thus for these fields leave a blank. If you later sort the file by invoice date for instance then records with no invoice data will be put at the front of the file.

Having entered your own details you will be asked whether you wish to alter anything. If you enter "Y" then you will be prompted to enter a new value for each field in turn. If you just press "ENTER" when prompted then the value for that field will be left unchanged. When you are happy with the record content, press "ENTER" when asked if you want to alter anything and you will be asked whether you wish to enter any more records. If you enter "Y" then the process above will begin again. If you just press "ENTER" then you will be returned to the main menu.

When you enter dates I suggest that you use the American format: YY/MM/DD rather than the British format: DD/MM/YY. If you do this then you will be able to sort your records by date using the sort routine in Datafile. You might wish to sort clients into Date of Birth order for example.

SORTING FILES

Select option "3" from that main menu. Assuming that you have more than one record on your file you can now sort them alphabetically or numerically.

The sort routine will first ask you for the name of the files by which you want to sort your records. For example, if you want to sort your file so that the records appear in alphabetical order of names then you might enter NAME.

If the field name that you have entered was a valid one then the message "SORTING" will appear on your screen. When sorting is complete you will be returned to the main menu. If you have entered the name of a field which does not exist then you will be asked to enter another. If you have forgotten the names you assigned to the fields of your record don't panic. Just press "ENTER" and you will be returned to the main menu. You can now select option "8" which will tell you the names that you gave (see print file detail).

AMENDING/DELETING RECORDS

Select option "4" from the menu. You will be asked for the name of a field. Say you know that the record which you wish to amend is that of Mr West. This means that the name WEST will be in the NAME field of that record. You should thus enter NAME when asked for a field and WEST when asked for a value for that field.

If you enter a field name which you have not defined then you will be asked to try another. Just pressing "ENTER" at this point will return you to the menu. If the value of the field does not exist on the file then a message will be displayed to this effect. For example, if you had not entered a record for Mr West when entering data.

Once the record that you require has been found you will be given the opportunity to alter each field in turn. If you do not wish to amend a particular field then just press "ENTER" when prompted for a new value.

To delete a record select option "5" from the menu. As when amending records, you will be asked for the information that the program requires to find the record that you wish to delete. When the record has been found it will be removed from the file and you will be returned to the main menu.

SAVING AND LOADING

Once you have got some data on your file you may want to save the file to tape. To do this select option "6" from the menu. The file will now be saved under the name by which it was defined above. If you followed the example your file will be saved as "CLIENT" DATA A$ ( ).

When the file has been saved you will be prompted to rewind the tape and prepare for verification. You will be returned to the main menu when verification is completed. If verification fails, use "GO TO 1" to return to the menu and then try again.

If you wish to update files which you have already saved to tape then select option "7" from the menu. You will be asked for the name of the file. If you have forgotten it then just press "ENTER". The program will then load the first file that it comes to on tape.

When the program has loaded your file it will decode the first record to determine the format of your file and the names which you gave to its field. It will then return to the menu. You can find out the names of the fields etc by selecting option "8".

PRINT FILE DETAIL

Select option "8" from the menu. The program will then print the names of each of the fields in the file and tell you the position that they occupy in each record.

For example if field INITS starts at 1 and ends at 6 then this tells you that INITS occupies the first 6 characters of each record, ie for record 2 inits would be A$, (2,1 TO 6).

To print the whole file select option "9" from the menu. Each record

on the file will be displayed in turn.

WARNING!

If you BREAK out of the program for any reason use "GO TO 1" to return to the menu. Do not use "RUN" since this will result in the loss of any data held on the current file.

A list if the variables used by Datafile is given in REM statements at the start of the program. You *must not* use these variables in your own programs when you use Toolkit to manipulate data held on files created by Datafile.

USING TOOLKIT

Each of the routines in TOOLKIT is designed to make using Datafile files in your program easy. Each routine is fully documented in REM statements.

1. Get field value. Each record in a Datafile file is stored as an element of the array A$. For example, the first record in the file will be A$(2). You will remember from the field definition stage that each field within a record also has a number and a name. The field number enables Toolkit to find out the position of a field in each record. It might, for instance, be A$(2,1 TO 6) as for INITS in the example. This routine when supplied with a record number in RN and a field number in FN will fetch the value of that field in record number RN. The value is returned in Q$ and trailing spaces are automatically removed.
2. Decoding dates. I have already explained the usefulness of storing dates in the American format. This routine will translate a date supplied in Q$ from the American form to the British form.
3. Removing spaces. If you have given a field in a record a value with less characters than the field length then the field will contain spaces after your value. This routine removes the unwanted spaces from a field value supplied in Q$ and returns the trimmed value in Q$.
4. Read header. I explained above that the first record on a Datafile contains information required by Toolkit to use the data in your file. When you use the LOAD routine this routine is automatically called to find out the format of your file.
5. Loading files. This routine will load a Datafile and call the header reading routine. The name of the file should be in Q$ on entry to the routine. If you don't know the name then let Q$ = "".
6. Find field number. The name of the field required should be in Q$ on entry to the routine. The routine will then find out the number of that field and return the result in FN. The value of FN allows you to get the start and end positions of that field from the numeric array F( ) which will have been filled by the header read routine. F (FN,1) holds the start position and F(FN,2) the end position. Thus, field 1 of record 1 extends from A$(1,F(1,1) to A$(1,F(1,2)).
7. Finding a record number. This routine returns in RN the number of the record which has a value equal to that of Q$ in the field designated by FN. You should define Q$ and FN before entry to the routine. The record number is returned by the routine in RN.

LETTERS: A SAMPLE APPLICATION

If you followed the example that I have given you will now have a file called CLIENT. LETTERS will input this file from tape and then write a letter to each client reminding them of the account that they have outstanding. Now you know why your own details should be entered first!

The listing of LETTERS should help you to develop your own applications. I hope that Datafile meets all your requirements.

```
  10 REM Tasman Interface Softwa
re
  12 REM @ Tasman Software 1983
  15 LET base=64716: LET len=652
: GO TO 50
  20 CLEAR 64715: GO SUB 4000: P
RINT  FLASH 1; AT 10,4;"Do NOT s
top the tape"; AT 11,4;"until th
e bytes have"; AT 12,11;"loaded.
"
  30 LET base=64716: LET len=652
```

```
  0> REM @ N.E.SALT 1983
  4 REM **********************
      *   DATAFILE           *
      *VARIABLES:-            *
      *N$=NAME OF FILE       *
      *A$(1)=DUMMY RECORD    *
      *NR=NUMBER OF RECS     *
      *SR=RECORD SIZE        *
      *NF=NUMBER OF FIELDS   *
      *F$(I)=NAME OF FIELD   *
      *F(I,1)=START FIELD    *
      *F(I,2)=END   FIELD    *
      *NRU=NO. RECS USED     *
```

```
  5 REM *FN=NO. OF FIELD      *
        *   FOUND BY SEARCH  *
        *   ROUTINE          *
        *RN=NO. OF RECORD     *
        *   FOUND BY SEARCH  *
        *   ROUTINE          *
        **********************
 98 CLS
 100 REM **********************
        *   SETS CAPS LOCK   *
        **********************

 101 IF  INT ( PEEK 23658/8)=2*
INT ( INT ( PEEK 23658/8)/2) THE
N  POKE 23658, PEEK 23658+8
 110 CLS : PRINT  AT 0,8; INVERS
E 1;"DATAFILE MENU"
 120 PRINT '"(1) DEFINE FILE"''"
(2) ENTER DATA"''"(3) SORT FILE"
''"(4) ALTER RECORD"''"(5) DELET
E RECORD"''"(6) SAVE FILE";
 121 PRINT ''"(7) LOAD FILE"''"(
8) PRINT FILE DETAIL"''"(9) PRIN
T WHOLE FILE"''"(0) EXIT FROM PR
OGRAM"
 130 PRINT  AT 21,0; INVERSE 1;"
ENTER NO. OF OPTION REQD": INPUT
  LINE Q$
```

```
 140 IF Q$="1" THEN  GO SUB 1000
 150 IF Q$="2" THEN  GO SUB 2500
 160 IF Q$="3" THEN  GO SUB 3500
 170 IF Q$="4" THEN  GO SUB 3700
 180 IF Q$="5" THEN  GO SUB 4000
 190 IF Q$="6" THEN  GO SUB 3000
 200 IF Q$="7" THEN  GO SUB 3100
 210 IF Q$="8" THEN  GO SUB 3200
 220 IF Q$="9" THEN  GO SUB 3300
 230 IF Q$="0" THEN  STOP
 300 GO TO 100
 999 REM **********************
        *HEADER    CREATE   *
        **********************
1000 CLS : INPUT "NAME OF FILE:
";n$
1010 INPUT "NUMBER OF RECORDS: "
;nr
1020 INPUT "RECORD SIZE: ";sr
1030 INPUT "NUMBER OF FIELDS: ";
nf
1035 LET start=1: DIM f$(nf,6):
DIM f(nf,2)
1036 CLS : PRINT "FILE NAME: ";n
$''nr;" RECORDS OF ";sr;" CHARAC
TERS"'';
1040 FOR i=1 TO nf
1050 PRINT ''"NAME OF FIELD ";i;
" "; FLASH 1;" "; CHR$ 8;: INPUT
 f$(i): PRINT f$(i)
1060 PRINT '"LENGTH OF FIELD ";i
;" "; FLASH 1;" "; CHR$ 8;: INPU
T sf: PRINT sf
1070 LET f(i,1)=start: LET start
=start+sf-1: LET f(i,2)=start: L
ET start=start+1
1080 NEXT i
1085 IF sr<nf*12+14 THEN  LET sr
=nf*12+14
1086 LET nru=1
1090 DIM a$(nr+1,sr)
1099 REM **********************
        *WRITE HEADER        *
        **********************
1100 LET a$(1,1)="9": LET a$(1,2
 TO 4)= STR$ nr: LET a$(1,5 TO 7
)= STR$ sr: LET a$(1,8 TO 10)= S
TR$ nf
1110 LET a$(1,11 TO 13)= STR$ nr
u
1200 FOR i=0 TO (nf-1)*12 STEP 1
2
1210 LET a$(1,14+i TO 14+i+5)=f$
(i/12+1)
1220 LET a$(1,14+i+6 TO 14+i+8)=
 STR$ f(i/12+1,1)
1230 LET a$(1,14+i+9 TO 14+i+11)
= STR$ f(i/12+1,2)
1250 NEXT i
1260 RETURN
1999 REM **********************
        *HEADER        READ  *
        **********************
2000 CLS : LET nr= VAL a$(1,2 TO
 4): LET sr= VAL a$(1,5 TO 7): L
ET nf= VAL a$(1,8 TO 10): LET nr
u= VAL a$(1,11 TO 13)
2010 DIM f$(nf,6): DIM f(nf,2)
2020 FOR i=0 TO (nf-1)*12 STEP 1
2
2030 LET f$(i/12+1)=a$(1,14+i TO
 14+i+5)
2040 LET f(i/12+1,1)= VAL a$(1,1
4+i+6 TO 14+i+8)
2050 LET f(i/12+1,2)= VAL a$(1,1
4+i+9 TO 14+i+11)
2060 NEXT i
2070 RETURN
2499 REM **********************
        *DATA ENTRY  SUB     *
        **********************
2500 CLS : LET nru=nru+1: LET a$
(1,11 TO 13)= STR$ nru
2505 FOR i=1 TO nf
2510 PRINT  AT 2*i,0;f$(i);"  ";
a$(nru,f(i,1) TO f(i,2)); FLASH
1;" "; FLASH 0
2511 INPUT q$: IF q$ <> "" THEN
 LET a$(nru,f(i,1) TO f(i,2))=q$
2520 PRINT  AT 2*i,0;f$(i);"  ";
a$(nru,f(i,1) TO f(i,2));"    "
2530 PRINT
2540 NEXT i
2545 INPUT "ALTER ANYTHING(Y/N)"
;Q$: IF Q$="Y" THEN  GO TO 2505
2550 INPUT "ENTER MORE RECS (Y/N
)";q$: IF q$="y" OR q$="Y" THEN
 GO TO 2500
2600 LET a$(1,11 TO 13)= STR$ nr
u
2610 CLS : RETURN
2999 REM **********************
        *     SAVE FILE      *
        **********************
3000 CLS
3010 SAVE N$ DATA A$()
3020 PRINT #1;"PREPARE FOR VERIF
ICATION AND"'"THEN PRESS <ENTER>
": PAUSE 0: CLS
3030 VERIFY N$ DATA A$()
3040 RETURN
3099 REM **********************
        *     LOAD FILE      *
        **********************
3100 CLS : INPUT "FILE NAME";N$
3110 LOAD N$ DATA A$()
3120 GO SUB 2000
3150 RETURN
```

```
3199 REM **********************
         *PRINT FILE DETAILS *
         **********************
3200 CLS : INPUT "HARD COPY (Y/N
)";Q$
3201 IF Q$="Y" THEN  OPEN # 2,"P
"
3203 CLS : PRINT  INVERSE 1;"FIE
LD"; AT 0,10;"START"; AT 0,20;"E
ND"''
3205 FOR i=1 TO nf
3210 PRINT f$(i); TAB 10;f(i,1);
 TAB 20;f(i,2)
3220 PRINT
3230 NEXT i
3235 PRINT "NUMBER RECS USED =";
nru
3250 OPEN # 2,"S"
3251 PRINT #1;"PRESS ANY KEY TO
CONTINUE": PAUSE 0
3260 RETURN
3299 REM **********************
         *PRINT WHOLE FILE   *
         **********************
3300 CLS : INPUT "HARD COPY (Y/N
)";Q$: IF Q$="Y" THEN  OPEN # 2,
"P"
3310 FOR i=2 TO nru
3320 CLS
3325 PRINT  TAB 10; INVERSE 1;"R
ECORD NUMBER: ";I''
3330 FOR j=1 TO nf
3340 PRINT f$(j); TAB 8;a$(i,f(j
,1) TO f(j,2)): PRINT ''
3350 NEXT j
3360 PRINT #1;"PRESS ANY KEY FOR
 NEXT RECORD": PAUSE 0: CLS
3370 NEXT I
3375 OPEN # 2,"S"
3380 RETURN
3499 REM **********************
         *SORT ROUTINE       *
         **********************
3500 GO SUB 5500
3540 IF  NOT fn THEN  RETURN
3541 PRINT  AT 10,10; FLASH 1; I
NK 2;"SORTING"
3545 LET sf=0
3560 FOR i=2 TO nru-1: IF a$(i+1
,f(fn,1) TO f(fn,2))<a$(i,f(fn,1
) TO f(fn,2)) THEN  LET s$=a$(i)
: LET a$(i)=a$(i+1): LET a$(i+1)
=s$: LET sf=sf+1
3570 NEXT i
3580 IF sf <> 0 THEN  GO TO 3545
3590 RETURN
3699 REM **********************
         *   ALTER ROUTINE   *
         **********************
3700 GO SUB 5500: IF  NOT FN THE
N  RETURN
3710 GO SUB 5600: IF  NOT RN THE
N  RETURN
3715 CLS
3720 FOR i=1 TO nf
3730 PRINT  AT 2*i,0;f$(i);"  ";
a$(rn,f(i,1) TO f(i,2)); FLASH 1
;" "; FLASH 0
3740 INPUT q$: IF q$ <> "" THEN
 LET a$(rn,f(i,1) TO f(i,2))=q$
3750 PRINT  AT 2*i,0;f$(i);"  ";
a$(rn,f(i,1) TO f(i,2));"    "
3760 PRINT
3770 NEXT I
3780 INPUT "ALTER ANYTHING(Y/N)"
;Q$: IF Q$="Y" THEN  GO TO 3720
3790 RETURN
3999 REM **********************
         *  DELETE ROUTINE   *
         **********************
4000 GO SUB 5500: IF  NOT fn THE
N  RETURN
4010 GO SUB 5600: IF  NOT rn THE
N  RETURN
4020 FOR i=rn+1 TO nru: LET a$(i
-1)=a$(i): NEXT i
4030 LET nru=nru-1: LET a$(1,11
TO 13)= STR$ nru
4040 RETURN
5499 REM **********************
         *FIND KEY FIELD NO. *
         **********************
5500 CLS : INPUT "NAME OF KEY FI
ELD: ";Q$
5505 LET fn=0
5506 IF q$="" THEN  RETURN
5510 FOR i=1 TO nf
5520 IF f$(i, TO  LEN'Q$)=q$ THE
N  LET fn=i
5530 NEXT i
5540 IF  NOT fn THEN  GO TO 5500
5550 RETURN
5599 REM **********************
         *FIND RECORD NUMBER *
         **********************
5600 CLS : INPUT "VALUE OF KEY:
";r$
5605 IF r$="" THEN  RETURN
5606 IF  LEN r$<f(fn,2)-f(fn,1)+
1 THEN  LET r$=r$+" ": GO TO 560
6
5610 LET rn=0
5630 FOR i=2 TO nru
5640 IF a$(i,f(fn,1) TO f(fn,2))
=r$ THEN  LET rn=i
5650 NEXT i
5660 IF  NOT rn THEN  GO TO 5600
5670 RETURN
```

```
1000 GO SUB 9970
1005 INPUT "DATE OF LETTERS";d$
1010 LET q$="ACCNO": GO SUB 9980
: LET n1=fn
1011 INPUT "HARD COPY(Y/N)";Q$:
IF Q$="Y" THEN  OPEN # 2,"P"
1020 LET q$="INVNO": GO SUB 9980
: LET n2=fn
1030 LET q$="NAME": GO SUB 9980:
 LET n3=fn
1040 LET q$="INVDAT": GO SUB 998
0: LET n4=fn
2000 FOR i=3 TO nru
2010 PRINT '''
2020 LET x1=i: LET x=n1: GO SUB
9920
2030 PRINT "ACCOUNT: ";q$'
2060 LET x1=i: LET x=n2: GO SUB
9920
2070 PRINT "INV No: ";q$;
2071 PRINT ''
2075 LET x1=2: LET x=n3: GO SUB
9920
2076 PRINT ' TAB 10;q$'
2080 LET x1=2: LET x=n3+1: GO SU
B 9920
2090 PRINT  TAB 10;q$'
3000 FOR j=2 TO 4
3010 LET x=n3+j: GO SUB 9920
3020 PRINT  TAB 10;q$
3030 NEXT j
3040 PRINT
3050 PRINT  TAB 10;"DATE: ";d$
3060 PRINT '''
3070 LET x1=i: LET x=n3: GO SUB
9920
3080 PRINT "Dear ";q$;","
3090 PRINT ''"   I refer to our
invoice dated"''
4000 LET x1=i: LET x=n4: GO SUB
9920: GO SUB 9940
4010 PRINT q$;". I would be grat
eful if "'''"you would settle the
 account"'''"as soon as possible.
"''
4020 PRINT  TAB 10;"Yours sincer
ely,"'''''; TAB 10;"N.E.SALT"'
4030 PRINT #1;"PRESS ANY KEY TO
CONTINUE": PAUSE 0: CLS
4040 NEXT i
4045 OPEN # 2,"S"
4050 STOP
```

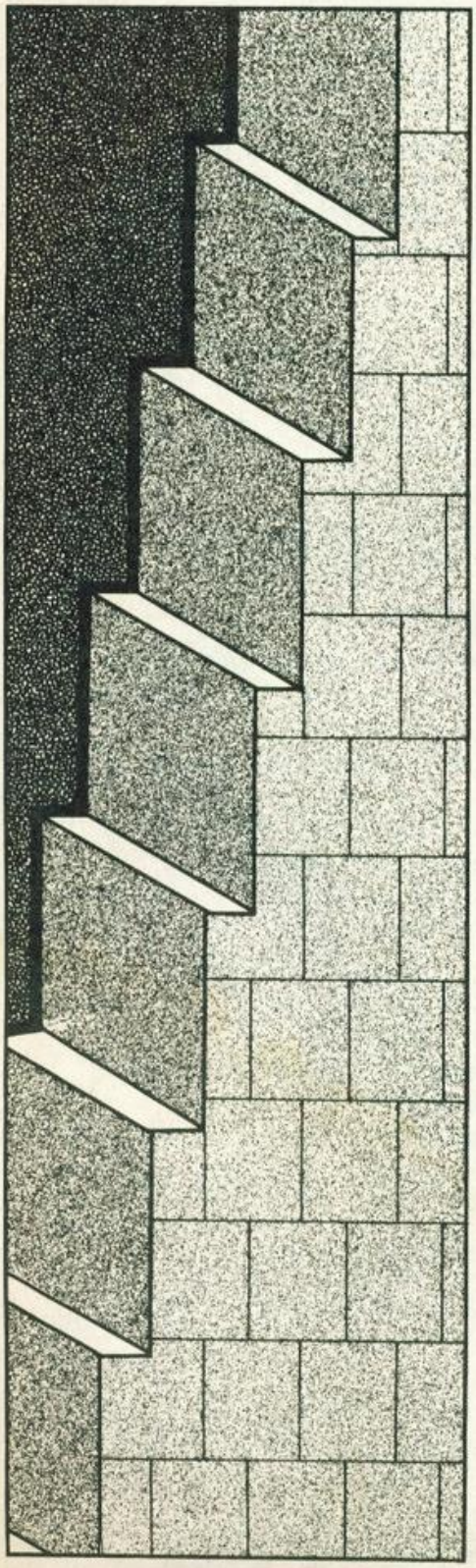
```
9900 REM **********************
         *       TOOLKIT      *
         *ROUTINES TO MERGE   *
         *WITH YOUR PROGRAMS  *
         *TO HANDLE FILES     *
         *CREATED BY DATAFILE*
         **********************
9901 REM ************************
         *ADDRESSES OF ROUTINES
         *9920:-GET FIELD      *
         *       CONTENTS      *
         *9940:-DECODE DATE    *
         *9950:-REMOVE SPACES  *
         *9960:-READ HEADER    *
         *9970:-LOAD FILE      *
         *9980:-GET FIELD NO.  *
         *9990:-GET RECORD NO.*
         ************************
9919 REM **********************
         *  GET FIELD VALUE   *
         *     FROM FILE      *
         *VARIABLES:          *
         *RN= RECORD NO. AND  *
         *     IS SUPPLIED BY *
         *     THE SUB AT 9990*
         *FN= FIELD NO. AND   *
         *     IS SUPPLIED BY *
         *     THE SUB AT 9980*
         *F() IS THE ARRAY    *
         *     DEFINED BY THE *
         *     SUB AT 9960 AND*
         *     IT HOLDS THE   *
         *     POSITION OF THE*
         *     THE FIELDS IN  *
         *     THE RECORD     *
         **********************
9920 LET q$=a$(rn,f(fn,1) TO f(f
n,2))
9922 GO SUB 9950
9923 RETURN
9939 REM **********************
         *    DATE DECODE     *
         *VARIABLES:          *
         *Q$ IS A DATE IN THE*
         *   FORMAT YY/MM/DD *
         *   THIS FORMAT CAN *
         *   BE USED TO MAKE *
         *   SORTING BY DATE *
         *   WITHIN DATAFILE *
         *   EASIER          *
         *   ON EXIT FROM THE*
         *   SUB Q$ HOLDS THE*
         *   DATE IN DD/MM/YY*
         *   FORMAT          *
         **********************
9940 LET q$=q$+q$(1 TO 2)
9941 LET q$( TO 2)=q$(7 TO 8)
9942 LET q$(7 TO 8)=q$(9 TO 10)
9943 LET q$=q$( TO 8)
```

```
9945 RETURN
9949 REM ************************
         *   REMOVE TRAILING  *
         *       SPACES       *
         *VARIABLES:          *
         *Q$ IS A FIELD GIVEN *
         *   BY THE USER OR   *
         *   THE SUB AT 9920  *
         *   THE ROUTINE REMOV*
         *   ES ANY SPACES AT *
         *   THE END OF THE   *
         *   STRING TO AVOID  *
         *   UNSIGHTLY GAPS IN*
         *   PRINTED TEXT     *
         ************************
9950 IF q$( LEN q$)=" " THEN  LE
T q$=q$( TO  LEN q$-1): GO TO 99
50
9951 RETURN
9960 REM ************************
         *HEADER        READ  *
         *VARIABLES:          *
         *A$(NR,SR)=A FILE    *
         *          CREATED BY*
         *          DATAFILE  *
         *NR=NUMBER OF RECS   *
         *SR=SIZE OF EACH REC*
         *NF=NUMBER OF FIELDS*
         *NRU=NUMBER OF RECS  *
         *    USED            *
         *F$(NF,6) HOLDS THE  *
         *    NAMES OF FIELDS*
         *    ON THE FILE     *
         *F(NF,2)  HOLDS THE  *
         *    POSITION OF THE*
         *    FIELDS WITHIN   *
         *    EACH RECORD     *
         ************************
9961 CLS : LET nr= VAL a$(1,2 TO
 4): LET sr= VAL a$(1,5 TO 7): L
ET nf= VAL a$(1,8 TO 10): LET nr
u= VAL a$(1,11 TO 13)
9962 DIM f$(nf,6): DIM f(nf,2)
9963 FOR i=0 TO (nf-1)*12 STEP 1
2
9964 LET f$(i/12+1)=a$(1,14+i TO
 14+i+5)
9965 LET f(i/12+1,1)= VAL a$(1,1
4+i+6 TO 14+i+8)
9966 LET f(i/12+1,2)= VAL a$(1,1
4+i+9 TO 14+i+11)
9967 NEXT i
9968 RETURN
9969 REM ************************
         *     LOAD FILE       *
         *VARIABLES:          *
         *Q$= THE NAME GIVEN  *
         *    BY THE USER TO  *
         *    A FILE CREATED  *
         *    BY DATAFILE     *
         *A$= THE ARRAY IN    *
         *    WHICH DATAFILE  *
         *    STORES FILES    *
         ************************
9970 CLS
9971 LOAD q$ DATA A$()
9972 GO SUB 9961
9973 RETURN
9980 REM ************************
         *FIND KEY FIELD NO.  *
         *VARIABLES:          *
         *Q$= THE NAME OF THE*
         *  FIELD TO BE FOUND*
         *F$(NF,6) HOLDS THE  *
         *  NAMES OF FIELDS   *
         *  ON FILE           *
         *FN= THE NUMBER OF   *
         *  FIELD CALLED Q$   *
         *  IF IT IS ON FILE  *
         *  AND IF NOT THEN 0*
         ************************
9981 CLS
9982 LET fn=0
9983 IF q$="" THEN  RETURN
9984 FOR i=1 TO nf
9985 IF f$(i, TO  LEN q$)=q$ THE
N  LET fn=i
9986 NEXT i
9987 IF  NOT fn THEN  CLS : PRIN
T "FIELD ";q$;" NOT IN THIS FILE
": STOP
9988 RETURN
9989 REM ************************
         *FIND RECORD NUMBER  *
         *VARIABLES:          *
         *Q$ HOLDS THE VALUE  *
         *   OF FIELD NUMBER  *
         *   FN THAT IS REQD  *
         *RN= THE NUMBER OF   *
         *   THE RECORD WHICH*
         *   HAS Q$ IN FIELD  *
         *   NUMBER FN OR 0   *
         *   IF NO SUCH RECORD*
         *   EXISTS           *
         ************************
9991 IF q$="" THEN  RETURN
9992 IF  LEN q$<f(fn,2)-f(fn,1)+
1 THEN  LET q$=q$+" ": GO TO 999
2
9993 LET rn=0
9994 FOR i=2 TO nru
9995 IF a$(i,f(fn,1) TO f(fn,2))
=q$ THEN  LET rn=i
9996 NEXT i
9997 IF  NOT rn THEN  CLS : PRIN
T "RECORD WITH KEY VALUE ";q$;'"
NOT FOUND": STOP
9998 RETURN
```

RAY ELDER

One Step Beyond

Programs may work perfectly well but still need a touch of expert development. Some ideas to get you thinking.



So you've done the really hard work, thought of something to write, sat down and produced a working, debugged program. You are pleased. The rest of the family make approving sounds and you decide that you will send your brainchild to be published. Some weeks later an envelope drops through the door and you rush to open it to find out just how many of those blue notes they are offering.

Shock! Horror! They are so unaware of real talent that they have rejected your masterpiece. Obviously they're pretty stupid so you cancel your subscription and look for another, more appreciative magazine.

Probably 75% or more of the programs submitted to magazines are rejected because they already have a similar program awaiting publication. The new one may have different features or been programmed differently, but once an editor has accepted a program then sooner or later we will publish it.

Most of the other rejections are because the program has been left in an unrefined state. What do I mean by that?

Steven Bridge has very generously given me permission to use his program to try and demonstrate how you can refine a program. I do appreciate how difficult it is to accept critism and to allow your work to be criticised in print must be twice as bad, so my most heartfelt thanks to him for taking the hot seat!

STEVE'S PROGRAM

Take a look at Program 1. This is Steve's original program, it has no drastic bug, (there is one minor one that is not fatal — can you spot it?), and it works fine, performing the task intended.

So what's wrong with it? Answer, nothing! But let's take what I call "the next step".

The first thing I noticed was the system of producing the superb music, obviously Steve had developed this note in the Data statement and used a system where 999 was the "rogue value" which signalled the end of that section.

However, now that the music was completely developed it is no longer essential. Also it seems a shame to only use it once. With this in mind I created a subroutine at 8000-8020 in my program (program 2).

The RESTORE allows me to call it more than once and find the correct starting point of the Data each time, knowing the exact length allowed a FOR/NEXT loop to be used, variables a and b can both be READ in one go and the whole lot fits in one line.

The next thing I noticed was the frequent use of PAUSE 100: CLS and so I thought another subroutine would be in order. On my listing I created this subroutine at line 8100.

PAUSING

Notice I used an empty loop instead of PAUSE. This is partly a leftover technique from my ZX81 days when PAUSE caused a screen flicker and an occasional crash. However I still use it on the Spectrum because on occasions a player may be slow to remove his finger from a key and cause the pause to be ignored. Remember that a pause is cut short by a key being pressed! Variable qu is initialised in Steve's program but never used. He was probably going to use it as a question counter then developed his program in another way. As it is redundant we ought to remove it for neatness' sake.

Steve owns a 16K Spectrum, so, for the sake of memory saving, I reduced variable "score" to "sc". It would have been better to have used "s" but I wanted to keep a balance between economy and legibility. For the latter Steve's name is the ideal!

THE BIG STEP

So much for the simple, nit picking modifications. To get a good overall view of his program, and in fact any program, it is essential to get a print out of the listing.

With a listing available the pattern of the program emerged and I traced it through looking for repetitive actions, initially in order to create subroutines but soon I became aware that it could be summarised in a simple form as follows:

1 Print at the same position the question number.
2 Print at the same position the question and 3 options.
3 Get an input.
4 Beep.
5 Check answer, if wrong print this at the same position and give correct answer.
6 Check answer, if correct print this, add 10 to score.
7 Next question.

The only things that changed were the questions, the options and the position the correct answer was in.

As so much was repeated I looked for a way to set up a format and use Data for the different questions each time.

The obvious one was the one I used. Each question was set up as a DATA line starting from line 9000, and each line consists of:

The question — a string variable
The 3 options — numeric variables
The position of the answer in the options — a numeric variable

This makes it simplicity itself to amend, alter, increase or replace any of the questions as they all follow the same layout. To store the three options I used a DIMensioned array. This would prove easier to find when printing the correct answer. Also while I was about it I thought it might be nice to give the correct answer even if the player had chosen correctly, a confirmation of his choice.

Finally I used variable a (for answer) to hold the position in the options of the correct answer.

THE FINAL STEP

Line 150 RESTOREd the Data pointer to the start of the questions and line 200 set up a loop counter i.

I decided that as we knew how many questions were to be asked a main loop was suitable.

Using the value of "i" it was straightforward to print up the question number as Steve had in his original. The data was READ into q$, each element of c() and "a" then the pause, cls subroutine was called. All in the one line.

Line 210 prints out all the information. Notice the loop "j" to print each c() element, and how I utilised its value to provide suitable spacing and the numbering. This was found by trial and error. Quite often a fair bit of time needs to be spent on just formatting the screen to give a balanced, effective display. Of course, each person has his own preferences for the way a screen is laid out, this only happens to be mine.

Line 212 gets the input. Here I used INKEY$. My reason was that the less key-pressing the user did the more he could concentrate on the questions. The fact that only a single number from one to three was required made it most appropriate. I also introduced a simple but effective check that a suitable key was pressed, all other keys (except BREAK of course) are ignored.

Line 215 provides the BEEP then goes on to check for the correct answer by comparing the VALue of the input a$ to variable "a" read from the Data. The variable "a" is also used to identify the correct element of c() for printing in both "correct" and "wrong" messages.

The line then increases the score and jumps over line 216 to line 220. Line 216 is the "wrong" message and is only reached if VAL a$ is not equal to "a" when checked by line 215.

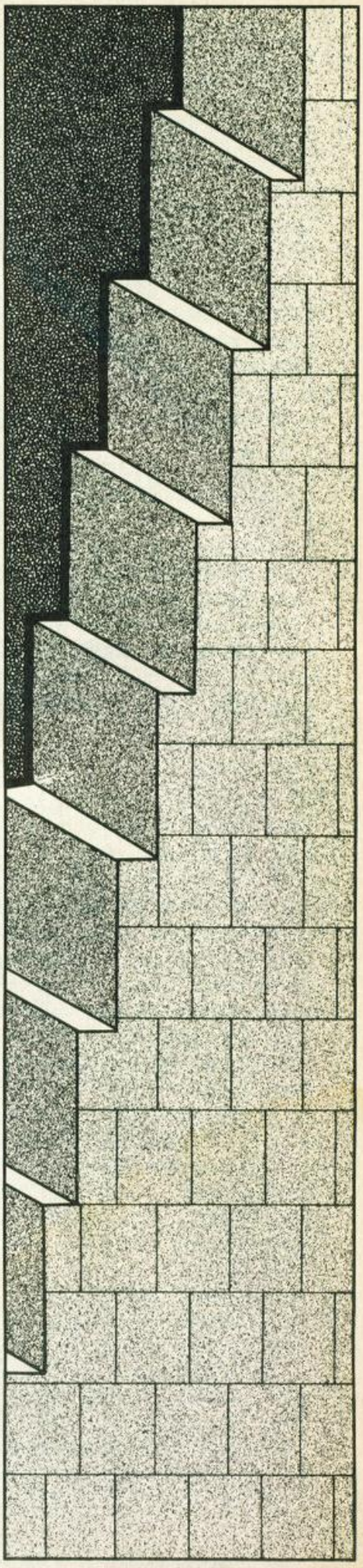
Lines 600 to 701 uses PAUSE 0 to wait for key. At this stage there has been a lengthy gap since the last required key press so there should not be much chance of lingering fingers.

FURTHER THOUGHTS

Well, Steve agreed with me that this program was much more efficient and easily adaptable than before, and we talked about other modifications it was now possible to make with the extra memory available.

Most of these we'll leave for you to experiment with. The first one was that, for the sake of user friendliness, a prompt to "Press 1,2 or 3" would be nice after each question the prompt removed on a valid input.

Then we thought that we could increase the number of questions. A possible idea — especially with a 48K machine — would have to be twice the number of questions in memory as were required to be asked and chose them at random. This would make for tremendous variation each time the game was played.



A system of checking that a question wasn't repeated would need devising for this!

We even mused on a way to present the options in a differnt order so that the data had to be remembered and not the position in the options. For this one we created these lines:

```
205 FORj=1 TO 4
206 LETx=INT (RND*3)+1:
LET y=INT (RND*3)+1: IF x=y
THEN GO TO 206
207 IFx=a THEN LET a=y: GO TO
209
208 IF y=a THEN LET a=x
209 LET z=c(x): LET
c(x)=c(y): LEY=c(y)=Z: NEXTj
```

This swaps two elements of c() at random, four times. Lines 207/208 keep track of the position of the correct answer.

Another idea was that the option data could be made into strings so that some answers could be words as well as numbers and open up a whole new range of questions. All that would be necessary would be to DIMension c$(3,4) and use c$() instead of c(), the number 4 would be altered to the length of the longest word in the answer.

As it was now getting to 1.30am and I had to write this lot up, we parted.

And so, here endeth the lesson.....

```
  2 LET score=0: LET qu=1
 100 BORDER 0: PAPER 0: INK 7: C
LS
 110 PRINT AT 4,7;"Q U I Z M A S
 T E R"
 112 PRINT AT 6,6; INK 4; BRIGHT
 1; FLASH 1;"Press any to to sta
rt"
 113 PAUSE 0: CLS
 115 PRINT AT 8,13;"QUESTION 1"
 116 READ a: IF a=999 THEN  GO T
O 200
 117 READ b: BEEP a/10,b: GO TO
116
 118 DATA 2,7,2,7,1,9,1,12,1,11,
1,9,2,14,2,14
 119 DATA 1,14,1,16,1,11,1,12,2,
9,2,9,1,9,1,12,1,11,1,9,2,7
 200 DATA 999
 201 CLS
 210 PRINT AT 2,0;"In what year
did Captain Scott  Reach the sou
th pole."
 213 PRINT AT 5,0;"1.1912 2.1911
 3.1910"
 214 INPUT  LINE a$
 215 IF a$<>"1" THEN  BEEP .1,1:
 PRINT AT 10,3;"WRONG THE ANSWER
 WAS 1912"
 216 IF a$="1" THEN  BEEP .1,2:
PRINT AT 10,13; FLASH 1;"CORRECT
": LET score=score+10
 217 PAUSE 100: CLS
 218 PRINT AT 8,13;"QUESTION 2":
 PAUSE 100
 220 CLS
 223 PRINT AT 2,0;"In what year
did London Airport open."
 224 PRINT AT 5,0;"1.1945  2.194
6  3.1943"
 225 INPUT  LINE a$
 226 IF a$<>"2" THEN  BEEP .1,1:
 PRINT AT 10,3;"WRONG THE ANSWER
 WAS 1946"
 227 IF a$="2" THEN  BEEP .1,1:
PRINT AT 10,13;"CORRECT": LET sc
ore=score+10
 228 PAUSE 100: CLS
 230 PRINT AT 8,13;"QUESTION 3":
 PAUSE 100: CLS
 231 PRINT AT 2,0;"In what year
was the first      London Marath
on."
 232 PRINT AT 5,0;"1.1982  2.198
1  3.1980"
 233 INPUT  LINE a$
 234 IF a$<>"2" THEN  BEEP .1,1:
 PRINT AT 10,3;"WRONG THE ANSWER
 WAS 1981"
 235 IF a$="2" THEN  BEEP .1,1:
PRINT AT 10,13;"CORRECT": LET sc
ore=score+10
 300 PAUSE 100: CLS
 310 PRINT AT 8,13;"QUESTION 4"
 312 PAUSE 100: CLS
 314 PRINT AT 2,0;"In What year
were the cats eyes first manufac
tured."
 315 PRINT AT 5,0;"1.1934  2.193
6  3.1935"
 316 INPUT  LINE a$
 317 IF a$<>"3" THEN  BEEP .1,1:
 PRINT AT 10,3;"WRONG THE ANSWER
 WAS 1935"
 318 IF a$="3" THEN  BEEP .1,1:
PRINT AT 10,13;"CORRECT": LET sc
ore=score+10
 320 PAUSE 100: CLS : PRINT AT 8
,13;"QUESTION 5": PAUSE 100: CLS
 321 PRINT AT 2,0;"In what year
was Halley's       comet first s
ighted."
 332 PRINT AT 5,0;"1.1910  2.191
1  3.1909"
 400 INPUT  LINE a$
 445 IF a$<>"1" THEN  BEEP .1,1:
 PRINT AT 10,3;"WRONG THE ANSWER
 WAS 1910": PAUSE 100: CLS
 450 IF a$="1" THEN  PRINT AT 10
,13;"CORRECT": LET score=score+1
0: PAUSE 100: CLS
```

```
 460 PRINT AT 8,13;"QUESTION 7":
 PAUSE 100: CLS
 467 PRINT AT 2,0;"In What year
was the opening of the forth bri
dge."
 468 PRINT AT 5,0;"1.1964  2.196
1 3.1963": INPUT  LINE a$
 469 IF a$<>"3" THEN  BEEP .1,1:
 PRINT AT 10,3;"WRONG THE ANSWER
 WAS 1963": PAUSE 100: CLS
 470 IF a$="3" THEN  BEEP .1,1:
PRINT AT 10,13;"CORRECT": LET sc
ore=score+10: PAUSE 100: CLS
 473 PRINT AT 8,13;"QUESTION 8":
 PAUSE 100: CLS
 474 PRINT AT 2,0;"In what year
did England win theworld cup."
 475 PRINT AT 5,0;"1.1966  2.196
7  3.1964": INPUT  LINE a$
 480 IF a$<>"1" THEN  BEEP .1,1:
 PRINT AT 10,3;"WRONG THE ANSWER
 WAS 1966": PAUSE 100: CLS
 481 IF a$="1" THEN  BEEP .1,1:
PRINT AT 10,13;"CORRECT": LET sc
ore=score+10: PAUSE 100: CLS
 490 PRINT AT 8,13;"QUESTION 9":
 PAUSE 100: CLS
 500 PRINT AT 2,0;"In what year
was oil discovered in the north
sea by B.P."
 510 PRINT AT 5,0;"1.1970  2.196
9  3.1973": INPUT  LINE a$
 511 IF a$<>"1" THEN  BEEP .1,1:
 PRINT AT 10,3;"WRONG THE ANSWER
 WAS 1970"
 512 IF a$="1" THEN  BEEP .1,1:
PRINT AT 10,13;"CORRECT": LET sc
ore=score+10
 513 PAUSE 100: CLS
 600 PRINT AT 10,14;"Game over":
 PAUSE 100: CLS
 700 PRINT AT 4,6;"Y O U R  S C
O R E = ";score
 701 PRINT AT 7,4; BRIGHT 1; INK
 2; FLASH 1;"Press any key to pl
ay again"
 702 INPUT  LINE a$: IF a$="y" T
HEN  RUN
 703 STOP
 800 FOR a=0 TO 3: SAVE "Quizmas
ter" LINE 1: NEXT a
```

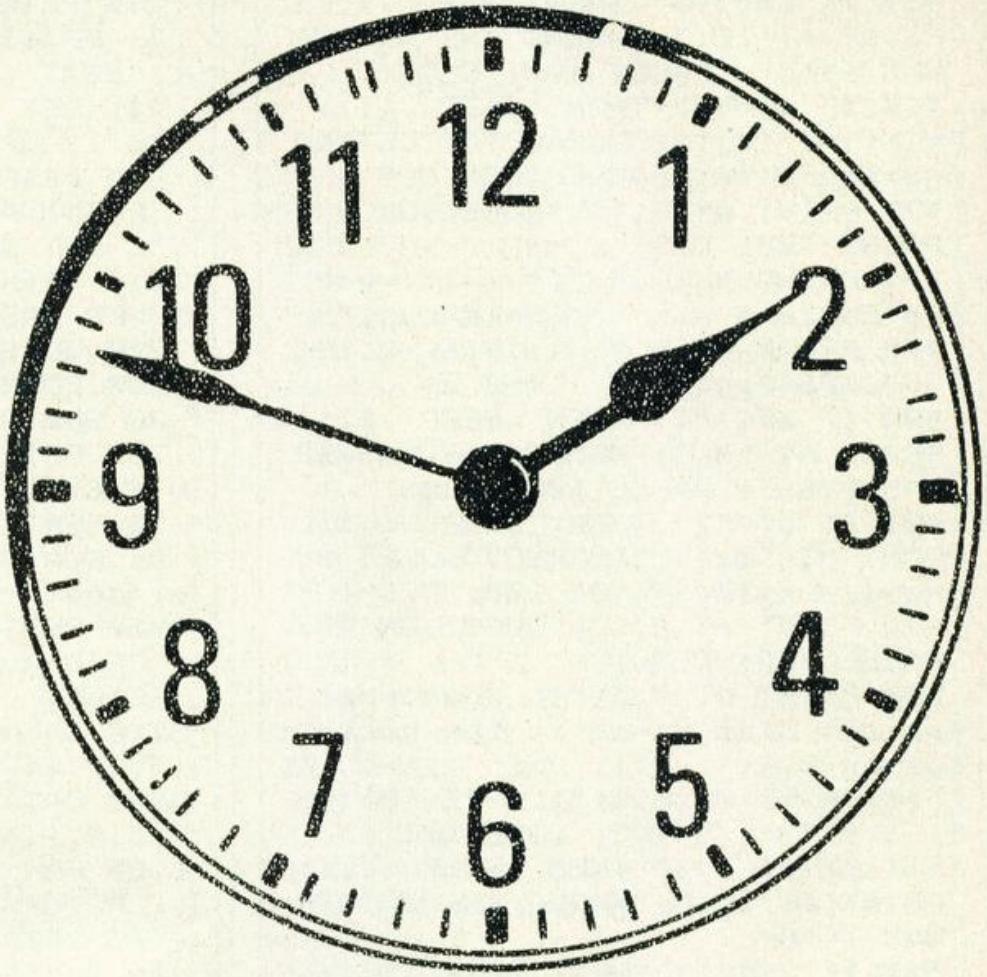
```
 100 LET sc=0: DIM c(3): BORDER
0: PAPER 0: INK 7: CLS
 120 PRINT AT 4,7;"Q U I Z M A S
 T E R";AT 6,6; INK 4; BRIGHT 1;
 FLASH 1;"Press a key to start."
 150 PAUSE 0: GO SUB 8000: CLS :
 RESTORE 9000
 200 FOR i=1 TO 8: PRINT AT 8,13
;"QUESTION ";i: READ q$,c(1),c(2
),c(3),a: GO SUB 8100
 210 PRINT AT 2,0;q$: FOR j=1 TO
 3: PRINT AT 4+j*2,13;j;" ";c(j)
;: NEXT j
 212 LET a$=INKEY$: IF a$="" OR
a$<"1" OR a$>"3" THEN  GO TO 212
 215 BEEP .1,1: IF VAL a$=a THEN
  PRINT AT 14,9;c(a);" is CORREC
T": LET sc=sc+10: GO TO 220
 216 PRINT AT 14,3;"WRONG, THE A
NSWER WAS ";c(a)
 220 GO SUB 8100: NEXT i
 600 PRINT AT 10,14;"Game over":
 GO SUB 8100
 700 PRINT AT 4,6;"Y O U R  S C
O R E = ";sc: GO SUB 8000
 701 PRINT AT 7,4; BRIGHT 1; INK
 2; FLASH 1;"Press any key to pl
ay again": PAUSE 0: RUN
8000 RESTORE 8010: FOR i=1 TO 19
: READ a,b: BEEP a/10,b: NEXT i:
 RETURN
8010 DATA 2,7,2,7,1,9,1,12,1,11,
1,9,2,14,2,14
8020 DATA 1,14,1,16,1,11,1,12,2,
9,2,9,1,9,1,12,1,11,1,9,2,7
8100 FOR j=1 TO 300: NEXT j: CLS
 : RETURN
9000 DATA "In what year did Capt
ain Scott  Reach the south pole.
",1912,1911,1910,1
9010 DATA "In what year did Lond
on Airport open.",1945,1946,1943
,2
9020 DATA "In what year was the
first       London Marathon.",198
2,1981,1980,2
9030 DATA "In What year were the
 cats eyes first manufactured.",
1934,1936,1935,3
9040 DATA "In what year was Hall
ey's        comet first sighted."
,1910,1911,1909,1
9050 DATA "In What year was the
opening of the forth bridge.",19
64,1961,1963,3
9060 DATA "In what year did Engl
and win theWorld Cup.",1966,1967
,1964,1
9070 DATA "In what year was oil
discovered in the north sea by B
.P.",1970,1969,1973,1
```

DOUGLAS RICHARDSON

Time Out

It may not fit on your wrist, but here's an easy way to turn your Spectrum into a wristwatch.

Chapter 18 of the Spectrum manual describes two analogue clock programs: the first version uses PAUSE as the 'hairspring' but, as the manual says, there is a difference of 1/50 between consecutive PAUSE numbers equivalent to half an hour per day. By altering the PAUSE valve you could probably increase the accuracy.



DIGITAL

This program prints out a digital watch and, as the result is displayed as a number, setting and synchronising are much easier — the PAUSE is compounded to give this accuracy. The performance of the watch then is dependent more on the stability of the timing delay function on the computer, rather then the ultimate accuracy of the internal clock.

The PAUSE at line 110 is the main reference, it allows a slight gain which is essential if a following PAUSE is to be corrective.

CALIBRATION

With PAUSE 49 at line 110 and REMs at 135 and 153, (see setting) run the watch for one hour. Seconds gained per hour times 5/6 rounded to the nearest integer gives you the PAUSE to be entered at line 135 in place of the REM. You must round down if you intend to use a third PAUSE for maximum accuracy. In the author's first check the gain was five seconds per hour:

5X5/6 = 4.16667

PAUSE 4 was then used at line 135.

If you opt for maximum precision then with PAUSEs entered at lines 110 and 135, and a REM at line 153, run the clock over a much greater length of time. Seconds gained per hour x50 will give you the PAUSE value to be entered.

Looking back to the author's example, the third PAUSE was not necessary as the error was only two minutes per day with one PAUSE, and 4.8 seconds per day with two PAUSEs. However, your Spectrum may be different and require this fine tuning.

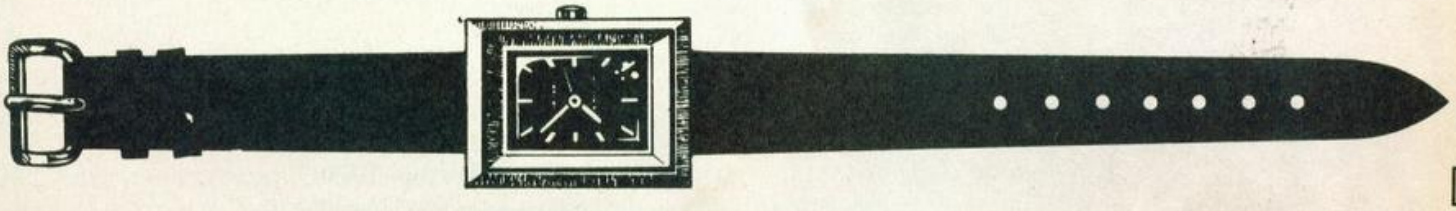
Subject to stability, the target accuracy should be 24/50 seconds per 24 hours. This would give you an error of one minute per 500 days!

SETTING

For ease of setting, the seconds increment by five, but run normally once set. Set the time with the seconds just ahead of real time then press G when the time is correct. Note: line 730 Graphics is "1". Data is for effect only and may be changed at line 400.

```
 7 REM ***DIGITAL WRIST WATCH
10 REM ************************
15 REM             D.Richardson
20 REM ************************
30 GO SUB 250
39 REM ************************
40 REM                  Set time
41 REM _____________________
45 IF  INKEY$  <> "" THEN  GO TO 45
50 IF  INKEY$ ="" THEN  GO TO 50
55 IF  INKEY$ ="h" THEN  PRINT  AT 10,17;h+1: LET h=h+1: IF h>12 THEN  LET h=1
```

```
: PRINT h$;"1 ": GO TO 45
  60 IF  INKEY$ ="m" THEN  PRINT m$;m+1: LET m=m+1: IF m>59 THEN  LET m=0: PRINT
 m$;"0 ": GO TO 45
  65 IF  INKEY$ ="s" THEN  PRINT s$;s+5: LET s=s+5: IF s>59 THEN  LET s=0: PRINT
 s$;"0 ": GO TO 45
  70 IF  INKEY$ ="p" THEN  LET r$="PM": PRINT a$;r$: GO TO 45
  75 IF  INKEY$ ="a" THEN  LET r$="AM": PRINT a$;r$: GO TO 45
  80 IF  INKEY$  <> "g" THEN  GO TO 45
  85 FOR n=0 TO 11: PRINT  AT n,0;"              ": NEXT n
  90 PRINT  AT 15,2;"As ""TIME""": PRINT  AT 16,5;"goes by"
  99 REM ************************
 100 REM              Clockworks          ______________________
 110 PAUSE 49
 120 LET s=s+1
 130 IF s<60 THEN  PRINT s$;s: GO TO 110
 135 REM ANOTHER PAUSE? SEE TEXT
 140 LET s=0: PRINT s$;"0 "
 145 LET m=m+1: IF m<60 THEN  PRINT m$;m: GO TO 110
 153 REM FURTHER PAUSE? SEE TEXT
 155 LET m=0: PRINT m$;"0 "
 160 LET h=h+1: IF h<12 THEN  PRINT h$;h: GO TO 110
 165 IF h=12 THEN  PRINT h$;h: GO TO 195
 170 IF h=13 THEN  LET h=0: PRINT h$;" 1": GO TO 110
 195 IF r$="AM" THEN  LET r$="PM": PRINT a$;r$: GO TO 110
 200 IF r$="PM" THEN  LET r$="AM": PRINT a$;r$: GO TO 110
 244 REM ************************
 245 REM              Build Watch         ______________________
 250 LET h=0: LET m=0: LET s=0
 255 BORDER 0
 260 LET h$= CHR$ 22+ CHR$ 10+ CHR$ 17
 270 LET m$= CHR$ 22+ CHR$ 10+ CHR$ 20
 280 LET s$= CHR$ 22+ CHR$ 10+ CHR$ 23
 285 PRINT h$;h,m$;m,s$;s
 290 LET r$="AM"
 295 LET a$= CHR$ 22+ CHR$ 14+ CHR$ 17: PRINT a$;r$
 300 PRINT  AT 0,0;"To set PRESS": PRINT " H for hours": PRINT " M  -  mins": PR
INT " S  -  secs": PRINT " AorP-AM/PM": PRINT '"   To START": PRINT "   PRESS 'G
'": PRINT ''" (Stopwatch": PRINT "    only""G"")"
 350 PRINT  AT 12,17;"Hr"; AT 12,20;"Mn"; AT 12,23;"Sc"
 400 PRINT  AT 14,21;"SA19"
 500 PRINT  AT 7,18;"ASEIKO"
 510 PRINT  AT 16,18;"Quartz"
 520 PLOT 143,128: DRAW 0,-97,.72* PI
 530 PLOT 192,31: DRAW 0,97,.72* PI
 540 PLOT 143,175: DRAW 0,-51: DRAW 0,-89,.67* PI : DRAW 0,-35
 550 PLOT 192,0: DRAW 0,35: DRAW 0,89,.67* PI : DRAW 0,51
 560 FOR y=0 TO 35 STEP 5: PLOT 143,y: DRAW 49,0: NEXT y
 570 FOR y=124 TO 175 STEP 5: PLOT 143,y: DRAW 49,0: NEXT y
 580 PLOT 203,99: DRAW 0,-15: DRAW -71,0: DRAW 0,15: DRAW 71,0
 590 PLOT 128,105: DRAW 79,0: DRAW 0,-51: DRAW -79,0: DRAW 0,51
 700 FOR n=0 TO 7
 710 READ a
 720 POKE  USR "1"+n,a: NEXT n
 730 PRINT  AT 12,28;"L"
 740 RETURN
 750 DATA 248,8,8,248,0,0,0,0
```



DEREK POPE

BASIC Break

An easy way of disabling the Break key from within a BASIC program.

ON-ERROR processing is one of the few useful features omitted on the Sinclair machines but I realised quite recently that it would be possible (with the aid of a small machine code routine) to provide such a function. In the section of the Spectrum manual which contains details of the system variables (Chapter 25) there is a WORD entry called ERR SP at address 23613 which is described as the "Address of item on machine stack to be used as error return", which I examined by using the PEEK command in a function as shown below.

NO NONSENSE IN BASIC

It was obvious that the word at 23613 pointed at an entry in the machine stack and this was printed giving the value 4867 which is within the Sinclair ROM. I wrote a small machine code routine (not included here) which simply jumped to 4867 after placing a required value into the system variable ERR NR and found that I got the appropriate error report as shown in the Sinclair manual in appedix B. It seemed reasonable to assume that, if the stack entry pointed to by ERR SP was altered to address a machine code routine other than 4867, then any subsequent error would cause the alternative routine to execute.

I developed the machine code routine shown quite quickly and it worked for all errors except 12 (C in the Sinclair manual) "Nonsense in BASIC" which, despite considerable further effort, I could not trap without putting the machine into a recursive loop. The machine code routine consists of two parts, the first labelled START is run from BASIC with an appropriate USR nnnn call and this places the address of the second part, TRAP, into the machine stack entry pointed to by ERR SP. The TRAP routine is then entered on any "error" completion (from 0 upwards).

NAMES AND ADDRESSES

Although comprehensive comments have been included with the machine code, a little additional explanation is probably worthwhile, particularly since the routines were written to be run wherever they are placed in memory. First, the address of TRAP placed onto the machine stack must be the correct 'absolute address' regardless of where the routine is stored. This is calculated dynamically by the first two instructions, relying on the fact that the BC register pair contains the address of START on entry from the USR nnnn function. This absolute address is placed into the machine stack entry pointed to by ERR SP by the next five instructions. Secondly, the TRAP routine must be able to replace its own entry address onto the machine stack if it is to handle successive errors and although this is done by the two instructions starting at PUSH, the value used for TRAP in the LD DE, TRAP instruction must be modified in order to allow it to function correctly when loaded at differing memory addresses. The last five instructions in the START routine before RET place the corrected value into the instruction labelled PUSH to overcome this problem.

It must be stressed that instruction modification of this type is very bad programming practice and should only be used where there is no alternative. The modified intructions should be adequately explained as should those which carry out the modification. In this instance I could see no way of avoiding dynamic instruction modification but if anyone can offer an alternative approach I should be most interested to hear from them.

BASICALLY SPEAKING

When the trap routine is entered (on any BASIC completion) it first checks the given error code and passes control to the normal error handling code at 4867 if the error is either 0 (OK), 9 (STOP), or 12 (Nonsense in BASIC); this gives the BASIC programmer the option to allow his program to end normally when required by executing STOP or issuing a GOTO for a line past the end of the program. Without this way out the only way to terminate a program with TRAP incorporated would be to pull out the power lead, a point which will not be missed by anyone who wishes to "protect" the code of a BASIC program. As indicated at the beginning of the article error 12 seems impossible to recover from and is therefore also given back to the Spectrum routine to handle.

ROUTING DECISIONS

For any error other than 0,9 and 12 the TRAP stack entry is restored and then the number of the BASIC line chosen to begin ON-ERROR processing is placed into the system variable NWPPC. I chose line 9900, statement 1 but this could easily be changed as required by following the comments against the EQUates for GTOLL, GTOHL and GTOST. In order than the users' error handling routine (beginning at line 9900) can make routing decisions based on the line where the error occurred, the next few instructions save the line, statement and error numbers. These details could be placed into any spare memory locations (four bytes are required) but, since I do not have a printer, I often use the printer buffer as workspace and the routine stores the crash details there. The error line number is placed into the word at 23296, the statement number within that line is put into the byte at 23298 and the actual error number (increased by 1) is stored in the byte at 23299. It should be noted that the error is stored as a binary number, not in the equivalent character form given in the normal error reports expected from the BASIC; thus 21 would be shown following an attempted "BREAK" if the contents of 23299 was PEEKed.

Finally in the TRAP routine, the ERR NR is reset to 255 (its normal value) and a jump is made to the normal processing loop used when executing a BASIC program. The address (7030) of MAINL was found by a small routine (not included here) which printed the contents of the first few machine stack entries, which always shows 7030 immediately below the word with 4867. Although the routine could be used from a direct command it would serve little purpose since the trap address would be rest by the Spectrum routine when the command execution completed. The TRAP only needs to be set up once in a BASIC program and will continue to pass errors to line 9900 until an error 0.9 or 12 is encountered.

USES

Any BASIC programmer who has ever wished fo an ON-ERROR func-

tion will have few problems finding uses for this routine but, as a starter, it is fun to produce a small program which keeps going regardless of attempts to "BREAK" it and only finishes when it has done its job:

```
10    RANDOMISE USR nnnn
20    PRINT "STOP ME IF
      YOU CAN."
30    FOR a=1 TO 10000
40    PRINT AT 1,0;a
50    NEXT a
60    STOP
9900  PRINT AT 4,0;"NO, I
      WON'T."
9910  PAUSE 100
9920  CLS
9930  GOTO 40
```

Of course the value used for "nnnn" will depend on where you decide to place the TRAP routine. I have not included a loader with the routine details because so many good examples have been printed in the past that most people must be able to enter a machine code program without problems.

The function for displaying WORD length data from any memory location which was mentioned at the beginning of the article is:

```
10 DEF FN a(x)=PEEK x+256*PEEK
   (x+1)
```

whereupon PRINT FN a(23613) gives a value just below RAMTOP, and entering PRINT FN a(FN a(23613) should return 4867.

```
211400    START   LD      HL,OFFSET   ;get offset to trap from entry point
09                ADD     HL,BC       ;add entry address from BC reg pair
ED5B3D5C          LD      DE,(ERRSP)  ;get addr of stacked error return
EB                EX      DE,HL       ;HL now points at stacked error rtn
73                LD      (HL),E      ;changed stacked value to point at TRAP
23                INC     HL          ;one byte at a time...
72                LD      (HL),D      ;any error will now execute TRAP
212700            LD      HL,TRAPA    ;get offset to our trap restore instruction
09                ADD     HL,BC       ;add entry address as above
73                LD      (HL),E      ;and store this into the PUSH below
23                INC     HL          ;one byte at a time...
72                LD      (HL),D      ;to ensure code can be loaded anywhere
C9                RET                 ;go back to BASIC, setup is finished
                  TRAP ROUTINE ENTERED ON ANY ERROR
210313    TRAP    LD      HL,BTRAP    ;prepare to use SPECTRUM error rtn
3A3A5C            LD      A,(ERRNR)   ;get the error code
3C                INC     A           ;add one to get correct value
2808              JR      Z,JUMP      ;if the BASIC opted to finish, do so
FE09              CP      9           ;error 9 is STOP, if the BASIC opted
2804              JR      Z,JUMP      ;for this then do so
FE0C              CP      12          ;error 12(C) is NONSENSE IN BASIC, if
2001              JR      NZ,PUSH     ;we trap that we are recalled for ever
E9        JUMP    JP      (HL)        ;finish via BASIC error routine on 0,9,12
11EE5C    PUSH    LD      DE,TRAP     ;this instruction is 'modified' as set up
D5                PUSH    DE          ;restore our TRAP entry point for futures
21425C            LD      HL,NWPPC    ;force a 'GOTO 9900' when we return to
                                      BASIC
36AC              LD      (HL),GTOLL  ;low byte of line number 9900
23                INC     HL          ;step to NEWPPC + 1
3626              LD      (HL),GTOHL  ;high byte of line number 9900
23                INC     HL          ;step to NSPPC
3601              LD      (HL),GTOST  ;statement 1 of line 9900
23                INC     HL          ;step HL to PPC, the line we trapped from
11005B            LD      DE,PRTBF    ;point DEstination at the vacant buffer
010300            LD      BC,3        ;set a byte count of 3, and save trapped
EDB0              LDIR                ;line and statement in print buffer
3A3A5C            LD      A,(ERRNR)   ;pick up the error code
3C                INC     A           ;step it to give corrected value
12                LD      (DE),A      ;and save the trap code in the p.buffer
3EFF              LD      A,255       ;reset trap code to 0 (minus 1)
323A5C            LD      (ERRNR),A   ;and replace in error code variable
C3761B            JP      MAINL       ;return to BASIC interpreter
          ;       EQUATES
0014      OFFSET  EQU     TRAP-START  ;used to create the actual TRAP address
0027      TRAPA   EQU     PUSH+-START ;gives offset to TRAP addr in inst. PUSH
1303      BTRAP   EQU     4867        ;the normal trap used by BASIC
5C42      NWPPC   EQU     23618       ;the system variable for GOTO line
5C3A      ERRNR   EQU     23610       ;system variable containing report code-1
5B00      PRTBF   EQU     23296       ;start of printer buffer, free is no pr.
0026      GTOHL   EQU     38          ;HI byte of line no 9900.INT(9900/256)
00AC      GTOLL   EQU     172         ;LO byte of line no 9900.9900-256*38
0001      GTOST   EQU     1           ;statement no within line 9900
1B76      MAINL   EQU     7030        ;BASIC interpreter to continur running
5C3D      ERRSP   EQU     23613       ;pointer to stacked error return address
                  END                 ;program length is 004Bh — 75 decimal
```

PAUL BATES

Game for a graph

Impress your teachers with a simple display of any sort of data.

This program will plot graphs from given information in one of several styles and can be reused without reentering the data in each of the plotting modes. It is Menu driven and simplicity itself to use. It can be used to display results from any type of data gathering exercise.

The program itself occupies about 5K and is entirely in BASIC. The Menu has six options:

1) Enter data
2) Choose dot character
3) Choose plot type
4) Draw graph
5) Regression analysis
6) Quit program

OPTIONS

OPTION 1 On selecting this option you will first be asked to enter the number of data pairs. The data pairs are then entered with the screen display the data pair to be entered next. After data entry you will be asked to give:

1) A graph title
2) An X axis title
3) A Y axis title

(Maximum 10 characters)

You will then be asked for a data to define the axes (this could have been done automatically using a bubble sort of data, but I prefer to define my own values to give a neat result). Enter X min, X max, Y min, Y max, as prompted. The screen will then be cleared and you will be asked to define the X and Y tic mark intervals.
For example:
X data ranges from
1 to 7.6
Y data ranges from
0 to 0.5

Xmin = 0 Xmax = 8
X tic mark = 2
Y min = 0 Ymax = 0.5
Y tic mark = 0.1

Beware of setting too small a value for tic mark interval, eg 0.5 for X axis — this would leave the X axis looking rather crowded. After data entry you will be offered a print out of the data pairs entered .

OPTION 2 Gives the choice of.*or+to portray the data points on the graph (see print out).

OPTION 3 Gives you the choice of a point only plot or having the points joined by straight lines (see print out).

OPTION 4 Draws the graph and can be used as many times as necessary after entering data (eg to change options 2 or 3).
(NOTE If options 2 and 3 are not set the default values give a point only print out, using a full stop to denote data position).

You will now be asked if you want a copy after which you will return to the Menu.

OPTION 5 Produces a table giving linear regression equation. This gives the equation of a straight line which best fits the data points entered. (This works over all the data points entered). The regression coefficient r indicates the goodness of fit.

r = 0 data does not follow a linear relationship
r = 1 perfectly fits equation determined

r values above 0.7 can be said to be a good indication of linear behaviour.

OPTION 6 Escapes from program giving a STOP report.

HOW IT RUNS

20-140	*Sets up Menu*
1000-1600	*Enter data option*
2000-2140	*Character option*
3000-3080	*Plot option*
4000-4040	*Draws axes*
4040-4320	*Plots tic marks and numbers axes*
4330-4420	*Plots Y axis title vertically*
4420-4520	*Plots points*
4700-4760	*Routine to join points by straight lines*
4800-4830	*Asks if copy wanted*
5000-5300	*Linear regression*

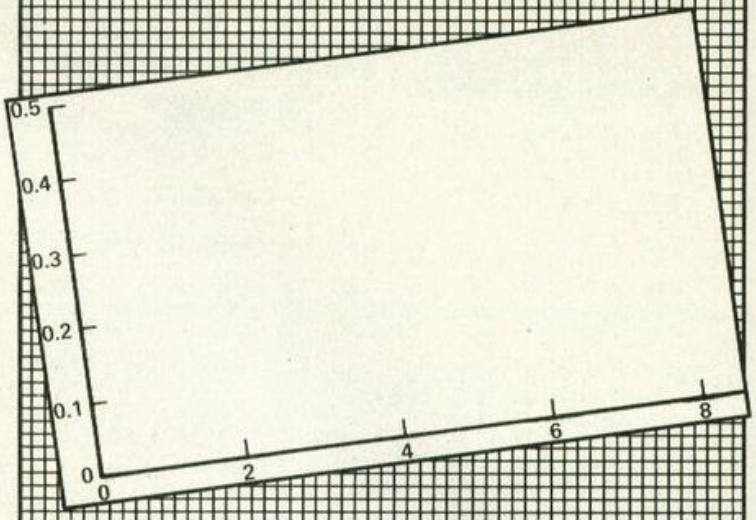
VARIABLES USED

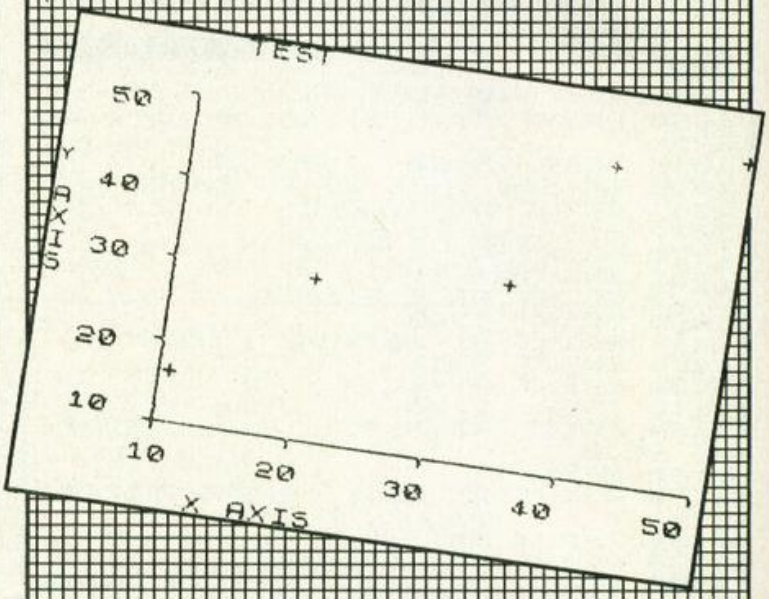
A	*Menu option*
B	*Number of data points*
C	*X min*
D	*X max*
E	*Y min*
F	*Y max*
G	*Dot character value*
H	*Plot choice*
O	*Y values for plotting*
P	*X values for plotting*
Y	*X tic mark interval*
Z	*Y tic mark interval*
A$	*Graph title*
B$	*X axis title*
C$	*Y axis label*
D$	*Copy of data?*
E$	*Copy of graph?*
X(I)	*X value* }
Y(I)	*Y value* } *Data pair*

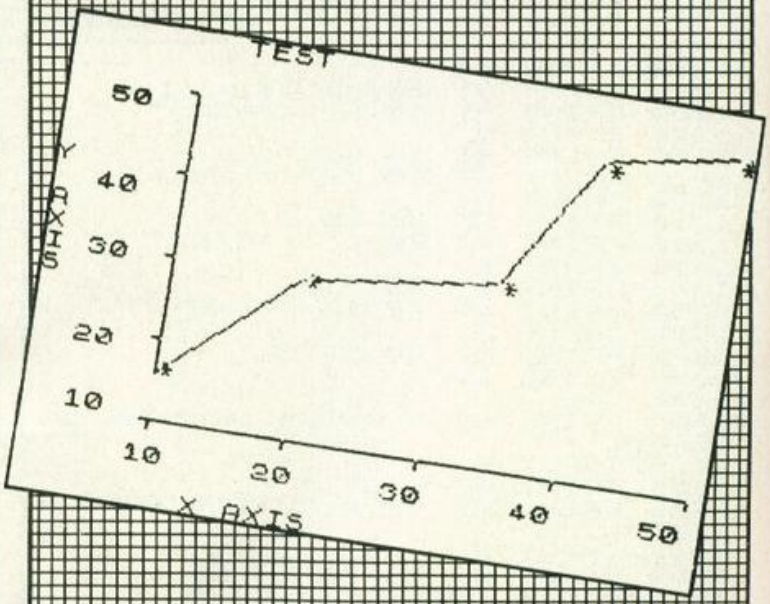
```
  10 BORDER 3: PAPER 6: INK 1: C
LS
  11 POKE 23658,8
  20 PRINT AT 1,3;" GRAPH IT- B
Y PAUL BATES "
  30 PRINT AT 3,11;" MENU "
  40 PRINT AT 5,2;"1-ENTER DATA"
  50 PRINT AT 7,2;"2-CHOOSE DOT
CHARACTER"
  60 PRINT AT 9,2;"3-CHOOSE PLOT
 TYPE"
  70 PRINT AT 11,2;"4-DRAW GRAPH
"
  80 PRINT AT 13,2;"5-REGRESSION
 ANALYSIS"
  90 PRINT AT 15,2;"6-QUIT PROGR
AM"
 100 PRINT AT 18,1;"ENTER OPTION
-0 RETURNS TO MENU"
 110 INPUT Z$
 120 IF Z$;"6" THEN GO TO 110
 125 IF Z$="6" THEN STOP
 130 LET A=VAL Z$
 140 CLS
 150 IF A=0 THEN GO TO 10
 160 GO TO A*1000
1000 BORDER 2: PAPER 1: INK 7: C
LS
1010 DIM B(50): DIM X(50): DIM Y
(50): DIM A$(10): DIM B$(10): DI
M C$(10): DIM D$(1)
1020 PRINT AT 1,5;" 1-DATA ENTR
Y "
1025 LET G=0: LET H=0
1030 PRINT AT 3,2;"NO.OF DATA PA
IRS :"
1040 BEEP .5,20: INPUT B
1045 IF B=0 THEN GO TO 1500
1050 PRINT AT 3,22;B
1060 FOR I=1 TO B
1070 PRINT AT 5,5;"X";I;"="
1080 INPUT X(I)
1090 PRINT AT 5,9;X(I)
1100 BEEP .1,20
1110 PRINT AT 10,5;"Y";I;"="
1120 INPUT Y(I)
1130 PRINT AT 10,9;Y(I)
1140 BEEP .1,20
1150 PRINT AT 5,9;"     ": PRINT
AT 10,9;"     "
1160 NEXT I
1170 PRINT AT 12,1;"GRAPH TITLE
:"
1180 INPUT A$
1190 PRINT AT 12,14;A$
1200 PRINT AT 14,1;"X-AXIS LABEL
:"
1210 INPUT B$
1220 PRINT AT 14,15;B$
1230 PRINT AT 16,1;"Y-AXIS LABEL
:"
1240 INPUT C$
1250 PRINT AT 16,15;C$
1260 PRINT AT 18,1;"X-MIN="
1270 INPUT C
1280 PRINT AT 18,7;C
1290 PRINT AT 18,12;"X-MAX="
1300 INPUT D
1310 PRINT AT 18,18;D
1320 PRINT AT 20,1;"Y-MIN="
1330 INPUT E
1340 PRINT AT 20,7;E
1350 PRINT AT 20,12;"Y-MAX="
1360 INPUT F
1370 PRINT AT 20,18;F
1380 PAUSE 100
1390 CLS
1400 PRINT AT 5,2;"X-TIC MARK IN
TERVAL :"
1410 INPUT Y
1420 PRINT AT 5,25;Y
1430 PRINT AT 10,2;"Y-TIC MARK I
NTERVAL :"
1440 INPUT Z
1450 PRINT AT 10,25;Z
1460 PAUSE 100
1500 CLS : PRINT AT 5,1;"DO YOU
WISH TO ADD OR
1510 PRINT AT 7,1;"THE DATA HAS
BEEN ENTERED ?"
1520 PRINT AT 11,2; INVERSE 1; F
LASH 1;"ENTER Y OR N"
1530 INPUT D$
1540 IF D$<>"Y" THEN GO TO 10
1542 LPRINT TAB 6;"TITLE :";A$
1544 LPRINT
1550 LPRINT TAB 10;"X","
  ";"Y"
1560 LPRINT TAB 9;"---","
  ";"---"
1570 FOR I=1 TO B
1580 LPRINT TAB 9;X(I);"
  ";Y(I)
1590 NEXT I
1600 GO TO 10
2000 BORDER 4: PAPER 6: INK 2: C
LS
2010 REM CHARACTER LETTER IS G
2020 PRINT AT 1,5;" CHARACTER
APE ": LET G=0
2030 PRINT AT 4,2;"OPTION 1: ."
2040 PRINT AT 8,2;"OPTION 2: *"
2050 PRINT AT 12,2;"OPTION 3: +"
2060 PRINT AT 14,1; INVERSE 1; F
LASH 1;" ENTER YOUR CHOICE (1,2,
OR 3) "
2070 INPUT G
2080 IF G=1 THEN LET G=46
2090 IF G=2 THEN LET G=42
2100 IF G=3 THEN LET G=43
2110 PRINT AT 18,2;"YOUR CHOICE
IS"
2120 PRINT AT 18,16;CHR$ G
2130 PAUSE 150
2140 GO TO 10
3000 BORDER 5: PAPER 7: INK 2: C
LS
3010 PRINT AT 1,5;" PLOT CHOICE
": LET H=0
3020 PRINT AT 5,2;"OPTION 1: POI
NTS ONLY"
3030 PRINT AT 10,2;"OPTION 2: PO
INTS JOINED BY"
3040 PRINT AT 11,12;"STRAIGHT LI
NES"
3050 PRINT AT 15,4; INVERSE 1; F
LASH 1;" ENTER YOUR CHOICE "
3060 INPUT H
3070 IF H<1 OR H>2 THEN GO TO 30
50
3080 GO TO 10
4000 IF G=0 THEN LET G=46
4002 IF H=0 THEN LET H=1
4005 BORDER 5: PAPER 7: INK 0: C
LS
4010 LET I=48: LET J=32
4020 PLOT I,J: DRAW 0,120
4030 PLOT I,J: DRAW 200,0
4040 LET K=17
4050 PLOT I,J: DRAW -3,0
4060 LET L=(F-E)/Z
4070 FOR A=1 TO L
4080 LET J=J+INT (120/L)
4090 PLOT I,J: DRAW -3,0
4100 NEXT A
4110 IF F>0 THEN LET L=(F-E)/Z:
IF F<0 THEN LET L=((E-F)/Z)*-1)
4115 LET Q=E
4120 FOR I=1 TO L+1
4130 PRINT AT K,2;Q
4140 LET Q=(Q+Z)
4150 LET K=(K-(17/(L+1)))
4160 NEXT I
4170 LET I=48: LET J=32
4180 LET L2=(D-C)/Y
4190 PLOT I,J: DRAW 0,-3
4200 FOR A=1 TO L2
4210 LET I=I+INT (200/L2)
4220 PLOT I,J: DRAW 0,-3
4230 NEXT A
4240 IF D>0 THEN LET L2=(D-C)/Y:
  IF D<0 THEN LET L2=((D-C)/Y)*-1
```

FEATURE

```
4250 LET K2=5: LET W=C
4260 FOR A=1 TO L2+1
4270 PRINT AT 19,K2;W
4280 LET W=(W+Y)
4290 LET K2=INT K2+(30/(L2+1))
4300 NEXT A
4310 PRINT AT 21,8;B$
4320 PRINT AT 0,8;A$
4330 REM Y AXIS TITLE
4340 LET M=6
4350 FOR I=1 TO 10
4360 PRINT AT M,0;C$(I)
4370 LET M=M+1
4380 NEXT I
4390 LET N=0: LET N=N+C
4400 LET N2=0: LET N2=N2+E
4420 REM  PLOT POINTS
4430 LET P0=32: LET O0=48
4440 FOR A=1 TO B
4450 LET O=((X(A)-N)/(D-N))
4460 LET O=INT ((O*25)+6)
4470 LET P=((Y(A)-N2)/(F-N2))*15
4480 LET P=INT ((P-17)*-1)
4490 PRINT AT P,O;CHR$ 3
4500 IF H=2 THEN GO TO 4700
4520 GO TO 4800
4700 LET Z5=1
4710 LET P1=INT (((P/21)-1)*-175
): LET O1=(O*8)
4720 PLOT O0,P0
4730 DRAW (O1-O0),(P1-P0)
4740 LET P0=P1: LET O0=O1
4750 LET Z5=Z5+1: IF Z5=8 THEN G
O TO 4800
4760 GO TO 4510
4800 DIM E$(1): INPUT "COPY (Y/N)
?";E$
4810 IF E$="N" THEN GO TO 10
4820 IF E$="Y" THEN COPY
4830 GO TO 10
5000 BORDER 6: PAPER 4: INK 0: C
LS
5010 PRINT AT 1,6;"LINEAR REGRES
SION"
5020 PRINT AT 4,2;"TEST NAME IS
:";A$
5030 LET D3=0: LET E3=0: LET F3=
0: LET G3=0: LET H3=0
5040 FOR I=1 TO B
5070 LET D3=D3+X(I)
5080 LET E3=E3+Y(I)
5090 LET F3=F3+(X(I)*X(I))
5100 LET G3=G3+(Y(I)*Y(I))
5110 LET H3=H3+(X(I)*Y(I))
5120 NEXT I
5130 LET L3=H3-((D3*E3)/B)
5140 LET M3=F3-((D3*D3)/B)
5150 LET N3=L3/M3
5160 LET O3=D3/B
5170 LET P3=E3/B
5180 LET Q3=SQR (M3*(G3-((E3*E3)
/B)))
5190 LET R3=L3/Q3
5200 LET S3=P3-(N3*O3)
5210 PRINT AT 6,2;"THE SLOPE IS
:";N3
5220 PRINT AT 8,2;"THE INTERCEPT
 IS :";S3
5230 PRINT AT 10,2;"HENCE THE EQ
UATION IS )"
5240 PRINT AT 12,5;"Y=";N3;"* X
 + ";S3
5250 PRINT AT 14,1;"THE CORRELAT
ION COEFFICIENT IS :";R3
5260 PRINT AT 18,2;"COPY (Y/N)
?"
5270 DIM J$(1): INPUT J$
5280 IF J$="Y" THEN COPY
5290 IF J$<>"Y" THEN GO TO 10
5300 GO TO 10
```





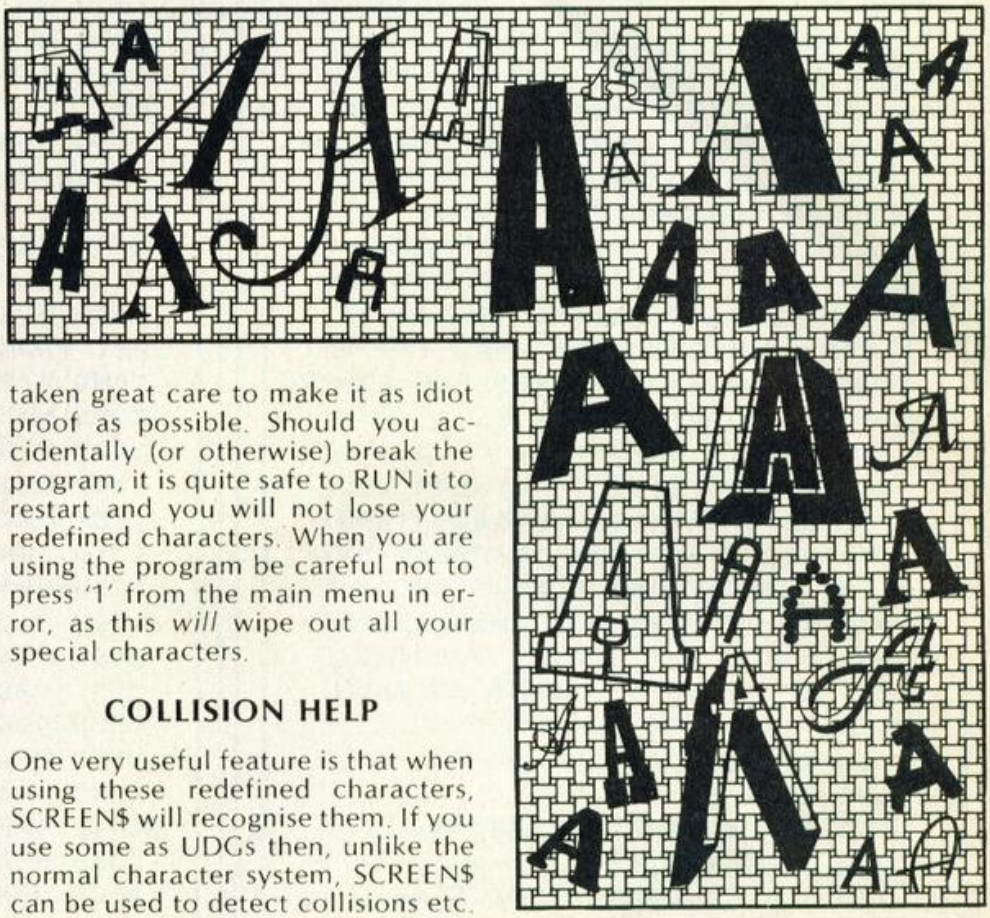


All Change

No it's not time to swop your Spectrum for something else. Merely a change of characters.

There are many occasions when an alternative set of characters is a very useful option — professional titles and Gothic style lettering for home programmed adventure games to name but two!

This program will allow you to redefine the entire character set and User Defined Graphics and to look at any RAM or ROM character at eight times the normal size. A simple machine code routine is used from within the program to copy the ROM character set into RAM. Notice that the program takes into account the memory size and makes appropriate RAMTOP changes with the CLEAR command.



TAKING CARE

Paul has made use of REM statements to explain the various sections of the program, you may leave these out of course, especially if working with 16K, but they are valuable when trying to debug any typing errors.

This is a good example of careful programming (although no doubt someone will find a way to crash it if they try) and Paul has taken great care to make it as idiot proof as possible. Should you accidentally (or otherwise) break the program, it is quite safe to RUN it to restart and you will not lose your redefined characters. When you are using the program be careful not to press '1' from the main menu in error, as this *will* wipe out all your special characters.

COLLISION HELP

One very useful feature is that when using these redefined characters, SCREEN$ will recognise them. If you use some as UDGs then, unlike the normal character system, SCREEN$ can be used to detect collisions etc.

VARIABLES USED

r,q	*Start of RAM character set.*
i$	*INKEY$ check.*
q$	*Titles.*
a$	*INPUT check.*
x,y,z	*Used in enlarging characters.*
a	*Value of input, etc.*
w,c	*Data for drawing/inputing characters.*
f,b,g	*FOR-NEXT variables.*

```
  1 REM   Character Change.
        @ 1983 Paul Matthews.
 10 LET q= PEEK 23675+256* PEEK
23676: LET q=q-801
 20 POKE 23609,20
 40 CLEAR q
 50 LET q= PEEK 23730+256* PEEK
23731: LET q=q+31
 60 GO SUB 9000: GO SUB 9200
 70 GO SUB 140: IF  PEEK 23681
<> 1 THEN  LET i$="1": GO TO 520
 80 GO TO 400
 90 STOP
100 POKE 23606,0: POKE 23607,60
 110 RETURN
 140 LET r=q/256: POKE 23606,(r-
 INT r)*256: POKE 23607, INT r-1
 150 RETURN
 390 REM #### MENU ####
 400 BORDER 7: PAPER 7: INK 2: C
LS : GO SUB 100
 410 PRINT  BRIGHT 1; INVERSE 1;
 INK 1; TAB 7;"CHARACTER CHANGE"
; TAB 31;" "; TAB 5;"@ 1983 Paul
 Matthews"; TAB 31;" "
 420 PRINT ' TAB 14; INK 1;"MENU
"
 430 PRINT ' TAB 10; INK 3;"CHAR
ACTER SET"
 440 PRINT '"1 Reset whole chara
cter set."'"2 View a ROM charact
er."'"3 View a RAM character."
 450 PRINT "4 Change a character
."'"5 Reset just one character."
 460 PRINT ' INK 3; TAB 5;"USER
DEFINABLE GRAPHICS"
 470 PRINT '"6 View a UDG."'"7 C
hange a UDG."
```

```
 480 PRINT  INK 3;'"8 A Complete
 view."'"9 LOAD/SAVE Routines."
 490 INK 0: PRINT #0; BRIGHT 1;
INK 7; PAPER 4; TAB 5;"Input you
r choice now..."; TAB 31;" "
 500 LET i$= INKEY$ : IF i$< CHR
$ 49 OR i$> CHR$ 57 THEN  GO TO
500
 520 BEEP .1, CODE i$-64
 530 IF i$="1" THEN  LET a= USR
(q-30): POKE 23681,1: GO TO 400
 540 IF i$="2" OR i$>"5" THEN  G
O SUB 100: GO TO 560
 550 GO SUB 140
 560 CLS : LET q$=("CHARACTER SE
T" AND i$<"6")+("USER DEFINABLE
GRAPHICS" AND (i$="6" OR i$="7")
)+("A COMPLETE VIEW" AND i$="8")
+("LOAD/SAVE ROUTINES" AND i$="9
")
 570 PRINT  INK 3; AT 0,15-( LEN
 q$/2);q$''
 580 GO SUB (( VAL i$)*200+600)
 600 PRINT #0; INK 1;"Press ENTE
R for Menu, else same."
 620 IF  INKEY$ ="" THEN  GO TO
620
 630 IF  INKEY$ = CHR$ 13 THEN
BEEP .1,12: GO TO 400
 650 GO TO 520
 690 STOP
 700 REM ###DRAW A BOX###
 710 PLOT 128,128
 720 INK 1: DRAW 64,0: DRAW 0,-6
4: DRAW -64,0: DRAW 0,64
 740 INK 0: RETURN
 800 REM ###ENLARGE CHARACTOR###
 810 LET z=w: LET y=128
 820 INK 1: PRINT  TAB 16;
 830 LET x= INT (z/y): PRINT  CH
R$ 164;: IF x=1 THEN  PRINT  CHR
$ 8;"■";: LET z=z-y
 850 LET y=y/2: IF y<1 THEN  INK
 0: BRIGHT 0: RETURN
 860 GO TO 830
1000 REM ##LOOK AT CHARACTERS##
1010 INPUT "View Which ROM Chara
cter? ";a$: LET a= CODE a$: IF a
<32 OR a>127 OR  LEN a$ <> 1 THE
N  GO TO 1010
1020 LET a=(a-32)*8+15616
1030 GO SUB 700
1040 FOR f=1 TO 16: PRINT a$;" "
;: NEXT f
1060 PRINT '''Code:"
1070 FOR f=a TO a+7: LET w= PEEK
 f: PRINT 'f;"=";w;: GO SUB 800:
 NEXT f
1080 PRINT '': FOR f=1 TO 16: PR
INT a$;" ";: NEXT f
1090 RETURN
1200 REM ##AS 1000 BUT IN RAM##
1210 INPUT "View Which RAM Chara
cter? ";a$: LET a= CODE a$: IF a
<32 OR a>127 OR  LEN a$ <> 1 THE
N  GO TO 1210
1220 LET a=(a-32)*8+q
1230 GO TO 1030
1400 REM ###CHANGE CHARACTER###
1420 INPUT "Character to be chan
ged...";a$
1430 LET a= CODE a$: IF a>127 OR
 a<32 OR  LEN a$>1 THEN  GO TO 1
420
1440 PLOT 128,152: GO SUB 720: L
ET c=(a-32)*8+q
1460 FOR f=c TO c+7: LET w= PEEK
 f: PRINT 'w;: PRINT  TAB 10;" "
 AND f/2= INT (f/2);a$;: GO SUB
800: NEXT f
1480 PLOT 128,80: GO SUB 720
1500 PRINT  INK 3;'"New code:";:
 FOR f=0 TO 7
1510 INK 2: INPUT w: IF w<0 OR w
>255 OR w <>  INT w THEN  GO TO
1510
1520 POKE c+f,w: PRINT 'w; TAB 1
0;" " AND f/2= INT (f/2);a$;: GO
 SUB 800
1530 NEXT f
1540 PRINT '': FOR f=1 TO 16: PR
INT  CHR$ a;" ";: NEXT f
1550 RETURN
1600 REM ##RESET A CHARACTER##
1620 INPUT "Character to be rese
t...";a$
1630 LET a= CODE a$: IF a>127 OR
 a<32 OR  LEN a$>1 THEN  GO TO 1
620
1640 PLOT 128,152: GO SUB 720: L
ET c=(a-32)*8+q: LET a=(a-32)*8+
15616
1650 FOR b=1 TO 2
1660 FOR f=0 TO 7: PRINT ';: LET
 w= PEEK (c+f): IF f=0 OR f=7 TH
EN  PRINT  TAB 0;: FOR g=1 TO 5:
 PRINT a$;" ";: NEXT g
1670 GO SUB 800: NEXT f
1680 IF b=1 THEN  FOR g=0 TO 7:
POKE (g+c), PEEK (g+a): NEXT g:
PRINT  AT 7,2;"WAS:"; AT 10,0: P
LOT 128,80: GO SUB 720
1690 NEXT b: PRINT  AT 16,2;"NOW
:"
1700 RETURN
1800 REM ###LOOK AT UDG###
1810 INPUT "UDG to be viewed..."
;a$: LET a= CODE a$: LET a=a+(79
 AND a>64 AND a<91)+(47 AND a>96
 AND a<123): IF a<144 OR a>164 O
```

```
R  LEN a$ <> 1 THEN  GO TO 1810
1820 GO SUB 700: FOR f=1 TO 16:
PRINT  CHR$ a;" ";: NEXT f
1830 PRINT ''"UDG Code:"
1840 LET c= USR a$
1850 FOR f=0 TO 7: LET w= PEEK (
f+c): PRINT 'w;: GO SUB 800
1860 NEXT f: PRINT ''
1870 FOR f=1 TO 16: PRINT  CHR$
a;" ";: NEXT f
1880 RETURN
2000 REM ###CHANGE UDG###
2010 INPUT "UDG to be changed.."
;a$: LET a= CODE a$: LET a=a+(79
 AND a>64 AND a<91)+(47 AND a>96
 AND a<123): IF a<144 OR a>163 T
HEN  GO TO 2010
2020 LET c= USR a$(1): PLOT 128,
152: GO SUB 720: FOR f=0 TO 7
2030 LET w= PEEK (f+c): PRINT 'w
; TAB 10;" " AND f/2= INT (f/2);
 CHR$ a;: GO SUB 800: NEXT f
2040 PRINT  INK 2;'"New UDG Code
:";
2050 PLOT 128,80: GO SUB 720
2060 FOR f=0 TO 7: INK 2: INPUT
w: IF w<0 OR w>255 OR w <>  INT
w THEN  GO TO 2060
2070 POKE (f+c),w: PRINT 'w; TAB
 10;" " AND f/2= INT (f/2); CHR$
 a;: GO SUB 800: NEXT f
2080 PRINT '': FOR f=1 TO 16: PR
INT  CHR$ a;" ";: NEXT f
2090 RETURN
2200 REM ###A COMPLETE VIEW###.
2210 PRINT "ROM CHARACTER SET:";
2220 INK 0: FOR g=1 TO 2
2240 FOR b=1 TO 6: PRINT : FOR f
=0 TO 14: PRINT " " AND (b/2= IN
T (b/2) AND f=0); CHR$ ((b*15)+f
+17);" ";: NEXT f: NEXT b
2250 PRINT  TAB 10;: FOR f=122 T
O 127: PRINT  CHR$ f;" ";: NEXT
f
2260 IF g=1 THEN  INK 2: PRINT '
"RAM CHARACTER SET:";: GO SUB 14
0
2270 NEXT g
2280 GO SUB 100
2290 INK 3: PRINT '"USER DEFINAB
LE GRAPTICS:": FOR f=144 TO 164
2300 PRINT  CHR$ f;" "; CHR$ 23+
 CHR$ 11+ CHR$ 0 AND f=159;
2310 NEXT f
2330 RETURN
2400 REM ##LOAD/SAVE ROUTINES##
2410 PRINT  INK 1;'"A..Save New
Character set."''"B..Save User D
efinable Graphics."''"C..Save Th
is Program."
2430 PRINT  INK 2;''"D..Load a C
haracter Set."''"E..Load User De
finable Graphics."''"F..Load Any
 Program."
2450 LET a$= INKEY$ : LET a= COD
E a$: LET a=a-(32 AND a>96 AND a
<103): IF a<65 OR a>70 THEN  GO
TO 2450
2460 LET b=3+(1 AND a>67)+((a-65
)*2): PRINT  AT b,0; INK 8; FLAS
H 1; OVER 1; TAB 7
2470 GO SUB (3000+((a-65)*100))
2480 IF  INKEY$ ="n" OR  INKEY$
="N" THEN  GO TO 2480
2490 RETURN
3000 SAVE "Char. Set" CODE q,768
3020 PRINT  AT 20,10;"VERIFY ? (
Y/N)": LET a$= INKEY$
3030 IF a$="n" THEN  GO TO 3080
3040 IF a$ <> "y" THEN  GO TO 30
20
3050 PRINT  AT 20,2;"Rewind Tape
 and Press Play."; AT 19,0: IF a
 <> 67 THEN  VERIFY "" CODE
3060 IF a=67 THEN  VERIFY ""
3080 RETURN
3100 SAVE "U.D.G.'s" CODE (q+769
),168
3120 GO TO 3020
3200 LET a$="y": SAVE "Character
." LINE 1
3220 GO TO 3020
3300 PRINT  AT 18,1;"Connect ear
 plug & start tape.";
3310 LOAD "" CODE q
3330 RETURN
3400 PRINT  AT 18,1;"Connect ear
 plug & start tape.";
3410 LOAD "" CODE (q+769)
3430 RETURN
3500 PRINT  AT 18,12; FLASH 1;"B
YE BYE."
3510 PRINT #1;"Press any key & t
hen start tape."
3520 LOAD "": RUN
8999 STOP
9000 REM ###SET UP UDG's###
9010 RESTORE 9000: FOR f=0 TO 7:
 READ a: POKE  USR "u"+f,a
9020 NEXT f: RETURN
9030 DATA 128,128,128,128,128,12
8,128,255
9200 RESTORE 9200: LET r=q/256
9210 FOR f=(q-30) TO (q-30)+11:
READ a: POKE f,a: NEXT f: RETURN

9220 DATA 17,(r- INT r)*256, INT
 r,33,0,61,1,0,3,237,176,201
9990 STOP
9999 GO SUB 100: POKE 23609,0
```

S. J. PATRICK

Thinchars

Put the squeeze on your screen with this slimmer character utility.

Thinchars is a machine code routine written for the Spectrum which displays 42 characters on each line of the screen, instead of the usual 32. Each character is 6 pixels wide instead of 8, including a blank column between letters. The shapes of the 96 characters (codes 32 to 127) are stored in a table at the end of the routine. Each character takes 5 bytes, one per column, which are read from left to right; the sixth column is always blank and is not stored in the table. The routine is *relocatable*, which means that it can be loaded at any address in memory and will work without modification. This makes use of the fact that when a "USR n" statement is executed in BASIC. the bc register is loaded with n, the start address of the routine called.

VARIABLES

Thinchars takes the text to be printed from the 1-dimensional character array s$, which must be set up by a DIM s$(. . .) statement at the start of the program to ensure that it is the first variable in the variables area. (An ordinary string variable would be moved to the end of the variables area each time its value was altered.) If the first variable is not array s$, the routine returns without doing any printing. The first two characters of s$ are not printed, but their codes are used as the x and y co-ordinates of where the printing is to begin; the x coordinate is the same as that used for PLOTting, but the value runs from 0 for the top row of the screen to 184 for the bottom row (to allow printing on all 24 rows), and is rounded off if necessary to a multiple of 8. The end of the string to be printed is signaled by a character with code 128. So a general statement to set up the string would be LET s$=CHR$ x+CHR$ y+ "text"+CHR$ 128. There is no limit to the length of s$; if the printing reaches the foot of the screen, it continues at the top.

COLOUR

Any of the colour items (PAPER, INK, INVERSE, OVER, etc) can be set before THINCHARS is called. Two ROM routines are used: call 3405 copies the permanent screen attributes to the temporary ones; and call 3035 uses the screen address in the hi register to set the attributes in the appropriate square. (This is done at the left hand and right-hand edges of each character, as they may be in different attribute squares.) The routine will fill the screen with 1008 characters in about 1.4 seconds.

LOADING THE ROUTINE

The routine is 199 bytes long, followed by a character table of 480 bytes; suitable start addresses at 64600 for a 48K Spectrum and 31900 for a 16K Spectrum, but any address may be used, and when the routine has been saved (SAVE "thinchars" CODE 64600,679) it may be reloaded to a different address. Remember to enter CLEARN n, where n is less than the chosen start address, before loading the routine. The following BASIC program may be used for loading: (addresses are in decimal, bytes in hex; enter x to go back one byte, s to stop).

LOADING PROGRAM

```
10  DEF FN h (p$)=CODE p$ — 48 —
    7*(p$>"9")-32*(p$="a")
20  DIM a$ (2)
30  INPUT "Start address:";a
40  INPUT (a), LINE a$
50  IF a$(1)="s" THEN STOP
60  IF a$(1)="x" THEN LET a=a-1: GO TO 40
70  POKE a, 16*FN h(a$(1))+FN h(a$(2))
80  PRINT a; TAB 6; a$
90  LET a=a+1: GO TO 40
```

LOADING THE CHARACTER TABLE

Figure 1 shows the 480 bytes of the character table, and the characters which they represent. They are loaded into memory immediately following the routine. Alternatively, the characters may be generated by means of the program THIN.GEN. In this, you are given a 5×8 grid in which to create a character, which is then stored in the appropriate place; the program displays all the characters so far loaded, and any of them may be returned to the grid for modification.

HOW IT RUNS

10	Ensures that s$ is the first variable
130-160	Obtains start address of THINCHARS; checks first character and halts if it is incorrect
200-260	Sets up a copy of the character set in s$; the BEEP in line 250 is to convince the user that something is happening
270-290	Print box for character, character set and instructions
300	Set cursor to top left-hand corner
310	Print cursor
320-330	Wait for key; remove cursor
340	Plot or unplot a square in the grid
370-390	Move cursor
400-496	Load grid with character i$. Line 460 tests if the bit read from the character table is the same colour as the relevant square on the grid, and if not, changes it. POINT was used instead of SCREEN$ as the latter treats both white and black squares as spaces.
500-595	Store grid in character table in place determined by i$, and re-print the character set
600-620	Prints the character set (on yellow PAPER, so that the user can see how many empty spaces are left).

MACHINE CODE LISTING

The addresses and the numbers in the assembler listing are in decimal, and the addresses are relative to the start of the program.

```
000 C5          push bc          save routine start address
001 DDE1        pop ix             in ix
003 FDCB0286    res 0, TV-FLAG   copy permanent attributes
007 CD4D0D      call 3405          to temporary attributes
010 012700      id bc, 39        set ix to base address of character
                                   table
013 DD09        add ix, bc
015 2A4B5C      ld hl,(VARS)     find first BASIC variable
018 7E          ld a,(hl)
019 FED3        cp 211           return it not array s$
021 C0          ret nz
022 010600      ld bc,6          jump over dimension
025 09          add hl,bc          information
026 5E          ld e,(hl)        load e and d with x and y
027 23          inc hl             co-ordinates of start
028 56          ld d,(h)           position
029 23          inc hl
030 7B          ld a,e           create mask byte showing
031 E607        and 7              which bit of display
033 47          ld b,a             bytes is to be altered,
034 04          inc b              and store in register c
035 3E01        ld a,1
037 0F          rrca
038 10FD        djnz 037
040 4F          ld c,a
041 CB3B        srl e            convert co-ordinates to
043 CB3B        srl e              address in display area
045 CB3B        srl e
047 7A          la a,d
048 FEC0        cp 192           (return if y co-ordinate is
050 D0          ret nc             off foot of screen)
051 17          rla
052 17          rla
053 E6E0        and 224
055 B3          or e
056 5F          ld e,a
057 7A          ld a,d
058 1F          rra
059 1F          rra
060 1F          rra
061 E618        and 24
063 F640        or 64
065 57          ld, d,a
066 EB          ex de,hl
067 1A          ld a,(de)        load character to be
                                   printed
068 13          inc de           increment pointer
069 FE80        cp 128           return if character had
071 C8          ret z              code of 128
072 D5          push de
073 C5          push bc          calculates address in character
074 EB          ex de,hl           table of first column of
075 DDE5        push ix            character to be printed
077 E1          pop hl             (ix+5*ad)
078 4F          ld c,a
079 0600        ld b,0
081 09          add hl, bc
082 09          add hl, bc
083 09          add hl, bc
084 09          add hl, bc
085 09          add hl, bc
086 EB          ex de, hl
087 C1          pop bc
088 0606        ld b, 6          load b with no. of columns
```

```
00 00 00 00 00 00 00 7A
00 00 00 70 00 70 00 24
7E 24 7E 24 12 2A 7F 2A
24 32 34 08 16 26 2C 52
5A 24 0A 00 10 60 00 00
00 00 3C 42 00 00 42 3C
00 00 10 54 38 54 10 08
08 3E 08 08 00 01 06 00
00 08 08 08 08 08 00 06
06 00 00 02 04 08 10 20
3C 46 5A 62 3C 00 22 7E
02 00 26 4A 4A 4A 32 44
52 52 52 2C 0C 14 24 7E
04 72 52 52 52 4C 3C 52
52 52 0C 40 42 44 48 70
2C 52 52 52 2C 30 48 48
48 3E 00 00 24 00 00 00
02 2C 00 00 00 08 14 22
00 14 14 14 14 14 00 22
14 08 00 30 40 4A 48 30
3E 41 4C 52 3E 3E 48 48
48 3E 7E 52 52 52 2C 3C
42 42 42 24 7E 42 42 24
18 7E 52 52 52 42 7E 50
50 50 40 3C 42 42 4A 2C
7E 10 10 10 7E 00 42 7E
42 00 04 02 02 02 7C 7E
10 18 24 42 7E 02 02 02
02 7E 20 18 20 7E 7E 20
18 04 7E 3C 42 42 42 3C
7E 48 48 48 30 3C 42 4A
44 3A 7E 48 48 4C 32 22
52 52 52 4C 40 40 7E 40
40 7C 02 02 02 7C 78 04
02 04 78 7C 02 1C 02 7C
46 28 10 28 46 60 10 0E
10 60 46 4A 52 52 62 00
00 7E 42 00 20 10 08 04
02 00 42 7E 00 00 10 20
7E 20 10 01 01 01 01 01
12 3E 52 52 42 04 2A 2A
2A 1E 7E 12 12 12 0C 1C
22 22 22 12 0C 12 12 12
7E 1C 2A 2A 2A 1A 00 3E
50 50 00 18 25 25 25 3E
7E 10 10 10 0E 00 12 5E
02 00 02 01 01 5E 00 7E
08 14 22 00 00 7C 02 02
00 3E 20 1E 20 1E 3E 20
20 20 1E 1C 22 22 22 1C
3F 24 24 24 18 18 24 24
24 1F 1E 20 20 20 10 12
2A 2A 2A 24 20 7C 22 22
00 3C 02 02 02 3C 38 04
02 04 38 3C 02 1C 02 3C
22 14 08 14 22 39 05 05
05 3E 22 26 2A 32 22 00
10 2C 42 00 00 00 7E 00
00 00 42 2C 10 00 40 80
40 20 40 3E 49 55 41 3E
00 00 00 00 00 00 00 00
```

```
090 E5          push hl         set attributes on screen where
091 D5          push de           first column of character is to
092 CDDB0B      call 3035         be printed
095 D1          pop de
096 E1          pop hl
097 1A          ld a,(de)       load character pattern
098 D5          push de
099 05          dec b           if last column then set attributes
100 2006        jr nz, 108        on screen
102 E5          push hl
103 CDDB0B      call 3035       and clear accumulator to print
106 E1          pop hl            blank column between
107 AF          xor a             characters
108 04          inc b
109 FDCB5756    bit 2, P-FLAG   if INVERSE is set then invert
113 2801        jr z, 116         character pattern
115 2F          cpl
116 1E08        ld e,8          number of rows
118 57          ld d,a          store pattern in d
119 FDCB5746    bit 0,P-FLAG    OVER?
123 280D        jr z, 138
125 CB12        rl d            (OVER=1) if pattern bit is set then
127 3003        jr nc, 132        flip the bit in the display
129 7E          ld a,(hl)
130 A9          xor c
131 77          ld (hl),a
132 24          inc h           move down one row
133 1D          dec e           go back for next row
134 20F5        jr nz, 125
136 1810        jr 154
138 CB12        rl d            (OVER=0) make the bit in the
140 7E          ld a,(hl)       display equal to the pattern bit
141 3003        jr nc, 146
143 B1          or c
144 1803        jr 149
146 2F          cpl
147 B1          or c
148 2F          cpl
149 77          ld (hl),a
150 24          inc h           move down one row
151 1D          dec e           go back for next row
152 20F0        jr nz, 138
154 CB09        rrc c           set mask and address for next
156 3001        jr nc, 159        column
158 2C          inc l
159 7C          ld a,h          move back up 8 rows
160 D608        sub 8
162 67          ld h,a
163 D1          pop de          increment pointer to character
164 13          inc de            table
165 10BA        djnz 097        next column
167 D1          pop de
168 3E1F        ld a, 31        if x co-ordinate 248...
170 A5          and l
171 FE1F        cp 31
173 2094        jr nz,067       (next character)
175 79          ld a,c
176 FE40        cp 64
178 308F        jr nc,067       (next character)
180 0E80        ld c, 128       ...then start a new row of
182 2C          inc l             characters
183 208A        jr nz, 067      if a new third of the screen then
185 7C          ld a,h            update screen address
186 C608        add a,8
188 67          ld h,a
189 E618        and 24          if off foot of screen then go to
191 EE18        xor 24            top of screen
193 2080        jr nz, 067      (next character)
195 2640        ld h,64         top of screen address
197 18EB        jr 178          (next character — jumps via 178
                                  as 067 is out of range)
```

```
 100 REM  THIN.GEN @ S J Patrick
 110 DIM s$(99)
 120 PRINT "     THIN CHARACTER G
ENERATOR"
 130 INPUT "Start address of THI
NCHARS       routine?  ";base
 140 IF  PEEK base=197 THEN  GO
TO     200
 150 PRINT "THINCHARS not loaded
 at ";base
 160 STOP
 200 REM ----------------load s$
 210 LET s$(1)= CHR$ 0
 220 LET s$(2)= CHR$ 8
 230 FOR i=32 TO 128
 240   LET s$(i-29)= CHR$ i
 250   BEEP .002,0
 260 NEXT i
 270 GO SUB 600
 280 PLOT 103,136: DRAW 41,0:
      DRAW 0,-65: DRAW -41,0:
       DRAW 0,65
 290 PRINT  AT 15,0;"Press 5,6,7
,8 to move cursor,          z to
plot/unplot a square,       s to
store the grid,             l to
load the grid."
 300 LET x=0: LET y=0
 310 PRINT  OVER 1; AT 5+y,13+x;
       "X"
 320 PAUSE 0: LET i$= INKEY$
 330 PRINT  OVER 1; AT 5+y,13+x;
       "X"
 340 IF i$="z" THEN  PRINT  AT 5
+y     ,13+x; OVER 1;"■": GO TO
       310
 350 IF i$="l" THEN  GO TO 400
 360 IF i$="s" THEN  GO TO 500
 370 LET x=x-(i$="5" AND x>0)
     +(i$="8" AND x<4)
 380 LET y=y-(i$="7" AND y>0)
     +(i$="6" AND y<7)
 390 GO TO 310
 400 REM --------------load grid
 410 INPUT "Character to load (E
NTER to     return):"; LINE i$
 420 IF i$="" THEN  GO TO 310
 430 FOR x=0 TO 4
 440   LET a= PEEK (base+39+5* C
ODE       i$+x)
 450   FOR y=0 TO 7
 460    IF (a>127) <>  POINT (10
5+8*        x,131-8*y) THEN  PRIN
T             OVER 1; AT 5+y,13+x
;"■"
 470    IF a>127 THEN  LET a=a-1
28
 480    LET a=2*a
 490   NEXT y
 493 NEXT x
 496 GO TO 300
 500 REM -------------store grid
 510 INPUT "Character to store (
ENTER to     return):"; LINE i$
 515 IF  CODE i$<32 THEN  GO TO
       310
 520 FOR x=0 TO 4
 530   LET a=0: LET b=128
 540   FOR y=0 TO 7
 550    IF  POINT (105+8*x,131-8
*        y) THEN  LET a=a+b
 560    LET b=b/2
 570   NEXT y
 580   POKE base+39+5* CODE i$+x
,a
 590 NEXT x
 595 GO SUB 600: GO TO 300
 600 REM -------print characters
 610 PAPER 6: LET 1= USR base
 620 PAPER 7: RETURN
```

ALAN RENWICK

Guitar Tutor

We can't promise a Top Ten hit, but we can demonstrate chords and help you tune up.

Written for the 48K Spectrum, this is a must for any guitarist, containing routines for expert and beginner alike, and is also a valuable educational aid. The program uses approximately 13.5K but 16K users could break the program into smaller units and record them as sequential programs.

CHORD CHECK

The program is designed to go one step beyond guitar tutor books by making use of the sound capabilities of the Spectrum to aid tuning and to check that you are playing chords correctly. There are limits to the ability of the Spectrum of course, one of which being the fact that only one note at a time can be played but Alan uses what is available to the full.

The program is structured in a simple manner which makes for easy expansion or contraction of the routines. Each particular subject is coded at a block of lines which is a multiple of 1000. This is selected from line 595.

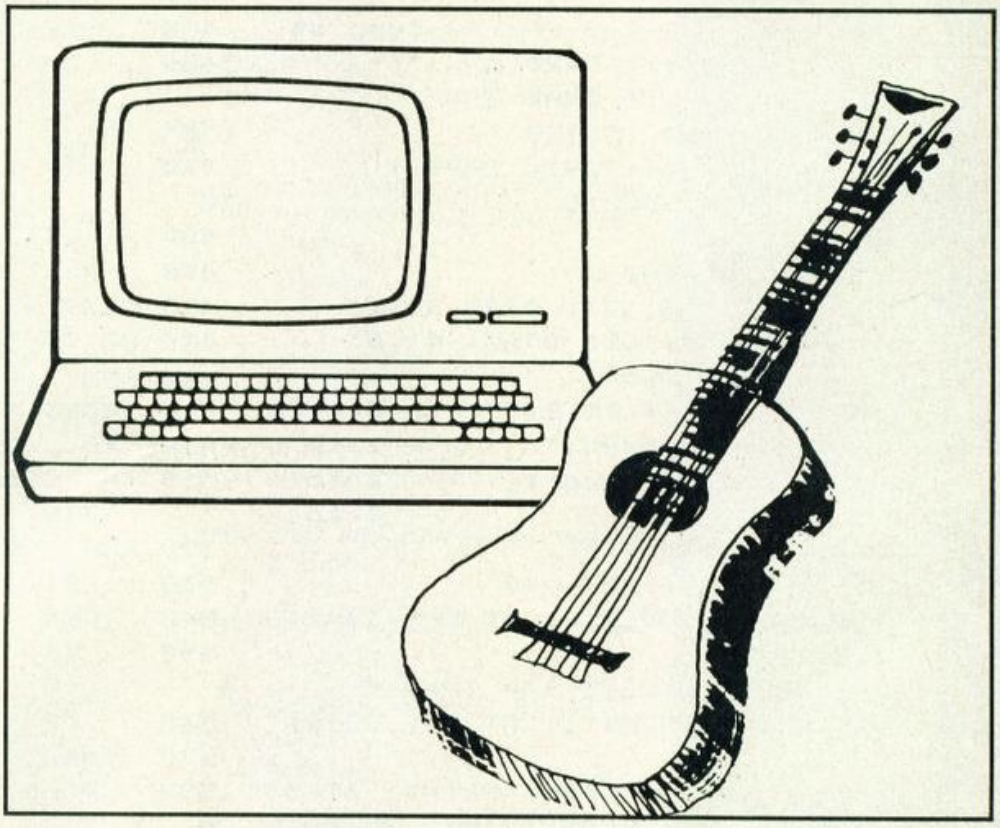
If you renumbered lines 6000-6030 to 5960-5999 you would have room for three routines of your own at 6,7 and 8000. The possibilities are endless — how about lyrics, song structure and rhythm patterns?

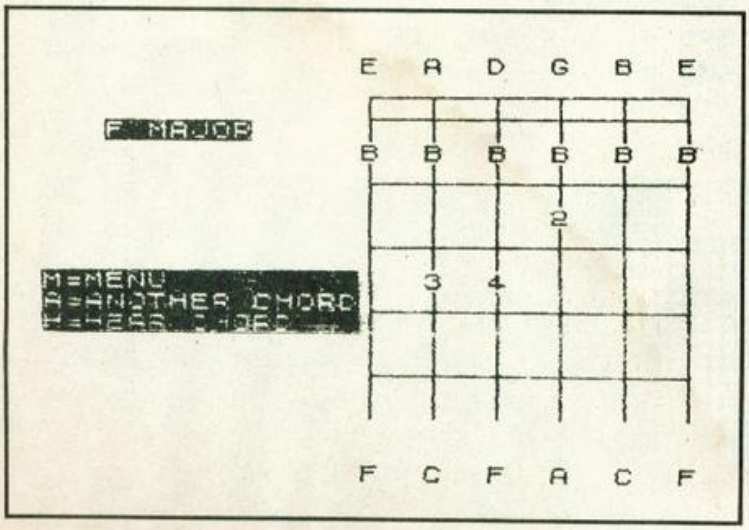
CARE WITH DATA

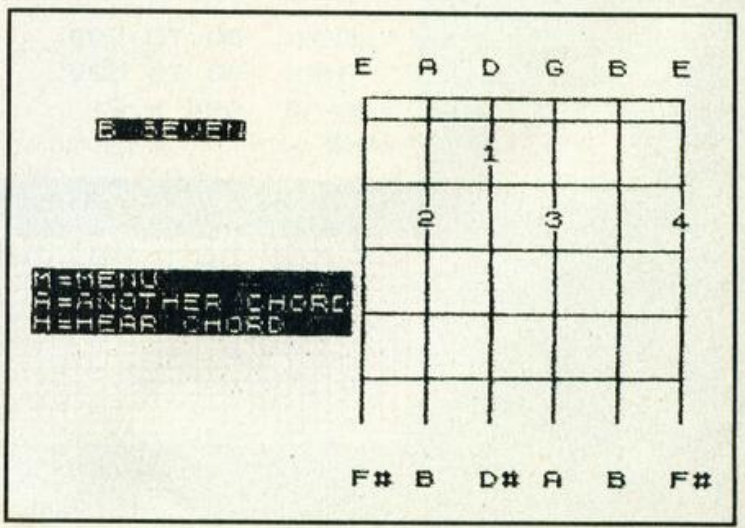
The whole program is built around data statements and it is important that these are typed in accurately — especially lines 9210 to 9420. If the correct number of spaces are not left the program will crash. Note that the flat character is the UDG character A, but in the listing I replaced it by "b" because my printer would print it as "A" and Ab is clearer then AA!

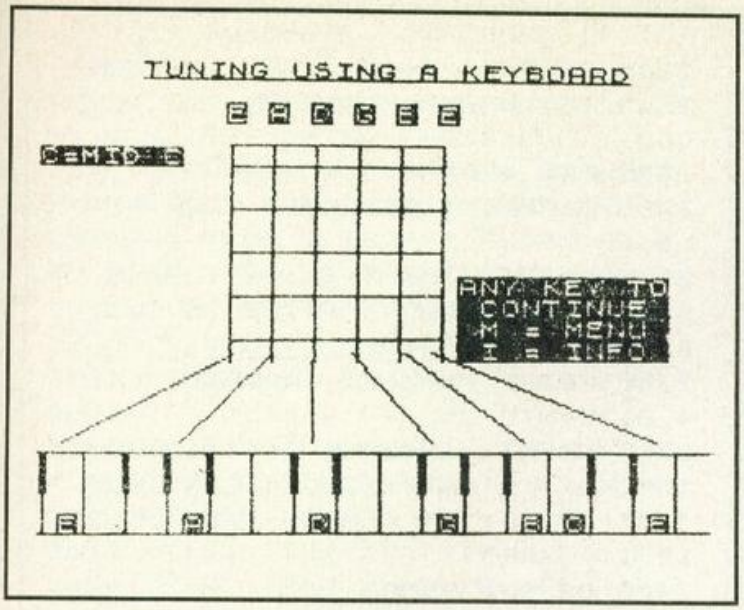
Twenty chords are included but more can be added if you wish by increasing the 20 in line 9210 DIM. and adding the extra chords to the DATA statements. These are constructed as follows:

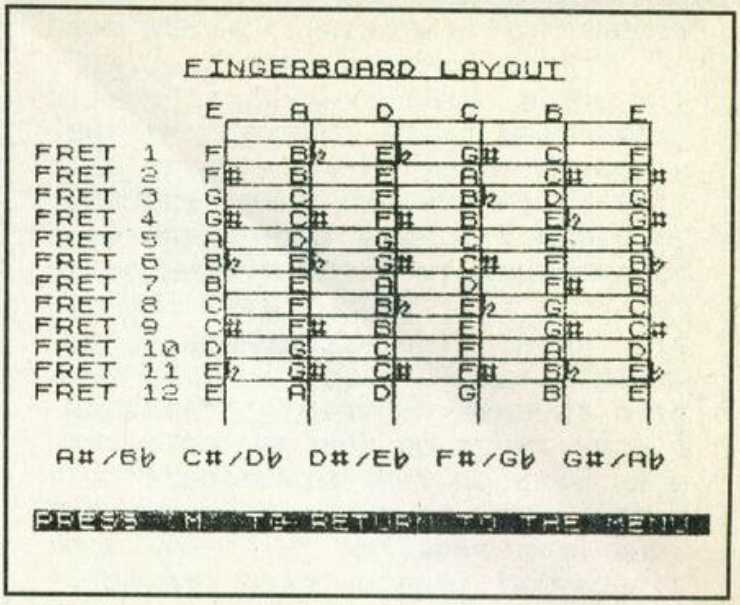
C$(x,1 to 12)	*Notes which appear under the chord window.*
13 to 16	*Print position for finger 1.*
17 to 32	*Print position for other four fingers.*
33 to 39	*Chord name.*
40 to 57	*Values of notes for BEEP if user wants to hear chord.*












```
    1 REM * GUITAR TUTOR       *
           ALAN RENWICK      *
          * AUGUST 1983        *
    2 REM * The "flat" notation *
    3 REM * (b in the listing)  *
    4 REM * should be replaced  *
    5 REM * with GRAPHICS "A"   *
    6 REM ************************

   10 PAPER 6: INK 2: BORDER 6: C
LS : PRINT AT 10,1; FLASH 1;"STO
P THE TAPE- THEN PRESS A KEY"
   20 PAUSE 0: CLS
   30 BORDER 0: PAPER 0: INK 2: C
LS
   40 GO SUB 9980
   45 PRINT  INK 7;AT 11,11;"GUIT
AR TUTOR";AT 17,10; MICRO ORAC
LE"
   50 PAUSE 300: PRINT  INK 6; FL
ASH 1;AT 0,6;"PRESS ANY KEY TO S
TART"
   60 PAUSE 0: CLS
  200 GO SUB 9000
  210 POKE 23658,8
  500 REM  *  MENu  *
  510 INK 7: PAPER 1: BORDER 1: I
NVERSE 1: CLS
  520 PRINT AT 1,8;"GUITAR TUTOR.
"
  525 PRINT  OVER 1;AT 1,8;"_____
_________"
  530 PRINT '''"CHAPTER 1-INTRODUC
TION TO TUTOR"
  540 PRINT '''"CHAPTER 2-TUNING T
HE GUITAR"
  550 PRINT '''"CHAPTER 3-FINGERBO
ARD DIAGRAM"
  560 PRINT '''"CHAPTER 4-TRANSPOS
ING SONGS"
  570 PRINT '''"CHAPTER 5-TWENTY B
ASIC CHORDS"
  580 INPUT  INK 6;"PLEASE TYPE I
N THE NUMBER OF THECHAPTER YOU W
ANT TO USE(0 QUITS PROGRAM) ";X$
  583 IF X$<>"0" AND X$<>"1" AND
X$<>"2" AND X$<>"3" AND X$<>"4"
AND X$<>"5" THEN  GO TO 580
  585 INVERSE 0
  586 IF X$="0" THEN  CLS : GO SU
B 9980: STOP
  590 IF VAL X$<1 OR VAL X$>5 THE
N  GO TO 580
  595 GO SUB VAL X$*1000
  600 GO TO 510
 1000 REM *  INTRODUCTION  *
 1010 PAPER 6: BORDER 6: INK 1: C
LS
 1100 PRINT  INVERSE 1;AT 0,8;"CH
APTER ONE."
 1110 INVERSE 0: PRINT AT 3,4;"GU
ITAR TUTOR IS DESIGNED TO PROVID
E BASIC INFORMATION FOR  GUITAR
 PLAYERS."
 1120 PRINT '"IT WILL HELP YOU WI
TH TUNING THEGUITAR, LEARNING CH
ORDS,AND EVENTRANSPOSING SONGS F
ROM ONE KEY  TO ANOTHER."
 1130 PRINT '"YOU CAN USE GUITAR
TUTOR LIKE A BOOK,DECIDING ON WH
ICH CHAPTER  YOU WANT TO USE, SO
 THERE'S NO  NEED TO GO  THROUGH
 THE WHOLE  PROGRAM IF YOU ONLY
 WANT ONE BITOF IT. EACH CHAPTER
 IS SELF-    EXPLANATORY."
 1140 PRINT  INVERSE 1; INK 2;"
TO RETURN TO MENU AT ANY TIME
       JUST PRESS M.          "
```

```
1150 PRINT  INVERSE 0; FLASH 1;"
     PRESS M  TO CONTINUE
"
1160 FLASH 0: IF INKEY$<>"M" THE
N  GO TO 1160
1170 IF INKEY$="M" THEN  RETURN
2000 REM * TUNING THE GUITAR *
2010 BORDER 7: PAPER 7: INK 0: C
LS
2100 PRINT AT 0,8; INVERSE 1;"CH
APTER 2."
2110 INVERSE 0: PRINT '"THERE AR
E MANY TYPES OF TUNING  WHICH CA
N BE USED ON THE GUITAR,BUT THE
MOST COMMON IS THE ONE  SHOWN ON
 THE NEXT PAGE."
2120 PRINT '"ABOVE EACH STRING I
S THE NAME OFTHE NOTE THAT THE S
TRING SOUNDS WHEN OPEN. THE DIAG
RAM SHOWS HOWTHE GUITAR CAN BE T
UNED IN A  RELATIVE WAY."
2130 PRINT '"THE NEXT PAGE SHOWS
 HOW TO TUNE THE GUITAR WITH THE
 HELP OF ANY KEYBOARD."
2135 PRINT '"THE LAST PAGE GIVES
 A METHOD OF TUNING USING THE SP
ECTRUM'S OWN LOUDSPEAKER"
2150 PRINT  FLASH 1;"PRESS A KEY
 TO CONTINUE(M/MENU) "
2155 IF INKEY$="" THEN  GO TO 21
55
2160 IF INKEY$="M" THEN  RETURN
2200 REM * PAGE 2 *
2205 BORDER 1: INK 7: PAPER 1: C
LS
2210 GO SUB 9500
2215 PRINT AT 0,7;"RELATIVE TUNI
NG"; OVER 1;AT 0,7;"_____________
___"
2220 PRINT  INVERSE 1;AT 14,9;"A
";AT 14,11;"D";AT 14,13;"G";AT 1
2,15;"B";AT 14,17;"E"
2225 PLOT 75,62: DRAW 16,83: PLO
T 91,62: DRAW 16,83: PLOT 107,62
: DRAW 16,83: PLOT 123,78: DRAW
16,68: PLOT 139,62: DRAW 16,83
2230 INK 7: PRINT AT 15,0;"TO TU
NE THE GUITAR IN RELATION  TO IT
SELF USE THE DIAGRAM ABOVE.E.G.T
O TUNE THE "; INVERSE 1;"A"; INV
ERSE 0;" STRING TO THE BOTTOM ";
 INVERSE 1;"E"; INVERSE 0;" ,FRE
T THE "; INVERSE 1;"E"; INVERSE
0;" AT THE 5TH FRET GIVING "; IN
VERSE 1;"A";".REPEAT FOR OTHER
STRINGS UNTIL ALL ARE IN TUNE."
2235 PRINT  INVERSE 1; INK 5;" I
=INFO  M=MENU  OTHER=CONTINUE "
2240 IF INKEY$="" THEN  GO TO 22
40
2245 IF INKEY$="M" THEN  RETURN
2250 IF INKEY$="I" THEN  GO TO 2
000
2375 REM * PAGE 3 *
2380 PAPER 7: BORDER 7: INK 0: C
LS
2385 PRINT  INK 1;"      TUNING U
SING A KEYBOARD": PRINT  OVER 1;
AT 0,5;"_________________________"
2390 PRINT  INK 2; INVERSE 1;AT
4,0;"C=MID C"
2395 PRINT  INVERSE 1;AT 10,20;"
ANY KEY TO";AT 11,20;" CONTINUE
";AT 12,20;" M = MENU ";AT 13,20
;" I = INFO "
2400 GO SUB 9500
2410 FOR Y=2 TO 257 STEP 16
2420 PLOT Y,0: DRAW 0,30
2430 NEXT Y
2440 LET U=1: GO SUB 9800
2450 FOR U=33 TO 80 STEP 16: GO
SUB 9800
2460 NEXT U
2470 FOR U=97 TO 128 STEP 16: GO
 SUB 9800
2480 NEXT U
2490 FOR U=145 TO 182 STEP 16: G
O SUB 9800
2500 NEXT U
2510 FOR U=209 TO 238 STEP 16: G
O SUB 9800
2520 NEXT U
2530 PLOT 0,0: DRAW 255,0: PLOT
0,30: DRAW 255,0
2540 PRINT  INVERSE 1;AT 21,1;"E
";AT 21,7;"A";AT 21,13;"D";AT 21
,19;"G";AT 21,23;"B"; INK 2;AT 2
1,25;"C"; INK 0;AT 21,29;"E"
2545 INK 2
2550 PLOT 10,34: DRAW 65,33
2555 PLOT 55,34: DRAW 35,33
2560 PLOT 105,34: DRAW 0,30
2570 PLOT 150,34: DRAW -30,33
2575 PLOT 185,34: DRAW -45,33
2580 PLOT 235,34: DRAW -85,33
2585 INK 0
2590 IF INKEY$="" THEN  GO TO 25
90
2595 IF INKEY$="M" THEN  RETURN
2597 IF INKEY$="I" THEN  GO TO 2
000
2598 REM *  PAGE 4  *
2600 PAPER 7: BORDER 7: INK 0: C
LS
2610 PRINT ;TAB 5;"TUNING USING
THE SPECTRUM": PRINT  OVER 1;AT
0,5;"_________________________"
2620 PRINT AT 3,0;"USING THE SPE
```

```
CTRUM'S INTERNAL   SPEAKER IT IS
 POSSIBLE TO TUNE  A GUITAR. ALT
HOUGH NOT ENTIRELY ACCURATE A VE
RY GOOD RESULT IS  POSSIBLE."
2630 PRINT AT 9,0;"USING STANDAR
D PRACTICE THE     STRINGS ARE N
UMBERED FROM 1 TO 6AS SHOWN BELO
W. SIMPLY PRESS THEREQUIRED NUMB
ER ON THE KEYBOARD AND THE CORRE
CT NOTE WILL SOUND.BY REPEATING
THIS PROCEDURE YOU CAN TUNE THE
GUITAR."
2640 PRINT  INVERSE 1; INK 6; IN
K 2;AT 18,0;" PRESSING 7 PLAYS A
LL STRINGS.  "
2650 PRINT  INK 2;AT 19,0;"   6
2650 PRINT  INK 2;AT 19,0;"   6
   5    4    3    2    1   ";AT
20,0;"   E    A    D    G    B
  E   "
2654 PRINT "       M = MENU I=INF
ORMATION    "
2655 INPUT Q$
2657 IF Q$="I" THEN  GO TO 2000
2658 IF Q$="M" THEN  RETURN
2660 IF Q$<>"1" AND Q$<>"2" AND
Q$<>"3" AND Q$<>"4" AND Q$<>"5"
AND Q$<>"6" AND Q$<>"7" THEN  BE
EP .3,20: GO TO 2655
2665 IF Q$="1" THEN  BEEP 1,4
2670 IF Q$="2" THEN  BEEP 1,-1
2675 IF Q$="3" THEN  BEEP 1,-5
2680 IF Q$="4" THEN  BEEP 1,-10
2685 IF Q$="5" THEN  BEEP 1,-15
2690 IF Q$="6" THEN  BEEP 1,-20
2695 IF Q$="7" THEN  BEEP 1,-20:
 PAUSE 50: BEEP 1,-15: PAUSE 50:
 BEEP 1,-10: PAUSE 50: BEEP 1,-5
: PAUSE 50: BEEP 1,-1: PAUSE 50:
 BEEP 1,4
2700 GO TO 2655
2999 PAUSE 0
3000 REM * FINGERBOARD *
3100 BORDER 0: PAPER 0: INK 7: C
LS
3110 PRINT AT 0,7;"FINGERBOARD L
AYOUT": PRINT  OVER 1;AT 0,7;"__
_________________"
3120 PRINT AT 5,0;"ON THE FOLLOW
ING PAGE THERE IS ADIAGRAM WHICH
 SHOWS THE NOTES   THAT ARE PROD
UCED BY FRETTING   STRINGS AT TH
E FIRST 12 FRETS.  AFTER 12 FRET
S THE PATTERN REPEATS ITSELF."
3130 PRINT '' INVERSE 1;"PLEASE
NOTE"; INVERSE 0;" THE FOLLOWING
 NOTES HAVE TWO NAMES:"
3140 PRINT ''" A#/Bb C#/Db D#/Eb
 F#/Gb G#/Ab  "
3150 PRINT '' FLASH 1;" M = MENU
  ANY OTHER = CONTINUE "
3155 IF INKEY$="" THEN  GO TO 31
55
3160 IF INKEY$="M" THEN  RETURN
3165 CLS
3167 REM * PAGE 2 *
3170 PRINT AT 0,7;"FINGERBOARD L
AYOUT": PRINT  OVER 1;AT 0,7;"__
_________________"
3175 PRINT AT 2,8;"E   A   D   C
   B   E"
3176 RESTORE 3000
3180 FOR N=1 TO 12
3190 READ D$
3200 DATA "F   Bb  Eb  G#  C   F
","F#  B   E   A   C#  F#","G
C   F   Bb  D   G","G#  C#  F#
B   Eb  G#","A   D   G   C   E
 A","Bb  Eb  G#  C#  F   Bb","B
  E   A   D   F#  B","C   F   Bb
  Eb  G   C","C#  F#  B   E   G#
  C#","D   G   C   F   A   D","E
b  G#  C#  F#  Bb  Eb","E   A
D   G   B   E"
3210 PRINT AT N+3,8;D$
3220 NEXT N
3225 FOR N=2 TO 14: PRINT  OVER
1;AT N,9;"_____________________":
 NEXT N
3226 PRINT  PAPER 2;AT 3,9;"____
_________________"
3230 FOR N=1 TO 12: PRINT  INK 6
;AT N+3,0;"FRET ";N
3235 NEXT N
3240 FOR N=72 TO 244 STEP 32
3245 PLOT N,40
3250 DRAW 0,112
3255 NEXT N
3260 PRINT ''" A#/Bb C#/Db D#/Eb
 F#/Gb G#/Ab  "
3270 PRINT  INVERSE 1''"PRESS 'M
' TO RETURN TO THE MENU "
3300 IF INKEY$<>"M" THEN  GO TO
3300
3310 RETURN
4000 REM * TRANSPOSING *
4020 RESTORE 3210
4050 BORDER 6: PAPER 6: INK 2: C
LS
4060 PRINT AT 0,6;"TRANSPOSING S
ONGS.": PRINT  OVER 1;AT 0,6;"__
_________________"
4070 PRINT AT 2,0;"IT IS OFTEN T
HE CASE THAT THE   KEY THAT A SO
NG IS WRITTEN IN ISUNSUITABLE FO
R YOU. IT COULD BE TOO HIGH,OR T
OO LOW,OR PERHAPS  THE CHORDS AR
E NOT ONES THAT YOUKNOW. THE TAB
```

```
LE ON THE FOLLOWINGPAGE ENABLES
YOU TO CHANGE THE  KEY OF A SONG
 SIMPLY.THIS IS THEPROCESS CALLE
D TRANSPOSING."
4080 PRINT ''"TO USE THE TABLE L
OOK DOWN THE  LEFT COLUMN FOR TH
E ORIGINAL KEYAND THE KEY YOU WA
NT TO CHANGE  IT TOO. BY LOOKING
 ACROSS YOU   WILL SEE WHICH CHO
RD IN THE NEW KEY CORRESPONDS TO
 THE ORIGINAL."
4090 PRINT #1; INVERSE 1;"   M=M
ENU   ANY OTHER=CONTINUE  "
4100 IF INKEY$="" THEN  GO TO 41
00
4110 IF INKEY$="M" THEN  RETURN
4120 CLS
4130 REM * PAGE 2 *
4190 PRINT  INVERSE 1;AT 0,8;"TR
ANSPOSITION CHART";AT 1,5;"KEY"
4200 FOR N=1 TO 9: READ T$
4250 PRINT AT N*2+2,5;T$
4255 PRINT  OVER 1;AT N*2+2,5;"_
_____________________________"
4260 NEXT N
4265 FOR N=39 TO 255 STEP 24
4270 PLOT N,8: DRAW 0,144
4275 NEXT N
4277 PRINT AT 2,5;"_____________
_____________"
4280 DATA "C  C  D  E  F  G  A
B  C","D  D  E  F# G  A  B  C# D
","Eb Eb F  G  Ab Bb C  D  Eb","
E  E  F# G# A  B  C# D# E","F  F
  G  A  Bb C  D  E  F","G  G  A
 B  C  D  E  F# G","Ab Ab Bb C
Db Eb F  G  Ab","A  A  B  C# D
E  F# G# A","Bb Bb C  D  Eb F  G
  A  Bb"
4290 FOR x=3 TO 20: PRINT  PAPER
 7; OVER 1;AT x,5;"    ": NEXT x
4300 PRINT #1; INVERSE 1;" PRESS
 M FOR MENU      I FOR INFO"
4305 IF INKEY$<>"M" AND INKEY$<>
"I" THEN  GO TO 4305
4310 IF INKEY$="I" THEN  GO TO 4
000
4320 IF INKEY$="M" THEN  RETURN
5000 REM * CHORDS *
5001 PAPER 1: BORDER 1: INK 7: C
LS
5150 PRINT  INK 6;AT 0,6;"20 USE
FUL CHORDS.": PRINT  OVER 1; INK
 6;AT 0,6;"__________________"
5160 PRINT '"THE NEXT PAGE CONTA
INS A LIST OFTWENTY OF THE MOST
COMMONLY USEDCHORDS.BY TYPING IN
 THE NUMBER  OF THE CHORD YOU CA
N SEE HOW IT IS PLAYED.EACH DISP
LAY ALSO HAS THE OPTION OF LISTE
NING TO THE  CHORD."
5170 PRINT  ''"USING THIS COLLEC
TION OF CHORDS IT IS POSSIBLE TO
 PLAY MOST POP SONGS."
5180 PRINT  INK 6; INVERSE 1;AT
20,0;"  M FOR MENU  OTHER TO CON
TINUE."
5190 PAUSE 0
5200 IF INKEY$="M" THEN  RETURN
5250 CLS
5300 REM * PAGE 2 *
5430 CLS
5444 FOR X=1 TO 20: PRINT AT X,8
;X: PRINT AT X,12;C$(X)(33 TO 39
): NEXT X
5450 INPUT "ENTER NUMBER OF CHOR
D ";Z
5455 IF Z<1 OR Z>20 THEN  GO TO
5450
5456 CLS
5457 REM *  PAGE 3  *
5460 LET F$=C$(Z)
5500 FOR X=124 TO 244 STEP 24
5510 PLOT X,144
5520 DRAW 0,-120
5530 NEXT X
5540 FOR X=136 TO 24 STEP -24
5550 PLOT 124,X
5560 DRAW 120,0
5570 PLOT 124,144
5580 DRAW 120,0
5590 NEXT X
5600 PRINT AT 2,15;"E";AT 2,18;"
A";AT 2,21;"D";AT 2,24;"G";AT 2,
27;"B";AT 2,30;"E"
5700 PRINT AT 21,15;F$(1 TO 2);A
T 21,18;F$(3 TO 4);AT 21,21;F$(5
 TO 6);AT 21,24;F$(7 TO 8);AT 21
,27;F$(9 TO 10);AT 21,30;F$(11 T
O 12)
5750 LET C=VAL F$(13 TO 14)
5760 LET D=VAL F$(15 TO 16)
5770 IF F$(17)="B" THEN  FOR A=D
 TO 30 STEP 3: PRINT AT C,A;"B":
 NEXT A
5780 PRINT AT C,D;F$(17)
5790 LET C=VAL F$(18 TO 19)
5800 LET D=VAL F$(20 TO 21)
5810 PRINT AT C,D;F$(22)
5820 LET C=VAL F$(23 TO 24)
5830 LET D=VAL F$(25 TO 26)
5840 PRINT AT C,D;F$(27)
5850 LET C=VAL F$(28 TO 29)
5860 LET D=VAL F$(30 TO 31)
5870 PRINT AT C,D;F$(32)
5890 PRINT  INVERSE 1;AT 5,3;F$(
33 TO 39)
5900 PRINT  INVERSE 1;AT 12,0;"M
```

```
=MENU          ";AT 13,0;"A=ANOTH
ER CHORD";AT 14,0;"H=HEAR CHORD
  "
5910 INPUT T$
5920 IF T$="M" THEN  RETURN
5930 IF T$="A" THEN  GO TO 5430
5960 IF T$<>"M" AND T$<>"A" AND
T$<>"H" THEN  GO TO 5910
6000 IF Z=2 OR Z=17 THEN  PAUSE
50: BEEP 1,VAL F$(40 TO 42): BEE
P 1,VAL F$(43 TO 45): BEEP 1,VAL
 F$(46 TO 48): BEEP 1,VAL F$(49
TO 51)
6010 IF Z=9 OR Z=10 THEN  PAUSE
50: BEEP 1,VAL F$(40 TO 42): BEE
P 1,VAL F$(43 TO 45): BEEP 1,VAL
 F$(46 TO 48): BEEP 1,VAL F$(49
TO 51): BEEP 1,VAL F$(52 TO 54)
6020 IF Z<>2 AND Z<>17 AND Z<>9
AND Z<>10 THEN  PAUSE 50: BEEP 1
,VAL F$(40 TO 42): BEEP 1,VAL F$
(43 TO 45): BEEP 1,VAL F$(46 TO
48): BEEP 1,VAL F$(49 TO 51): BE
EP 1,VAL F$(52 TO 54): BEEP 1,VA
L F$(55 TO 57)
6030 GO TO 5910
9000 REM *  DATA  *
9050 RESTORE 9000
9100 FOR N=0 TO 7: READ A: POKE
USR "A"+N,A: NEXT N
9120 DATA 64,64,72,84,100,72,80,
96
9210 DIM C$(20,57)
9220 LET C$(1)="E A E A C#E  921
1 9242 92730000 A MAJOR-20-15-08
-03001004"
9230 LET C$(2)="X X E A C#G  921
B 000  000 12302A SEVEN-08-03001
006000000"
9240 LET C$(3)="E A E A C E  627
1 9212 92430000 A MINOR-20-15-08
-03000004 "
9250 LET C$(4)="F#B F#B D#F# 915
B152121524315274B MAJOR-18-13-06
-01003006"
9260 LET C$(5)="F#B D#A B F# 621
1 9182 9243 9304B SEVEN-18-13-09
-03-01006"
9270 LET C$(6)="F#B F#B D F# 930
1122721521315244B MINOR-18-13-06
-01002006"
9280 LET C$(7)="G C E G C E  627
1 92121218312154C MAJOR-17-12-08
-05000004"
9290 LET C$(8)="G C E BbC E  627
1 92121218312244C SEVEN-17-12-08
-05000004"
9300 LET C$(9)="X A D A D F# 924
1 9302122730000 D MAJOR-15-10-03
002006000"
9310 LET C$(10)="X A D A C F# 62
71 9242 93030000 D SEVEN-15-10-0
3000006000"
9320 LET C$(11)="X A D A D F  63
01 9242122730000 D MINOR-15-10-0
3002005"
9330 LET C$(12)="E B E G#B E  62
41 9182 92130000 E MAJOR-20-13-0
8-04-01004"
9340 LET C$(13)="E B E G#D E  62
41 9182 921312274E SEVEN-20-13-0
8-04002004"
9350 LET C$(14)="E B E G B E  92
12 91830000 0000 E MINOR-20-13-0
8-05-01004"
9360 LET C$(15)="F C F A C F  61
5B 92421218312214F MAJOR-19-14-0
7-03000005"
9370 LET C$(16)="F C EbA C F  61
5B 9242121830000 F SEVEN-19-12-0
9-03000005"
9380 LET C$(17)="F C F AbC F  61
5B12182122130000 F MINOR-07-0400
0005000000"
9390 LET C$(18)="G B D G B G  91
8212153123040000 G MAJOR-17-13-1
0-05-01007"
9400 LET C$(19)="G B D G B F  63
01 9182121530000 G SEVEN-17-13-1
0-05-01005"
9410 LET C$(20)="G D G BbD G 121
5B18183182140000 G MINOR-17-14-1
0-05002007"
9420 RETURN
9500 FOR X=74 TO 154 STEP 16
9510 PLOT X,144
9520 DRAW 0,-80
9530 NEXT X
9540 FOR X=136 TO 72 STEP -16
9550 PLOT 74,X
9560 DRAW 80,0
9570 PLOT 74,144
9580 DRAW 80,0
9590 NEXT X
9600 PRINT  INVERSE 1;AT 2,9;"E"
;AT 2,11;"A";AT 2,13;"D";AT 2,15
;"G";AT 2,17;"B";AT 2,19;"E"
9610 RETURN
9800 PLOT U,15: DRAW 0,15
9810 PLOT U+2,15: DRAW 0,15
9820 RETURN
9950 REM *   OPENING/CLOSING   *
          *      GRAPHICS      *
9980 FOR A=0 TO 43: LET X=70*SIN
 A: LET Y=70*COS A
9985 PLOT 128,88: DRAW X,Y,PI: N
EXT A
9986 RETURN
```

ADRIAN MARSH

ROM roundup

A quick guided tour of the Spectrum ROM reveals some useful routines.

The Spectrum ROM holds many secrets. The most interesting are the monitor routines and in this article I shall outline a few of these routines, and explain how they may be used.

ON SCREEN PRINTING

There are a number of ways of transferring information to the screen:

1 Hex address: 0B24-0BDA. This is the PRINT-ANY character subroutine. On entry the HL register pair holds the pixel address of where the character is to be printed; the BC register pair holds the current line and column values and the A register holds the character code.

This method is complicated so it may be easier to use the second method.

2 This subroutine is contained within another, therefore the whole routine begins at 0D68 and ends at 0EAB. At the present the reader is interested in location 0DD9. This sets the printing locations to BC:

i) Load the BC register pair with the appropriate values.
ii) Call CL-SET and Call 0DD9 which enters required values.

BC is equivalent for a position AT a,b; the B register holds the line (hex) 18-a and the C register holds the column (hex) 21 -b. Therefore, a routine to produce an equivalent to PRINT At 11,15; would be:

```
01 13 0D      LD BC, 0D13
CD D9 0D      CALL 0DD9
3E??          LD A, (character)
D7            RST 0010
C9            RET
```

3 The routine PR-STRING can be used tp print any string. The monitor routine consists of:

```
PR-STRING address   203C
(label) PR-STRING   LD A,B,
                    OR C
                    DEC BC
                    RET Z
                    LD A,(DE)
                    INC DE
                    RST 00010
                    JR PR-STRING
```

Any string of characters can therefore be printed by:

i) Loading the start address into DE
ii) Loading the length of the address into BC
iii) Call the PR-STRING at address 203C

CLEARING THE SCREEN

The B register holds a value in the range of (hex) 01-18. Therefore, (hex) 18 would clear the whole screen.

The CL-LINE routine begins at 0E44 and is very short:

```
06 18         LD B, 18
CD 44 0E      CALL CL-LINE
C9            RET
```

This will clear the entire screen whereas if B were loaded with 17, all but the top line would be cleared.

SOUND

There are two routines in the ROM for producing sound, the BLEEPER and the BEEP.

1 The hex address for the BLEEPER is 03B5-03F7. On entry the HL register pair hold the pitch and the DE register pair holds the duration. The duration value has to be increased as the pitch value is decreased. The pitch for middle C is 0666 and the duration for a second is 0105 so a routine for BEEP 1,0 would be:

```
11 05 01      LD DE, 0105
21 66 06      LD HL, 0666
CD B5 03      CALL BLEEPER
C9            RET
```

2 The hex address for the BEEP routine is 03F8-046D. This routine uses the calculator to change the duration and pitch into appropriate values for the DE and HL register pairs.

USING THE KEYBOARD

The main routine is the KEY-SCAN.

1 Hex address for the KEY-SCAN is 028E-02BE. On leaving the routine, DE is returned with a key value. The zero flag is reset if more than one key is pressed at the same time. The D register indicates which shift keys are being pressed and the E register contains the key number (hex) 00-27.
2 At location 02BF-03B4 are found the KEYBOARD subroutines. These handle the repeat facility and decode the key-value to give the required character code. If the code is accepted then it is placed in the system variable LAST-K and bit 5 of the FLAGS is set.
3 At location 10A8-111C there is the KEYBOARD-INPUT routine. This routine copies the value from LAST-K and depends on bit 5 of the FLAGS. It then returns with the carry flag set or reset if the code is printable.
4 The subroutine at location 15D4-1651 literally has the effect of a PAUSE 0 or 'waiting for a key to be pressed'. So, a pause 0 in machine code would consist of:

CD D4 15 CALL 15D4

LOADING AND SAVING

The whole routine starts at 04C2 and ends at 09F3.

1 SAVING This subroutine begins at 04C2-053E and passes the DE bytes by starting at the (IX) location and continuing to the cassette recorder with the initial marker byte and parity byte. An example of this is:

i) The amount of bytes to save is 255 (FFh)
ii) The block is stored at location 25000 (61A8h)

Therefore:

```
3EFF          LD,A,255
DD21 A861     LD IX, + START
110 001       LD DE, + COUNT
C9            RET
```

This routine saves 255 bytes starting at 25000. However, these bytes are saved without a header, and can only be loaded if the count is known.

2 LOADING This subroutine begins at 0556-0604 and loads the DE bytes and the IX register pair points to the first location. When loading the carry flag must be set, but if it is reset then VERIFY can be

used. Therefore:

```
(37)                    (SCF loading only)

3EFF                    LD A,FF
0D21 A861               LD IX,START

110001                  LD DE,COUNT
CD5605                  LOAD ROUTINE
C9                      RET
```

This routine loads 255 bytes into 25000.

I shall now continue to shed some light on the RST commands or restart commands.

1 RST 0 (0000-0007h). This does a number of things in this order:

i) Disable the maskable interrupt
ii) Clears the A register
iii) Loads the DE register pair with 65535 (FF-FFh)
iv) Jumps forward to 11CB

At 11Cb are the initialisation routines (equivalent to NEW).

2 RST 8 (0008-000Fh).

These are the error routines. There are two outcomes of this:

i) The stack will be cleared.
ii) The appropriate report is given.

3 RST 10(0010-0012). This is the PRINT A character routine. A jump forward to location 15F2 is made.
4 RST 18 and RST 20 (0018-0024h). This fetches the current character pointed to by the CH-ADD routine.
5 RST 28 (0028-0029h). This jumps forward to location 335B which is the first address of the calculator.
6 RST 30 (0030-0037h). This is the BC-SPACES and jumps forward to 16 9E to make a space in the work space.

A useful routine at 11DA is the RAM-CHECK routine. All locations from RAMTOP to 4000 are tested. The HL register pair holds the address of the last location of memory available.

By now the reader should have a good idea of the monitor programs in the ROM. Although I have only outlines a handful, they are sure to come in useful in writing new machine code routines.

HUGH WALKER

Warlock

Adventurous types strap on your shields and swords (after typing this game in first, of course!).

One of the problems with adventures is that you can often complete the game from the knowledge gained by entering it. Hugh has adopted two methods to prevent this from spoiling your pleasure. Firstly he has not REMmed the listing and has laid it out in an illogical way, and secondly he has used random elements to provide surprises without losing too much player skill.

You are a 'novice-hero' whose task is to raid the Warlock's mansion and find all his gold, using the keys which you find to unlock any doors and fighting off his pet monsters with your sword and shield. Each key may be used once only, but not all the doors are locked.

TREASURE

The questionable ethics of a young hero turning thief are satisfied by the fact the Warlock's treasure has been extorted from the oppressed peasants in his locality. Should you be successful, you will, no doubt, restore the gold to its rightful owners. The hero-rating which you acquire directly reflects the amount of gold you have found, and no one, except the Warlock, can take this from you. Indeed, it is possible that, in some future program, you will return as a wiser, stronger hero to deal with the Warlock's permanently!)

To find the treasure and the keys you must open doors, explore hallways and defeat monsters whenever the opportunity arises. However, you may not find anything and be given an opportunity to rest and build up your strength. This strength is limited and it is reduced by fighting, so the opportunity to rest is useful. The Warlock is away at the start of the game, but he may return and catch you. If he does he will either fight you, proving difficult to chase away, or he will use his magic to turn you into a small creature, robbing you of your hero status (hero-rating).

TIME TO GO

When you have collected all the treasure you will be congratulated and told to find your way out "...if you can!" The way out is through a door on level 0 so you are also told which level you are on. You then have to make your way to level 0, open the door (which may be locked) and leave triumphantly.

Whenever you are confronted by a monster or the Warlock himself in a fighting mood, you will be given the option to "hit it (him)" or to "run away". You cannot kill the monsters (you are only a young hero) but you can chase them away. However, some are difficult to defeat and each blow you deliver uses up some of your strength. If you use up all your strength, then you have lost! Nevertheless, if you choose to run away you will be called a COWARD and lose some of your HQ (hero-quotient), and you MAY lose some treasure...or, you MAY be trapped by the monster. Your HQ is a factor in calculating your 'Hero-rating'...so your honour is at stake! If your attempt to hit the monster, or Warlock, fails to chase it away or your attempt to run away has been unsuccessful, you will be given the option to 'hit it' or 'run away' again.

As you move around the mansion you will find one of three locations: 'Door', 'Hallway' and 'Stairway'.

Oh! By the way, only the computer knows what the total amount of treasure to be found is and it won't tell you until you have reached that amount (it will be between 1500 and 2500 coins). Good luck!

INSTRUCTIONS

1. *Door: (may be locked) :you have two options: 'Open' or 'Leave'.*
   If you open the door, you may find a reward...or a monster...or nothing, and have to leave. If the door is locked and you have no key, then you have to leave...each time you leave a location, whether by choice or not, you risk meeting a monster, which may result in loss of treasure.
2. *Hallway : three options : 'Move', 'Explore', or 'Status'.*
   Move: moving backwards or sideways may result in a bump on the head from a low beam, causing some loss of strength; moving forwards is safe because you would see the low beam in front of you!
   Explore: you may find treasure, a key, a rest, a monster or another location or nothing and have to leave...with the same risk as above.
   Status: Lists your possessions, what level you are on and returns you to the options 'Move, Explore or Status' when you are ready.
3. *Stairway :three options: 'Up', 'Down' or 'Leave'.*
   You will not know if the stairway goes up or down until you try. If you make the wrong choice you will have to leave.

VARIABLES USED

1. The following are DIMensioned arrays, READ from DATA.

m$(9,9)	*:The name of each monster (and Warlock).*
a$(3,9)	*:The three locations found (door, hallway, stairway).*
k$(3,6)	*:The three small creatures into which the Warlock may turn you.*
m(9)	*:The resistance of each monster to your blows.*
n(9)	*:The endurance of each monster to withstand successful blows.*
g(9)	*:The value by which your Hero-quotient is increased when you beat each monster and the length of*

each monster's name (for tabulation in lines 3090 and 3110).

z(11) :The length of the BEEPs in line 4000.

y(11) :The pitch of the BEEPs in line 4000.

2. The following are 'single' variables.

hi :High Score, carried forward to successive games ('hi'='hr')

*hr :Hero-rating, ie overall score in each game.

hq :Hero-quotient, increased by defeating monsters (by a randomly-adjusted value of 'g').

ded :The number of monsters beaten.

ht :The force of your blow when you hit a monster (randomly selected).

blo :The value by which a successful blow reduces the monster's endurance.

mn :Random value (1 to 9) to select monster from m$ (array), and associated values of 'm', 'n', and 'g'.

str :Your strength, initally 500: reduced by fighting or bumping your head: increased by resting.

wl :The value by which 'str' is reduced when you bump your head (randomly selected).

tres :The maximum amount of treasure to find (between 1500 and 2500; randomly selected).

tr :A logic variable to change course of program when 'tres' has been reached. It prevents more treasure being found and allows you to leave from line 1630 if you are on level 0.

csh :The amount of treasure you possess.

cn :The amount of treasure found each time.

ky :The number of keys in your possession.

wy :Random value (1 to 3) to select a$ and direct GO SUB in line 140.

fl :The level (floor) which you are on.

bp :Used in lines 400 to 460; 'bp'=+1 'Up' and bp=−1 for 'Down': it alters 'fl' and the pitch of the BEEP routine (line 460).

(*Your hero-rating is calculated by adding 'csh'+('ky'*5) + 'hq' and is uprated each time you change your location). In the course of the program 'f', 'i' and 'h' are used as FOR-NEXT variables for various purposes.

Please note:

1. Unlabelled Random-values are used to direct some of the 'GO TOs'.
2. INKEY$ is used for your responses to the various options offered you. This allows single-key entries to speed up operation of the game. However, PAUSEs and BEEPs have to be used to slow down the display.

EXTRAS

1. A lateral dimension could be added to this program so that the 'Move' routine translates sideways, forwards or backwards; within a 2-dimensional array from a fixed exit point, maintaining the vertical dimension ('fl') requirements to qualify for successful exit at line 1630. This exit point could be randomly generated in each game and added to the exit requirements at line 1630.
2. A random value could be generated (say between 0 and 10) as the vertical dimension which 'fl' would have to equal to qualify for successful exit at line 1630.

HOW IT RUNS

Line 20 :Initialise High Score for successive games.

Lines 50 to 60 :*Initialise each game (routine completed by calling subroutine 5000 to 5500)*

Lines 100 to 150 :*'Core-routine' to which all other routines return. Selects and PRINTs locations and directs GO SUB for each.*

Location *(lines 200 to 460)*

Lines 200 to 220 :*'Hallway' PRINTs and acts on options, using 'Go TO': 500, 600, or 700.*

Lines 300 to 330 :*'Door' PRINTs and acts on options using 'GOTO': 800 or 8200.*

Lines 400 to 460 :*'Stairway' PRINTs and acts on options, routine complete.*

Response to 'Hallway routine *(lines 500 to 780)*

Lines 500 to 520 :*Move selected by INKEYs "1" in line 220; PRINT's etc options (complete).*

Line 600 :*'Explore' selected by INKEYs "2" in line 220; uses 'GO TO' line 1650.*

Line 700 to 780 :*'Status' selected by INKEYs "3" in line 220, PRINTs possession, score, what level you are on and returns to line 200 when you are ready.*

Response to 'Door' routine

Line 800 :*'Open' selected by INKEYs "1" in line 320. Random chance of: (GO TO 1500 door locked) or (GO TO 1600 door open).*

Responses to 'Explore' and 'Door opens' routines *(lines 1000 to 1460)*

Lines 1000 to 1800 (from 1650) (or 8200) :*'Meet Monster' routine; Selects monster, PRINTs and acts on options to fight (GO TO line line 3000) or run away (GO TO line 8500) or random chances of monster stealing gold instead of fighting and if Warlock is met, random chance of Warlock using magic (GO TO line 1700).*

Lines 1100 to 1150 :*'Find gold' routine, from line 1650 (also from line 3130). If maximum amount has not been found, calculates the amount of each find (amount reduces in proportion to total already found), PRINTs amount found and increments gold score, then checks gold score against maximum.*

Lines 1200 to 1220 :*'Find key' routine from line 1650 (also from line 3130). Random chance of finding a key, or another location. if a key is found, PRINTs and increments one key to key score.*

Line 1300 (from 1650) :*'Find nothing' routine; uses lines 8000 to 8210 to complete routine.*

Lines 1400 to 1460 (from 1650) :*'Find rest' routine; random chance of finding rest of another location. If rest, then PRINTs, BEEPs and increments strength score.*

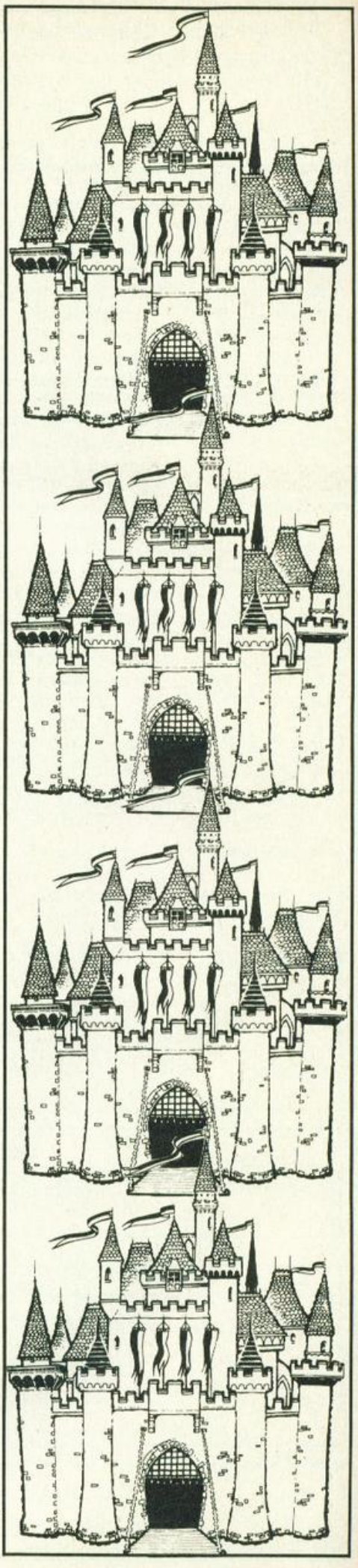
Continuation of 'Door' routine *(lines 1500 to 1650) from line 800*

Lines 1500 to 1550 :*'Door locked' routine; checks if you have a key (if not then GO TO 6500). If you have a key, then asks if you want to use it? (if "No" GO TO 6500). If "Yes" then decrements one from key-score and carries on to line 1600.*

Lines 1600 to 1650 :*'Door opens' routine; (from line 1550 or line 800). PRINTs, uses 4100 for BEEPs, checks if you have finished and randomly selects 'GO TO' :lines 1000, or 1100 or 1200 or 1300 or 1400.*

END routines *(Lines 1700 to 2050)*

Lines 1700 to 1770 :*'Bitter End' routine; (Warlock 'uses magic') from line 1030. PRINTs, selecting k$ from array, uses line*

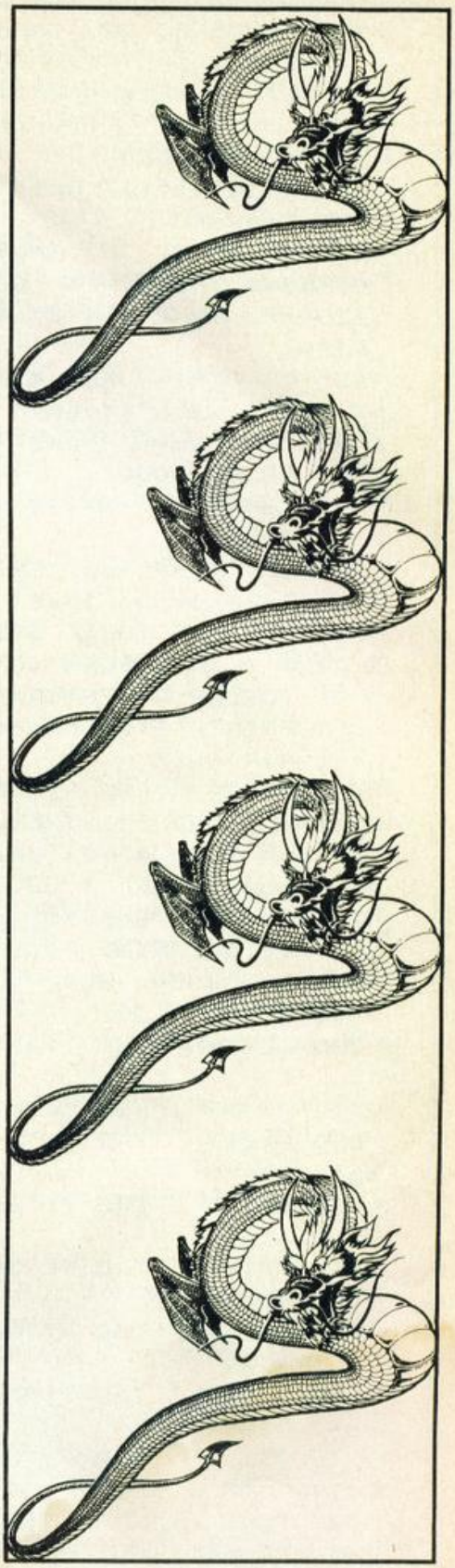


1900 to 1950 to complete.

Lines 1800 to 1880	:'Successful End' routine; from line 1630. PRINTs scores and uses 1900 to 1950 to complete.
Lines 1900 to 1950	:'Hi score' and 'Play again' routines; from lines 1770, 1880 or 2050. Increments High Score and asks if you want to try again. If "Yes" then goes to line 50; if "No" then STOPs.
Lines 2000 to 2050	:'Out of Strength' routine; from line 3010. PRINTs and uses lines 1850 to 1950 to complete.
Lines 2500 to 2570	:'All treasure found' routine; from line 1140. PRINTs, sets logic variable tr to 1, returns to 'core-routine' (L 100).

Fight Routine (lines 3000 to 3130), from line 1080.'

Lines 3000 to 3040	:Calculates the force of your blow, decrements strength, checks to see if you injured the monster, if you did, then 'GO TO' line 3050, if you did not, then goes to line 1050 for another try.
Lines 3050 to 3130	:Calculates how badly you hurt the monster, decrements monster's endurance; if monster is still fit to fight, then returns to line 1050, if monster is not fit to continue then PRINTs, increments 'hero-quotient', uses 'GO TO': 1100 or 1200 for reward.
Line 3500	:'Title' subroutine: PRINTs title, used in various parts of program to which it returns.
Lines4000 to 4500	:Beep routines (subroutines). Returns to 'core-routine' if 'GO TO 4000' etc is used.
Lines 5000 to 5190	:'Initialise routine' subroutine called by line 60. DIMs arrays, READs arrays from DATA (lines 7500 to 7800), PRINTs instructions and initialises variables, uses line 5500 to continue.
Line 5500	:'CONTINUE' routine; from various parts of program, returns to 'Core-routine' (1200)
Lines 6000 to 6500	:'Monster steals gold' routine; from line 1040. PRINTs, decrements gold-score, PRINTs new gold-score, checks if new gold score is below maximum and resets tr if necessary.
Line 6500	:"No!" routine from lines 1510. 1540; uses GO TO 8000 to continue.
Line 8000	:'Leave routine' from lines: 320, 430, 1300, 1460 and 6500 — carries on to 8200.
Lines 8200 to 8210	:Random chance of meeting another monster. From lines 330, 420 and 8000.
Lines 8500 to 8530	:'Run away' routine; from lines 1070. PRINTs "Coward" and decrements 'hero-quotient' then random chance of:
Line 8520	:(a) Dropping gold uses GO TO 6010 to decrement gold-score and return to 'core-routine'.
Line 8540	:(b) being trapped; returns to line 1050 for another try.
Lines 9000 to 9050	:'SAVE' routine; accessed by 'BREAK', 'GO TO 9000'.



```
  10 REM WARLOCK
  20 LET hi=0
  50 PAPER 0: INK 9: BORDER 0: C
LS
  60 GO SUB 5000
 100 LET wy= INT ( RND *3)+1
 110 CLS : PRINT " You find a ";
a$(wy)
 120 LET hr=csh+ky*5+hq
```

```
 140 GO SUB 100+100*wy
 150 GO TO 100
 200 PRINT '"(1) Move; (2) Explor
e; (3) Status "
 210 PAUSE 0: IF  INKEY$ <"1" OR
  INKEY$ >"3" THEN  GO TO 210
 220 GO TO 400+100* VAL  INKEY$
 300 PRINT '"(1) Open; (2) Leave
  "
```

```
 310 PAUSE 0: IF  INKEY$ <"1" OR
  INKEY$ >"2" THEN  GO TO 310
 320 IF  INKEY$ ="1" THEN  GO TO
 800
 330 GO TO 8200
 400 PRINT '"(1) Up;  (2) Down;
 (3) Leave"
 410 PAUSE 0: IF  INKEY$ <"1" OR
  INKEY$ >"3" THEN  GO TO 410
 420 IF  INKEY$ ="3" THEN  GO TO
 8200
 430 IF  INT ( RND *3) THEN  PRI
NT '" No way ";("up!" AND  INKEY
$ ="1");("down!" AND  INKEY$ ="2
");: GO TO 8000
 440 LET bp=( INKEY$ ="1")-( INK
EY$ ="2")
 450 LET f1=f1+bp: PRINT '" O.K.
 You are now on level ";f1
 460 FOR f=2 TO 30 STEP 2: BORDE
R  INT (f/4): BEEP .05,f*bp: NEX
T f: BORDER 0: RETURN
 500 PRINT '"1:Forward 2:Back 3:
Left 4:Right"
 510 PAUSE 0: IF  INKEY$ >"1" AN
D  INT ( RND *2) THEN  PRINT '"
You run into a low beam"'" and
hurt your head";: GO SUB 4500: L
ET w1= INT ( RND *5)+1: LET str=
str-w1*10: PRINT ".....Ouch!"
 520 PAUSE 50: RETURN
 600 GO TO 1650
 700 CLS : PRINT  TAB 12;"STATUS
"
 710 PRINT '"You have:"
 720 PRINT ' TAB 7;"A sword and
shield"
 730 PRINT ' TAB 7;"Strength....
....."str
 740 PRINT ' TAB 7;"Gold coins..
....."csh
 750 PRINT ' TAB 7;"Keys.......
....."ky
 760 PRINT ' TAB 7;"Monsters Bea
ten.."ded
 770 PRINT ' TAB 7;"Hero rating.
....."hr
 780 PRINT '' TAB 7;"You are on
level "f1 : GO SUB 5500: GO TO
200
 800 GO TO 1500+100* INT ( RND *
2)
1000 LET mn= INT ( RND *9)+1
1010 CLS : PRINT '" SUDDENLY!"''
" You meet the ";m$(mn)
1020 FOR f=2 TO 21: BEEP .04,10:
 BORDER  INT ( RND *7): BEEP .05
,-10: NEXT f: BORDER 0
1030 IF  NOT  INT ( RND *5) AND
mn=9 THEN  GO TO 1700
1040 IF  NOT  INT ( RND *10) THE
N  GO TO 6000
1050 INPUT "": PRINT '" What do
you do?"'"(1) Hit ";("it" AND m
n<9);("him" AND mn=9);"!  (2) Ru
n away!"
1060 PAUSE 0: IF  INKEY$ <"1" OR
  INKEY$ >"2" THEN  GO TO 1060
1070 IF  INKEY$ ="2" THEN  GO TO
 8500
1080 IF  INKEY$ ="1" THEN  GO TO
 3000
1100 IF tr THEN  PAUSE 50: RETUR
N
1110 LET cn= INT ( RND *(500/( I
NT (csh/500)+1)))+50
1120 PRINT ' INK 6;" You find ";
cn;" gold coins."
1130 LET csh=csh+cn
1140 IF csh >= tres THEN  GO TO
2500
1150 GO TO 4100
1200 IF  INT ( RND *2) THEN  PAU
SE 50: RETURN
1210 PRINT '" You find a key."
1220 LET ky=ky+1: GO TO 4100
1300 PRINT '" Nothing there";: G
O TO 8000
1400 IF  INT ( RND *2) THEN  RET
URN
1410 PRINT '" You find some some
"'" food and wine."'" Have a r
est."
1420 LET str=str+ INT ( RND *50)
+5: IF str>500 THEN  LET str=500
1430 PRINT '" ZZZZZ ZZZZZ ZZZZZ
ZZZZZ ZZZZZ"
1440 FOR f=1 TO 5: BEEP 1.5,-48:
 BEEP 0.2,-47: PAUSE 25: NEXT f
1450 PRINT '" Oh dear! Time to
go.";
1460 GO TO 8000
1500 PRINT '" Door locked!";: GO
 SUB 4500: PRINT "....Have you a
 key?"
1510 PAUSE 50: IF ky<1 THEN  GO
TO 6500
1520 PRINT '" Yes.";: PAUSE 20:
PRINT "do you want to use it?(Y/
N)"
1530 PAUSE 0: IF  INKEY$  <> "y"
 AND  INKEY$  <> "Y" AND  INKEY$
  <> "n" AND  INKEY$  <> "N" THE
N  GO TO 1530
1540 IF  INKEY$ ="n" OR  INKEY$
="N" THEN  GO TO 6500
1550 PRINT '" Yes......Unlock do
or.";: LET ky=ky-1: PAUSE 20: PR
INT "...click"
1600 PRINT '" The door opens.";
```

```
1610 GO SUB 4100
1620 PRINT ".....look!"
1630 IF  NOT f1 AND tr THEN  GO
TO 1800
1640 PAUSE 50
1650 GO TO 1000+100* INT ( RND *
5)
1700 CLS : GO SUB 3500
1710 PRINT ' TAB 9;"THE BITTER E
ND"' TAB 9;"**************"
1720 PRINT ''"    You met your E
nd at the "
1730 PRINT '"    hands of the WA
RLOCK."''"    He turned you into
 a ";k$( INT ( RND *3)+1)
1740 PRINT '"    Your Hero-ratin
g WAS ";hr
1750 PRINT ' INK 6;"    You foun
d ";csh;" gold coins."
1760 GO SUB 4000: PRINT  AT 19,1
0;"NOW HOP IT!": PAUSE 20: PRINT
  AT 19,14;"   "; AT 18,14;"HOP"
: PAUSE 10: PRINT  AT 18,14;"
"; AT 19,14;"HOP"
1770 GO TO 1900
1800 GO SUB 3500
1810 PRINT ' TAB 8;"Congratulati
ons!"
1820 PRINT '"   You've cheated t
he Warlock"
1830 PRINT ' TAB 8;"and his pets
!"
1840 FOR i=1 TO 3: GO SUB 4100:
NEXT i
1850 PRINT ' INK 6;"   You found
 ";csh;" gold coins."
1860 PRINT '"   You have ";ky;"
key";("s" AND ky <> 1);" left."
1870 PRINT '"   You defeated ";d
ed;" Monsters."
1880 PRINT '"   Your hero-rating
 ";("i" AND str>0);("wa" AND str
 <= 0);"s ";hr
1900 IF hr>hi THEN  LET hi=hr
1910 PRINT  AT 21,7;"Highest Sco
re:";hi
1920 INPUT "": PRINT #0; TAB 7;"
Another go ? (Y/N) ": PAUSE 0
1930 IF  INKEY$ ="Y" OR  INKEY$
="y" THEN  CLS : GO TO 50
1940 IF  INKEY$ ="n" OR  INKEY$
="N" THEN  STOP
1950 GO TO 1920
2000 GO SUB 3500
2010 PRINT '"   You've run out o
f strength."
2020 PRINT '"   The Warlock has
won again!"
2030 GO SUB 4000
2040 PRINT ' TAB 8;"Before you d
ied"
2050 GO TO 1850
2500 GO SUB 4100: CLS : PRINT  A
T 1,10;"WELL DONE!"
2510 PRINT '' TAB 3;"You have fo
und all of "
2520 PRINT ' TAB 3;"the Warlock'
s treasure!"
2525 PRINT ' TAB 3;"Now get out"
;: PAUSE 20: PRINT "..if you can
!"
2530 PRINT '' TAB 3;"You are on
level ";f1
2540 PRINT ' TAB 3;"The way out
is through "
2550 PRINT ' TAB 3;"a door on le
vel 0"
2560 PAUSE 30: PRINT ''' TAB 10;
"Good luck!"
2570 LET tr=1: GO TO 5500
3000 LET ht= INT ( RND *10)+1
3010 LET str=str-ht*( INT ( RND
*2)+1): IF str<1 THEN  GO TO 200
0
3020 IF ht>m(mn) THEN  GO TO 305
0
3030 PRINT '" You didn't hurt th
e ";m$(mn)
3040 GO SUB 4500: GO TO 1050
3050 LET blo=ht-m(mn)
3060 PRINT '(" You hurt the " AN
D blo >= 3);(" You frightened th
e " AND blo <= 2);m$(mn): GO SUB
 4100
3070 LET n(mn)=n(mn)-blo
3080 IF n(mn)<1 THEN  GO TO 3110
3090 IF n(mn)<3 THEN  PRINT '" T
he ";m$(mn)( TO g(mn));" looks v
ery ill."
3100 GO TO 1050
3110 PAUSE 50: PRINT '" The ";m$
(mn)( TO g(mn));" runs away": GO
 SUB 4100: PRINT '" BUT your str
ength is now ";str
3120 LET hq=hq+g(mn)*( INT ( RND
 *3)+1): LET ded=ded+1
3130 GO TO 1100+100* INT ( RND *
2)
3500 CLS : PRINT  AT 1,12;"WARLO
CK"' TAB 12;"*******": RETURN
4000 FOR f=1 TO 11: BEEP z(f),y(
f): NEXT f: RETURN
4100 FOR f=10 TO 20: FOR h=0 TO
3: BORDER h: BEEP .016,f*2-h: NE
XT h: NEXT f: BORDER 0: RETURN
4500 FOR f=5 TO 6: BEEP .5,-6*f:
 NEXT f: PAUSE 50: RETURN
5000 GO SUB 3500
5010 PRINT  AT 15,6;"Instruction
s follow"
```

```
5020 GO SUB 5500: GO SUB 3500
5030 DIM m$(9,9): DIM a$(3,9)
5040 DIM m(9): DIM n(9): DIM g(9
)
5050 DIM k$(3,6): DIM z(11): DIM
 y(11)
5060 RESTORE : FOR f=1 TO 9: REA
D m$(f): READ m(f): READ n(f): R
EAD g(f): NEXT f
5070 FOR f=1 TO 3: READ a$(f): N
EXT f
5080 FOR f=1 TO 11: READ z(f): R
EAD y(f): NEXT f
5090 FOR f=1 TO 3: READ k$(f): N
EXT f
5100 PRINT '" You are a novice h
ero raiding"'' TAB 5;"the Warloc
k's Mansion."
5110 PRINT '" You hope to find h
is "; FLASH 1; PAPER 6;"TREASURE
."
5120 PRINT '" ......The Warlock
is away....."'' TAB 13; FLASH 1;
"BUT"; FLASH 0
5130 PRINT '" He has left his pe
ts guard it!"
5140 PRINT ''"BEWARE THE RETURN
OF THE WARLOCK"
5150 FOR f=1 TO 3: FOR i=4 TO 7:
 FOR h=2 TO 6: BEEP .01,f+i+h: B
ORDER h: BEEP .01,i+h-f: BORDER
i: BEEP .01,h+f-i: BORDER f: NEX
T h: NEXT i: NEXT f: BORDER 0
5160 LET hr=0: LET tr=0: LET ky=
0
5170 LET tres=1500+100* INT ( RN
D *10)
5180 LET fl=0: LET ded=0: LET hq
=0
5190 LET str=500: LET csh=0
5500 PRINT  AT 21,3;"Press any k
ey to CONTINUE": PAUSE 0: CLS :
RETURN
6000 PRINT '" Who knocks you ove
r and"'" takes some of your gol
d."
6010 LET csh=csh- INT (csh/( INT
 ( RND *3)+1))
6020 PRINT ''; INK 6;" You now h
ave ";csh;" gold coins."
6030 IF tr AND csh<tres THEN  LE
T tr=0: PRINT ''' TAB 3;"Start l
ooking again."
6050 GO TO 4500
6500 PRINT '" No!";: GO TO 8000
7500 DATA "Goblin",5,8,6,"Troll"
,3,10,5,"Demon",2,10,5,"Vampire"
,2,8,7,"Werewolf",4,9,8,"Python"
,3,4,6,"Ghost",2,8,5,"Dragon",4,
8,6,"WARLOCK",8,12,7
7600 DATA "Hallway.","Door.","St
airway."
7700 DATA 1,4,1,4,0.3,4,1.2,4,0.
75,7,0.5,6,1,6,0.3,4,0.7,4,0.5,3
,1,4
7800 DATA "rabbit","frog","flea"
8000 PRINT ".....Leave!": GO SUB
 4500
8200 IF  NOT  INT ( RND *5) THEN
  GO TO 1000
8210 RETURN
8500 PRINT '" Coward": LET hq=hq
-g(mn)*( INT ( RND *3)+1): IF hq
<1 THEN  LET hq=0
8510 GO TO 8520+10* INT ( RND *3
)
8520 PRINT '" Unfortunately, in
your haste"'" you trip and drop
 some gold.": GO TO 6010
8530 GO TO 4500
8540 PRINT '" Oh! Bad luck! He t
raps you.": GO SUB 4500: GO TO 1
050
9000 CLEAR : LET S$="WARLOCK"
9010 SAVE S$ LINE 0: BEEP 1,10
9020 PRINT "REWIND & PRESS ANY K
EY TO VERIFY "
9030 PAUSE 0
9040 VERIFY S$: BEEP 1,0
9050 STOP
```

```
WARLOCK
*******
THE BITTER END
**************
You met your End at the
 hands of the WARLOCK.
He turned you into a flea
Your Hero-rating WAS 65
You found 0 gold coins.
NOW HOP IT!
Highest Score:65
```

```
SUDDENLY!
You meet the Medusa
What do you do?
(1) Hit it!  (2) Run away!
You frightened the Medusa
The Medusa runs away
BUT your strength is now 398
```

Zipping and Zapping

Spice up your games with a few novel sound effects. Zip 'em and zap 'em!

Anyone who has tried to write their own games on the Spectrum will have found the BEEP command very limited and many people have bought add-ons to raise the Spectrum's sound to that of the BBC micro in versatility as well as volume. However, for most people £20-£30 is a high price for such a simple task.

Fortunately, there is a compromise solution which uses software rather than hardware to achieve special effect sounds such as explosions at negligible cost.

ZIP!
by David Mold

This is a m/c program which produces a much more interesting sound from the Spectrum speaker than its usual 'beep'. It can be called very easily from BASIC and the sound produced can be varied by using DEF FN. The program was written for 16K Spectrum and could easily be run on a 48K model, although the addresses should really be changed to put it higher in memory on the 48K version.

The listing of the m/c is in three columns: the first shows the address of the corresponding bytes, the second shows the decimal codes of those bytes and the third shows the Z80 opcodes.

It should be entered to the address 32500 using the short BASIC program.

The code can then be SAVEd using:

SAVE "beep" CODE
32500,32

and when it is required for use with a program — existing or to be written, it can be loaded from tape thus:

CLEAR 32499:LOAD "beep" CODE

and then used in conjunction with the program as described previously.

To call this routine from a program, first

DEF FN b(x,y)=USR 32500

then, when the beep is required use

LET 1=FN b(N,N)

where N,N represents two numbers you can specify which, very broadly, set the duration and pitch respectively. They should both be between 0 and 255 inclusive. Experiment to find the effect of varying the two factors. Also, try

POKE 32528,12

as a direct command, and experiment further to find its effect.

BASIC PROGRAM

```
10  CLEAR 32499
20  DATA 42,11,92,17,4,0,25,78,25,25,70,,58,72,92,203,47
30  DATA 203,47,203,47,238,16,211,254,81,21,32,-3,13,
    16,-11,201
40  FOR N=1 TO 32: READ a
50  POKE 32499+n,a: NEXT n
```

BEEP ROUTINE

```
32500   42,11,92    ld hl,(DEFADD)
32503   17,4,0      ld de,4
32506   25          add hl,de
32507   78          ld c,(hl)
32508   25          add hl,de
32509   25          add hl,de
32510   70          ld b,(hl)
32511   58,72,92    ld a,(BORDCR)
32514   203,47      sra a
32516   203,47      sra a
32518   203,47      sra a
*32520  238,16      xor 16
32522   211,254     out (254),a
32524   81          ld d,c
@32525  21          dec d
32526   32,-3       jrnz @
32528   13          dec c
32529   16,-11      djnz,*
32531   201         ret
```

ZAP!
by E. French

The second program contains three pre-defined sounds:

1) A Zap
2) A Machine gun shot
3) An Explosion.

As it stands the program occupies ¼ KByte which is normally reserved for the printer so as to conserve memory.

The program is in six parts: the first contains two subroutines which can produce white noise (for gunshots and explosions) or tones (for sirens and zaps). The second, third and fourth parts produce the predefined sounds. The two remaining routines play the user-defined sound: the first with white noise (a

kind of rushing sound) and the other with tones.

USER DEFINED SOUNDS

The BEEP command produces a neat regular regular waveform reminiscent of an electric organ which is not very useful to the games programmer.

Special effect sounds are not so simple and require more information in order to be produced. With a BEEP command it is necessary to provide only two figures or parameters: the first indicates how long you want the sound to last for and the second indicates the pitch of the resultant sound. A special effect sound can best be thought of as a succession of BEEPs one after another. However, special effect sounds cannot be produced by the BASIC BEEP command because the interval between individual BEEPs is too great.

This, thankfully, is not an insurmountable problem, because by using machine code routines the delay between BEEPs becomes neglible.

If sounds such as explosions are to be produced, successive BEEPs will not suffice. Such sounds are produced with white noise which on its own gives a rushing noise.

Now to the practicalities: *There is no need for the user to know anything about machine code.* All that is required is to follow these instructions:

1. Type in listing one.
2. RUN it.
3. Type SAVE "Sounds" CODE 23926,256 and enter.
4. Rewind the tape and VERIFY""CODE and enter.
5. If you get the message "Tape loading error" then go to stage 3.
6. The BASIC program has now done its job and it is safe to NEW. To recover the machine code routine use: LOAD""CODE.

If you have an assembler you should type in listing 2, assemble it and SAVE the code.

RUN OLD RAY GUN

The next thing you will want to do is listen to your new sounds. To do this simply enter the following:

```
10  PAUSE 0: REM waits for
    keypress
20  LET (an unused variable)
    =USR 23375; REM make a
    ZAP
30  GO TO 10: REM do it again
```

When this is RUN it turns your Spectrum into a ray gun. By changing line 20 to:

```
20  LET (an unused variable)=
    USR 23386
```

Now your Spectrum is a machine gun. And making line 20 read:

```
20  LET (an unused variable)=
    USR 23398
```

Now whenever you press a key your Spectrum happily explodes!

So to sum up:

```
LET  (an unused variable)=USR
     23375 produces a Zap.
LET  (an unused variable)=USR
     23386 produces a machine
     gun shot.
LET  (an unused variable)=USR
     23398 produces an explo-
     sion!
```

YOUR OWN SOUNDS

The more skilfull will want more than just these three and impressive sounds can be custom made. However, take great care: if you go wrong on this section you might have to wait several days before your machine has finished your sound.

So, to produce your own effects you must tell the computer the length and pitch of the eight beeps which make up the sound you want to produce. You must also tell it how many times to play each beep. Thus, for each beep there are three parameters: length, pitch and the number of times you want each beep to be produced. So in total there are 24 parameters. The procedure to tell these to the computer is:

```
10  DATA (length of 1st),(pitch
    of 1st),(No. of times),(length
    of 2nd),(pitch of 2nd),(No.of
    times)....up to....(length
    of 8th),(pitch of 8th),(No.of
    times)
20  FOR a=23300 TO 23323:
    READ d:POKE a,d: NEXT a
```

The length of each beep is a number from 1 to 255 and the longest note is 0. The pitch of each beep is a number from 1 to 255 where 1 is the highest pitch (but 0 is the lowest). Similarly the multiplier (how many times) goes from 1 to 255 (again 0 gives 256 cycles).

This program as it stands resides in the printer buffer which for most people is unused. However if you enter any of the printer commands: LLIST, LPRINT or COPY then the sound will generate a delay followed by NEW, so take care.

Once you have provided the machine with the data you will need to know how to play the sounds. For this there are two commands:

```
LET  (an unused variable)=USR
     23410 this will play your
     sound with tones
LET  (an unused variable)=USR
     23432 this will play your
     sound with white noise.
```

Another slight snag encountered with this routine is that when the sounds are produced the border goes white. If you use an assembler to write in this programm you can change it to suit your needs.

HINTS

It is a wise idea to make the sounds last for as short as possible because it is then less obvious that the program stops to produce the sound. Also, the volume of the sounds is increased by resting the machine on a hard flat surface such as a desk or table.

LISTING ONE (BASIC)

```
10  DATA 197,213,67,16,254,10,203,199,203,207,203,215,
    211,254,12,21,32,240,209,193,201
20  DATA 197,213,175,203,199,203,207,203,215,67,16,
    254,203,231,211,254,67,16,254,203,167,211,254,21,
    32,239,209,193,201
30  DATA 22,3,30,128,205,50,91,29,32,250,201
40  DATA 30,0,22,32,205,29,91,28,21,32,249,201
50  DATA 30,0,22,128,205,29,91,28,21,32,249,201
60  DATA 33,3,91,35,86,35,94,35,78,35,205,29,91,13,32,
    250,125,254,28,56,239,201
70  DATA 33,3,91,35,86,35,94,35,78,35,205,50,91,13,32,
    250,125,254,28,56,239,201
80  FOR a=23325 to 23453:READ d:POKE a,d:NEXT a
```

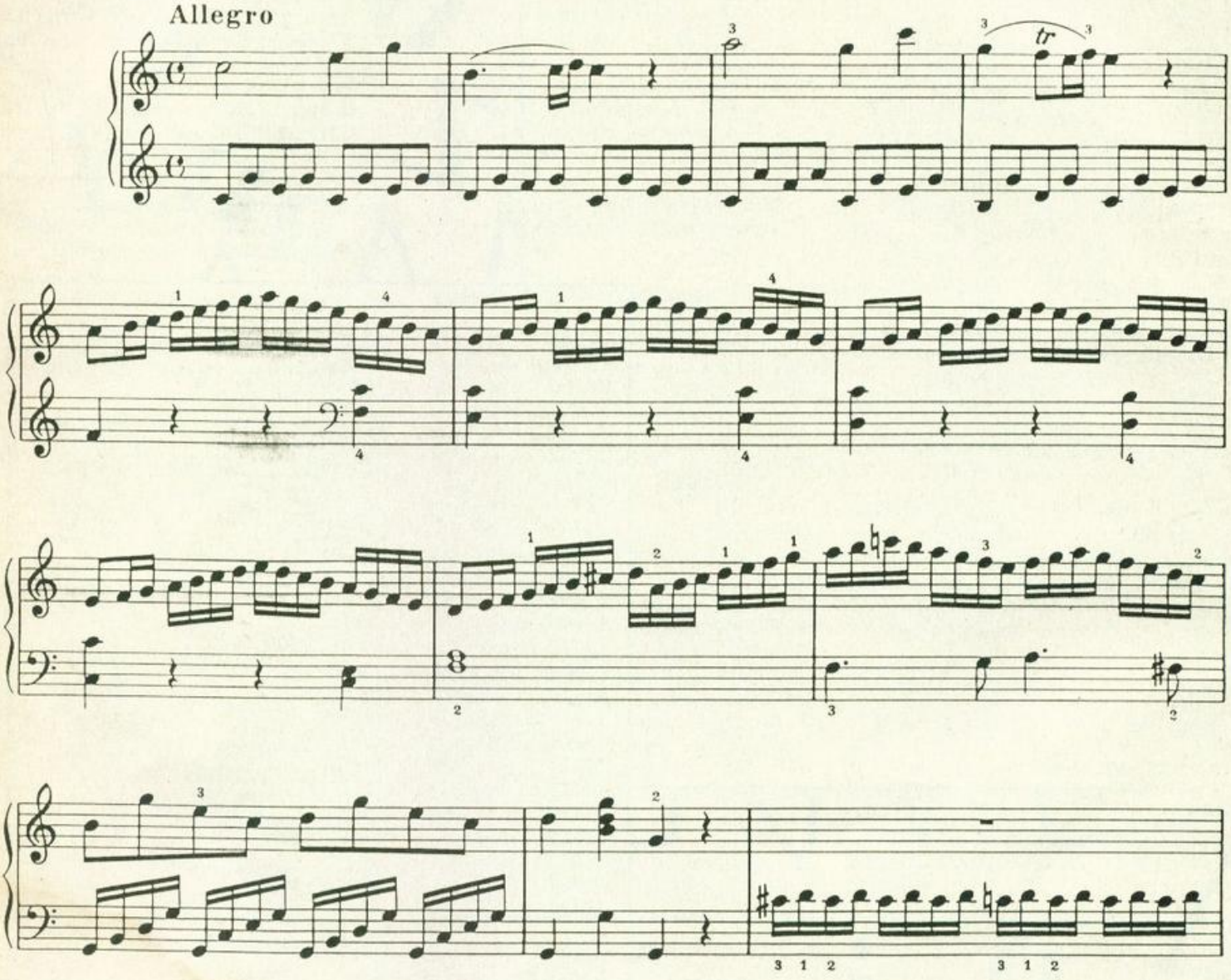
LISTING FOR USE WITH AN ASSEMBLER

ADDRESS	NMONICS	DECIMAL CODES
23325	push bc	197
23326	push de	213
23327	ld b,e	67
23328	djnz — 2	16,254
23330	ld a,(bc)	10
23331	set 0,a	203,199
23333	set 1,a	203,207
23335	set 2,a	203,215
23337	out(254),a	211,254
23339	inc c	12
23340	dec d	21
23341	jr nz — 16	32,240
23343	pop de	209
23344	pop bc	193
23345	ret	201
23346	push bc	197
23347	push de	213
23348	xor a	175
23349	set 0,a	203,199
23351	set 1,a	203,207
23353	set 2,a	203,215
23355	ld b,e	67
23356	djnz — 2	16,254
23358	set 4,a	203,231
23360	out(254),a	211,254
23362	lb b,e	67
23363	djnz — 2	16,254
23365	res 4,a	203,167
23367	out(254),a	211,254
23369	dec d	21
23370	jr nz — 17	32,239
23372	pop de	209
23373	pop bc	193
23374	ret	201
23375	ld d,3	22,3
23377	ld e,128	30,128
23379	call 23346	205,50,91
23382	dec e	29
23383	jr nz — 6	32,250
23385	ret	201
23386	ld e,0	30,0
23388	ld d,32	22,32
23390	call 23326	205,29,91
23393	inc e	28
23394	dec d	21
23395	jr nz — 7	32,249
23397	ret	201
23398	ld e,0	30,0
23400	ld d,128	22,128
23402	call 23326	205,29,91
23405	inc e	28
23406	dec d	21
23407	jr nz — 7	32,249
23409	ret	201
23410	ld hl,23299	33,3,91
23413	inc hl	35
23414	ld d,(hl)	86
23415	inc hl	35
23416	ld e,(hl)	94
23417	inc hl	35
23418	ld c,(hl)	78
23419	inc hl	35
23420	call 23326	205,91
23423	dec c	13
23424	jr nz — 6	32,250
23426	ld a,l	125
23427	cp 28	254,28
23429	jr c — 17	56,239
23431	ret	201
23432	ld hl,23299	33,3,91
23435	inc hl	35
23436	ld d,(hl)	86
23437	inc hl	35
23438	ld e,(hl)	94
23439	inc hl	35
23440	ld c,(hl)	78
23441	inc hl	35
23442	call 23346	205,50,91
23445	dec c	13
23446	jr nz — 6	32,250
23448	ld a,l	125
23449	cp 28	254,28
23451	jr c — 17	56,239
23453	ret	201

CHI-YEUNG CHOY

From beep to Mozart

And you thought all your Spectrum could do was beep quietly at you! Think again, music lovers.



The Spectrum's oft maligned beep has been considered too feeble for any reasonable musical application in the past, but now we present a program which will make you think again!

Instead of the usual selection of sound effects or brief burst of barely recognisable tunes, Chi-Yeung has successfully programmed a computer version of the first movement of Mozart's piano sonata in C Major K545. What is even more amazing is that it is written all in BASIC and fits into the 16K machine!

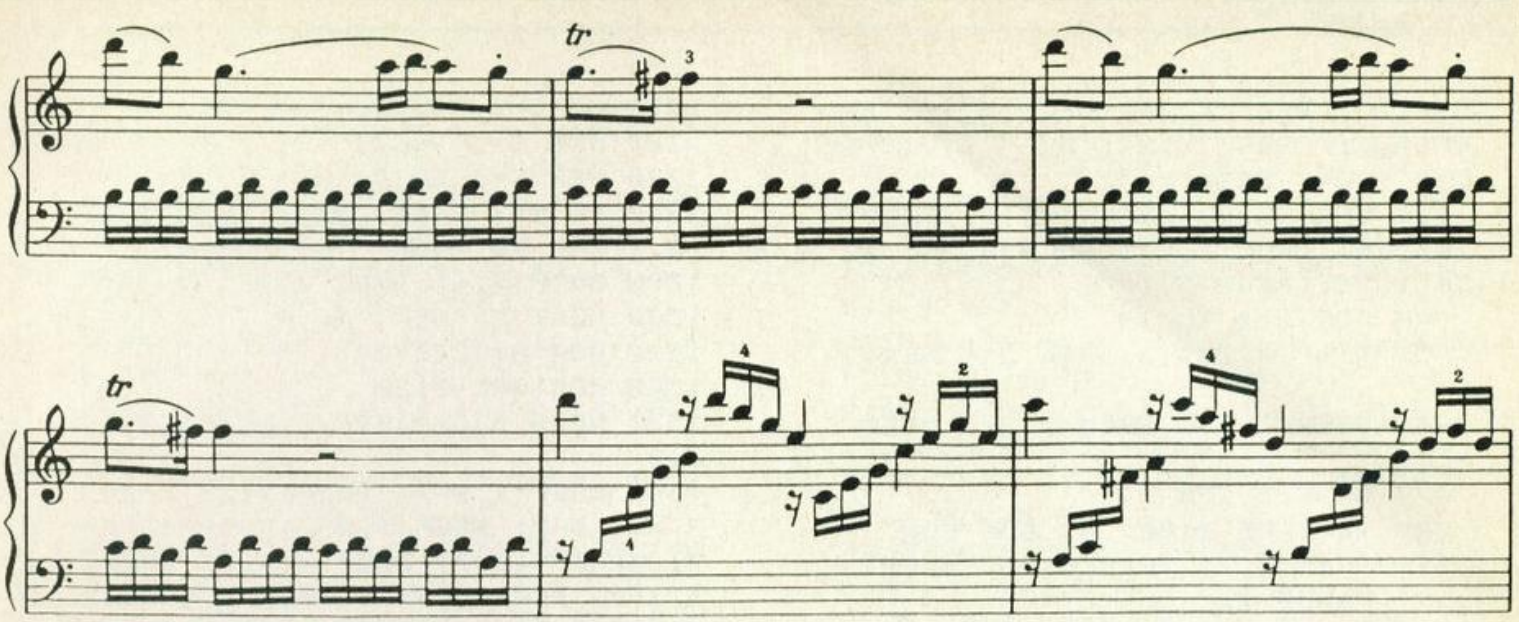
ANDANTE

A great deal of thought and attention to technical detail has gone into this program for instance, the subroutines have been put at the start of the program to get every ounce of speed from the computer.

Chi-Yeung tells us that his greatest problem was fitting it into the 16K due to the amount of DATA required and that he almost gave up. Type it in and then give your fingers a rest and your ears a treat. Play on maestro!

HOW IT RUNS

Lines 30-230	*Subroutines.*
Lines 240-520	*DATA statements containing pitch of notes.*
Line 1000	*Set up variables for note duration.*
Lines 1010-2070	*Main program consisting of FOR-NEXT loops.*




```
   Ø>REM ************************
         *         SONATA       *
         *----------------------*
         *  C-Y Choy    1983    *
         ************************

  1Ø GO TO 5ØØ

  2Ø REM ************************
  3Ø REM SUBROUTINES
  4Ø REM ************************
  5Ø READ B,C,D
  6Ø BEEP 2*X,B: BEEP X,C: BEEP
X,D: RETURN
  7Ø FOR M=1 TO 16
  8Ø READ B: BEEP Z,B
  9Ø NEXT M: RETURN
 1ØØ FOR N=1 TO 8
 11Ø READ B: BEEP Y,B
 12Ø NEXT N
 13Ø BEEP X,14: BEEP X,19: BEEP
X,7: PAUSE 25
 14Ø RETURN
 15Ø READ B,C,D,E,F
 16Ø BEEP 2*X,B: BEEP Z,C: BEEP
Y+Z,D: BEEP Z,E: BEEP Y+Z,F: RET
URN
 17Ø FOR N=1 TO 12
 18Ø READ B: BEEP Z,B
 19Ø NEXT N: RETURN
 2ØØ BEEP X,19
 21Ø FOR N=1 TO 12
 22Ø READ B: BEEP Z,B-12
 23Ø NEXT N: RETURN
 24Ø READ B,C,D
 25Ø BEEP 2*X,B+5: BEEP X,C+5: B
EEP X,D+5: RETURN

 5ØØ REM ************************
 51Ø REM MAIN PROGRAM
 52Ø REM ************************
 53Ø LET X=1/2: LET Y=X/2: LET Z
=X/4: LET A=X/8
 54Ø GO SUB 5Ø
 55Ø READ B,C,D,E
 56Ø BEEP X+Y,B: BEEP Z,C: BEEP
Z,D: BEEP X,E: PAUSE 25
 57Ø GO SUB 5Ø
 58Ø BEEP X,19
 59Ø FOR N=1 TO 3
 6ØØ BEEP A,19: BEEP A,17: NEXT
N
 61Ø BEEP A,16: BEEP A,17: BEEP
X,16: PAUSE 25
 62Ø FOR N=1 TO 5
 63Ø READ B: BEEP Y,B
 64Ø FOR M=1 TO 14
 65Ø READ B: BEEP Z,B: NEXT M: N
EXT N
 66Ø GO SUB 7Ø
 67Ø GO SUB 1ØØ
 68Ø FOR N=1 TO 2
 69Ø FOR M=1 TO 4
 7ØØ IF N=1 THEN  BEEP Z,1: BEEP
 Z,2: GO TO 72Ø
 71Ø BEEP Z,Ø: BEEP Z,2
 72Ø NEXT M: NEXT N
 73Ø RESTORE 212Ø
 74Ø FOR N=1 TO 2
 75Ø RESTORE 212Ø
 76Ø READ B,C,D,E,F,G,H,I,J,K,L,
M,P
 77Ø BEEP Y,B: BEEP Y,C: BEEP X+
Y,D: BEEP Z,E: BEEP Z,F: BEEP Y,
G: BEEP Y,H: BEEP A,I: BEEP A,J:
 BEEP A,K: BEEP A+Z,L: BEEP Z,M:
 BEEP X,P: PAUSE 5Ø
 78Ø NEXT N
 79Ø FOR N=1 TO 4
 8ØØ GO SUB 7Ø: NEXT N
 81Ø GO SUB 15Ø
 82Ø BEEP Z,2Ø: BEEP X+Z,21: BEE
```

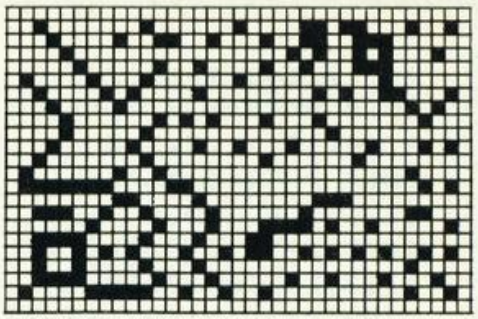
```
P A,23: BEEP A,21: BEEP A,20: BE
EP A,21: BEEP Y,24: BEEP Y,21: B
EEP Y,24: BEEP Y,21
 830 BEEP Y,23: BEEP Y,19: BEEP
2*X,26: BEEP Z,24: BEEP Z,23: BE
EP Z,21: BEEP Z,19
 840 FOR N=1 TO 15
 850 BEEP A,23: BEEP A,21: NEXT
N
 860 BEEP Z,19: BEEP Z,21: BEEP
X,19
 870 GO SUB 170
 880 RESTORE 2180: GO SUB 200
 890 BEEP X,7: BEEP X,23: BEEP X
,19: PAUSE 25
 900 BEEP X,7: GO SUB 170
 910 RESTORE 2190: GO SUB 200
 920 FOR N=1 TO 2
 930 GO SUB 70: NEXT N
 940 BEEP X,17
 950 RESTORE 2190
 960 FOR N=1 TO 12
 970 READ B: BEEP Z,B-5: NEXT N
 980 BEEP X,14
 990 RESTORE 2190
1000 FOR N=1 TO 12
1010 READ B: BEEP Z,B-17: NEXT N
1020 RESTORE 2220
1030 FOR N=1 TO 7
1040 GO SUB 70: NEXT N
1050 RESTORE 2030: GO SUB 240
1060 READ B,C,D,E
1070 BEEP X+Y,B+5: BEEP Z,C+5: B
EEP Z,D+5: BEEP X,E+5: PAUSE 25
1080 GO SUB 240
1090 BEEP X,24
1100 FOR N=1 TO 3
1110 BEEP A,24: BEEP A,22: NEXT
N
1120 BEEP A,21: BEEP A,22: BEEP
X,21: PAUSE 25
1130 FOR N=1 TO 4
1140 READ B: BEEP Y,B+5
1150 FOR M=1 TO 14
1160 READ B: BEEP Z,B+5
1170 NEXT M: NEXT N
1180 BEEP 2*X,21: PAUSE 25: BEEP
 X,21
1190 BEEP 2*X,19: PAUSE 25: BEEP
 X,19
1200 BEEP 2*X,17: PAUSE 25: BEEP
 X,17
1210 BEEP 2*X,16: PAUSE 25: BEEP
 X,16
1220 RESTORE 2300
1230 GO SUB 70
1240 RESTORE 2100
1250 GO SUB 70: GO SUB 100
1260 FOR N=1 TO 2
1270 FOR M=1 TO 4
1280 IF N=2 THEN  BEEP Z,5: BEEP
 Z,7: GO TO 1300
1290 BEEP Z,6: BEEP Z,7
1300 NEXT M: NEXT N
1310 FOR N=1 TO 2
1320 RESTORE 2120
1330 READ B,C,D,E,F,G,H,I,J,K,L,
M,P
1340 BEEP Y,B-7: BEEP Y,C-7: BEE
P X+Y,D-7: BEEP Z,E-7: BEEP Z,F-
7: BEEP Y,G-7: BEEP Y,H-7: BEEP
A,I-7: BEEP A,J-7: BEEP A,K-7: B
EEP A+Z,L-7: BEEP Z,M-7: BEEP X,
P-7: PAUSE 50
1350 NEXT N
1360 FOR N=1 TO 2
1370 FOR M=1 TO 16
1380 READ B: BEEP Z,B-7
1390 NEXT M: NEXT N
1400 FOR N=1 TO 2
1410 FOR M=1 TO 16
1420 READ B: BEEP Z,B+5
1430 NEXT M: NEXT N
1440 RESTORE 2290
1450 GO SUB 150
1460 BEEP 2*X,21: BEEP Z,20: BEE
P Y+Z,21: BEEP Z,20: BEEP Y+Z,21
1470 BEEP Y,19
1480 RESTORE 2310
1490 FOR N=1 TO 14
1500 READ B: BEEP Z,B: NEXT N
1510 FOR N=1 TO 15
1520 BEEP A,16: BEEP A,14: NEXT
N
1530 BEEP Z,12: BEEP Z,14: BEEP
X,12
1540 RESTORE 2180
1550 FOR N=1 TO 12
1560 READ B: BEEP Z,B-7: NEXT N
1570 BEEP X,12
1580 RESTORE 2180
1590 FOR N=1 TO 12
1600 READ B: BEEP Z,B-19: NEXT N
1610 BEEP X,0: BEEP X,16: BEEP X
,12: PAUSE 25

2000 REM ************************
2010 REM DATA
2020 REM ************************
2030 DATA 12,16,19,11,12,14,12
2040 DATA 21,19,24
2050 DATA 9,11,12,14,16,17,19,21
,19,17,16,14,12,11,9
2060 DATA 7,9,11,12,14,16,17,19,
```

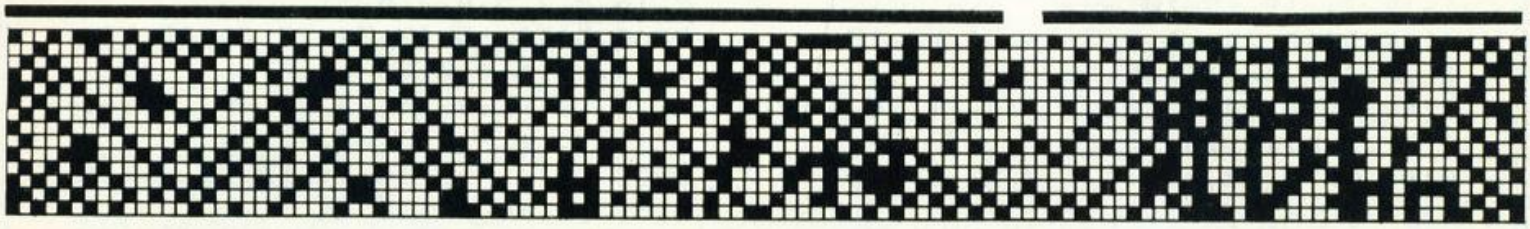
```
17,16,14,12,11,9,7
2070 DATA 5,7,9,11,12,14,16,17,1
6,14,12,11,9,7,5
2080 DATA 4,5,7,9,11,12,14,16,14
,12,11,9,7,5,4
2090 DATA 2,4,5,7,9,11,13,14,9,1
1,13,14,16,17,19
2100 DATA 21,23,24,23,21,19,17,1
6,17,19,21,19,17,16,14,12
2110 DATA 11,19,16,12,14,19,16,1
2
2120 DATA 26,23,19,21,23,21,19,2
1,19,21,19,18,18
2130 DATA 26,-1,2,7,11,26,23,19,
16,0,4,7,12,16,19,16
2140 DATA 24,-3,0,6,9,24,21,18,1
4,-1,2,6,11,14,18,14
2150 DATA 23,-5,-1,4,7,23,19,16,
12,-3,0,4,9,12,16,12
2160 DATA 21,-6,-3,2,6,21,18,14,
11,-5,-1,2,7,19,14,11
2170 DATA 9,11,12,15,16
2180 DATA 19,14,19,23,26,23,19,2
3,24,21,18,21
2190 DATA 19,14,19,22,26,22,19,2
2,24,21,18,21
2200 DATA 7,-17,-15,-14,-12,-10,
-8,-6,-5,19,22,21,19,17,16,14
2210 DATA 13,-15,-13,-11,-10,-8,
-6,-4,-3,25,28,26,25,22,21,19
2220 DATA -7,2,4,5,7,9,11,13,14,
2,5,4,2,0,-1,-3
2230 DATA -4,11,12,14,16,18,20,2
1,23,-4,-1,-3,-4,-7,-8,-10
2240 DATA -12,21,28,26,24,23,21,
19,17,2,9,7,5,4,2,0
2250 DATA -1,19,26,24,23,21,19,1
7,16,0,7,5,4,2,0,-1
2260 DATA -3,17,24,23,21,19,17,1
6,14,-1,5,4,2,0,-1,-3
2270 DATA -4,16,23,21,20,17,16,1
4,12,-3,0,-1,-3,-5,-7,-8
2280 DATA -10,10,14,12,10,9,7,5,
4,5,7,9,10,12,14,16
2290 DATA 14,13,14,13,14
2300 DATA 14,2,4,5,7,9,11,13,14,
9,11,13,14,16,17,19
2310 DATA 21,23,24,26,28,26,24,2
3,21,19,17,16,14,12
```

TONY HOULTON

Transforming graphic characters



Speed up the creation of characters by letting your computer do all the hard work



Most home computer programmers find that at some stage there is a need to use larger graphics characters built up from a number of individual user defined graphic characters. We may also need to use the same large character, but facing in a different direction. This program will considerably reduce the time and effort needed in calculating the required POKE values.

To illustrate the use of the program we will go through the steps needed to create a 2x2 character representing a small aeroplane which can be displayed on the screen facing in four different directions. This involves calculating the DATA for 16 user defined characters; a total of 128 POKE values. Using the program we only need to define two characters with a total of 16 POKE values.

The computer will calculate the rest.

MAKING IT PLAIN

In Fig. 1 you will see half the outline of the aeroplane and the DATA values which must be calculated to generate these characters. These values have been included in the program as lines 9000 and 9010. Also by entering GOTO 8970 in the direct addressing mode, these two characters will be POKEd into the UDG areas 'A' and 'B'.

Since the full character is symmetrical about its longitudinal axis, a reflection in the vertical plane (i.e. the right hand edge of each character) will produce the two characters needed to complete the character. To do this RUN the program and select option 1 from the menu.

When asked for the line number, enter 9000. The old values (i.e. line 9000) will be displayed together with the new DATA values. You will now be asked to choose which graphics character you would like the values

VARIABLES USED

A	*number of DATA line from which values of bytes are to be read*
F	*array to hold new DATA values*
M	*array to hold old DATA values*
B,C,D,E,	*transient variables used to hold values during calculations and whose values may change during a calculation*
G	*value of first address in memory of current UDG character*
H	*variable having a value between 0 and 20 used as a pointer to the chosen UDG character. Value is obtained from ASCII number of letter chosen*
I,J,K,L,	*counters used in FOR...TO...NEXT loops*

HOW IT RUNS

7000-7110	*menu display and option selection*
7120-7220	*vertical reflection routine. Each pixel in each row is read, if used, an appropriate value is added to the row total and the final value stored*
7230-7240	*prints old and new DATA values to screen*
7250-7330	*offers UDG character choice and POKEs values into chosen area of memory*
7340-7400	*offers printer, STOP and return to menu options (note that lines 7230 to 7400 serve all the menu options)*
7410-7450	*horizonal reflection or inversion routine. The values for each row of pixels are re-ordered. Value 1 becomes value 8, value 2 becomes value 7, and so on.*
7460-7590	*rotation through 90 degrees routine. The first pixel in each row is read; if used an appropriate value is added to the variable C; the final value of C then becomes the value of row 1. The process is repeated for the second pixel in each row to obtain a value for row 2 and so on till all eight values have been found*

POKEd into. Press 'C' and then take the STOP option when it is offered. Using the DATA still display on the screen, enter into the program 9020 DATA (the values shown without the final comma). Repeat the process again, but this time using initial line number 9010, character 'D' and program data line 9030.

COMBINATIONS

You have now added two lines of data and should have four characters which can be combined to give the shape shown in Fig. 2.

The procedure is the same for menu options 2 and 3, while option 4 allows you to escape from the program.

To invert the aeroplane RUN, choose option 2, repeat procedure four times using lines 9000, 9010, 9020, and 9030 as initial lines; characters 'E', 'F', 'G' and 'H'; and storing the data in program lines 9040, 9050, 9060 and 9070. The four latest characters can now be combined to produce an aeroplane pointing down the screen.

NEVER FORGET YOU HAVE A CHOICE

Finally by RUNing, choosing option 3, using lines 9000 to 9070 as initial lines, POKEing characters 'I' to 'P' and storing the data in program lines 9080 to 9150, you will have the remaining eight characters needed to display the areoplane facing left and right.

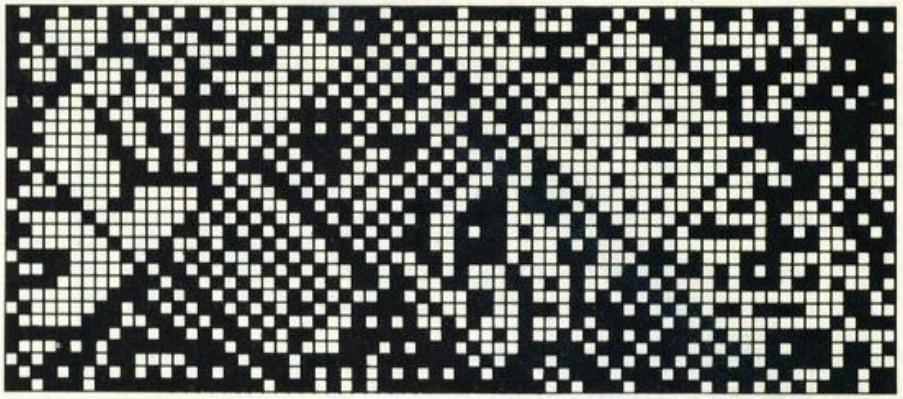
The whole process takes much longer to describe than to perform and half an hour's practice and experimentation should put you in command of a most useful programming aid.

The line numbering has been chosen to occupy an area which will not often be used when writing your own programs. This means that the program can be loaded before you start to write. You should replace lines 9000 and 9010 with your own data.

If you don't want to delete the program when you have written your own program it is advisable to include a line such as _6999 STOP or 6999 GOTO (the last line number of your program).

An option to record the old and new DATA by means of a printer is also included.

The program can also be used to transform nonsymmetric characters, but you may have to combine a number of reflections and rotations (holding the DATA in intermediate line numbers) before the desired orientation is obtained.



```
7000 REM GRAPHIC CHARACTER                    TRANSFORM
ATION                    ON THE SPECTRUM
     BY R.A.HOULTON                          NOVEMBER 1984
7010 CLS : PRINT "    WHICH TRANSFORMATION DO YOU
REQUIRE? (PRESS THE APPROPRIATE NUMBER KEY)"
7020 PRINT : PRINT "1. REFLECTION IN THE VERTICAL
     PLANE."
7030 PRINT : PRINT "2. REFLECTION IN THE HORIZONTA
L    PLANE."
7040 PRINT : PRINT "3. ROTATION THROUGH 90 DEGREES
     CLOCKWISE."
7050 PRINT : PRINT "4. ESCAPE FROM PROGRAMME."
7060 IF INKEY$="" THEN  GO TO 7060
7070 IF INKEY$="1" THEN  GO TO 7120
7080 IF INKEY$="2" THEN  GO TO 7410
7090 IF INKEY$="3" THEN  GO TO 7460
7100 IF INKEY$="4" THEN  STOP
7110 GO TO 7060
7120 REM VERTICAL REFLECTION                  LEFT T
O RIGHT
7130 INPUT "ENTER LINE NUMBER OF DATA TO BE TRANSF
ORMED ";A
7140 RESTORE A: CLS : DIM F(8): DIM M(8)
7150 FOR J=1 TO 8
7160 READ B: LET M(J)=B: LET C=128: LET D=1: LET E
=0
7170 FOR K=1 TO 8
7180 IF B>=C THEN  LET B=B-C: LET E=E+D
7190 LET C=C/2: LET D=D*2
7200 NEXT K
7210 LET F(J)=E
7220 NEXT J
7230 PRINT "OLD DATA ";: FOR L=1 TO 8: PRINT M(L);
",";: NEXT L
7240 PRINT : PRINT : PRINT "NEW DATA ";: FOR L=1 T
O 8: PRINT F(L);",";: NEXT L
7250 PRINT AT 15,0;"INTO WHICH USER DEFINED GRAPHI
C CHARACTER DO YOU WANT TO POKE   THESE NEW VALUES
?(PRESS A LETTERBETWEEN ""A"" AND ""U"")"
7260 IF INKEY$="" THEN  GO TO 7260
7270 IF CODE INKEY$<65 OR CODE INKEY$>85 AND CODE
INKEY$<97 OR CODE INKEY$>117 THEN  GO TO 7260
7280 IF CODE INKEY$>64 AND CODE INKEY$<86 THEN  LE
T H=CODE INKEY$-65
7290 IF CODE INKEY$>96 AND CODE INKEY$<118 THEN  L
ET H=CODE INKEY$-97
7300 LET G=65368+(H*8)
7310 FOR J=1 TO 8
7320 POKE G,F(J): LET G=G+1
7330 NEXT J
7340 PRINT AT 19,0;"TO COPY TO PRINTER PRESS ""P""
 TO STOP PRESS ""S"" ANY OTHER KEY TO RETURN TO ME
NU."
```

```
7350 IF INKEY$="" THEN  GO TO 7350
7360 IF INKEY$="P" OR INKEY$="p" THEN  GO TO 7390
7370 IF INKEY$="S" OR INKEY$="s" THEN  STOP
7380 GO TO 7010
7390 LPRINT "OLD DATA ";: FOR L=1 TO 8: LPRINT M(L
);",";: NEXT L
7400 LPRINT : LPRINT : LPRINT "NEW DATA ";: FOR L=
1 TO 8: LPRINT F(L);",";: NEXT L: LPRINT : GO TO 7
340
7410 REM HORIZONTAL REFLECTION
7420 CLS : INPUT "ENTER LINE NUMBER OF DATA TO BE
TRANSFORMED ";A
7430 RESTORE A: DIM F(8): DIM M(8)
7440 FOR I=1 TO 8: READ B: LET M(I)=B: LET F(9-I)=
B: NEXT I
7450 GO TO 7230
7460 REM 90 DEGREES CLOCKWISE                ROTATION
7470 CLS : INPUT "ENTER LINE NUMBER OF DATA TO BE
TRANSFORMED ";A
7480 RESTORE A: DIM B(8): DIM F(8): DIM M(8)
7490 FOR I=1 TO 8: READ D: LET B(I)=D: LET M(I)=D:
 NEXT I
7500 LET E=128
7510 FOR I=1 TO 8
7520 LET C=0: LET G=1
7530 FOR J=1 TO 8
7540 IF B(J)>=E THEN  LET C=C+G: LET B(J)=B(J)-E
7550 LET G=G*2
7560 NEXT J
7570 LET F(I)=C: LET E=E/2
7580 NEXT I
7590 GO TO 7230
8970 FOR I=USR "A" TO USR "A"+15
8980 READ B: POKE I,B
8990 NEXT I
9000 DATA 1,1,255,255,127,15,1,1
9010 DATA 1,1,1,15,7,0,0,0
```

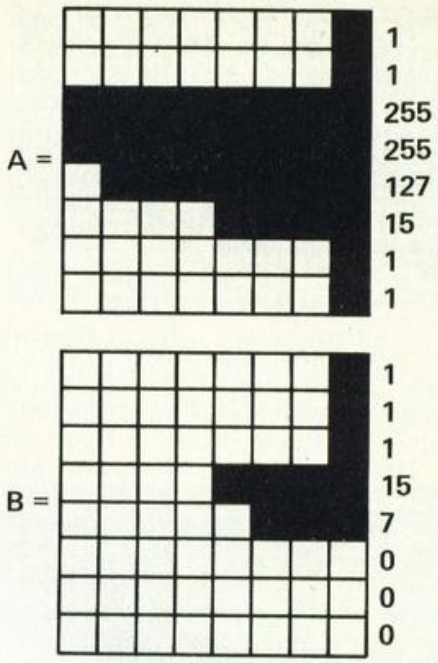


Fig 1.

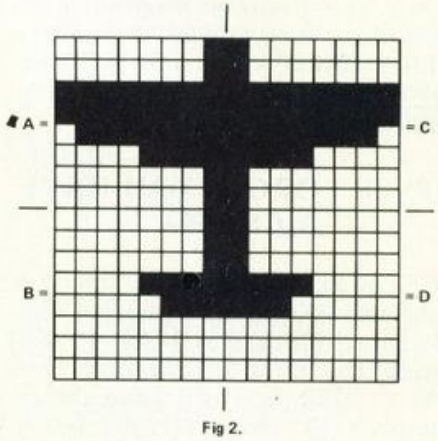


Fig 2.

TONY HOULTON

Compose Yourselves

Write your own music with this program or use it to convert Dragon listings.

This program simulates the Dragon PLAY command by allowing you to enter a string of letters and numerals which are then translated into the Spectrum BEEP commands.

The main variables are as follows: T tempo, O octave, L note length and P pause length. Each let ter is followed by a number, if you don t include them then the follow ing default values are set by the computer:

T2 – roughly equivalent to 4/4 time; O2 – the octave below mid dle C; L4 – gives a note length equivalent to a ¼ note duration. No default value is set for P, but the pauses have the same duration as the note lengths e.g. P8 gives a pause equivalent to the duration of an eighth note. The notes are entered by using the letters A to G.

You can also modify notes by using the relevant symbols for sharps, flats and dotted notes etc.

MUSIC TO CODE

The first part of the program describes the variables and their parameters and shows the normal musical notation and the program equivalents. It should be possible, with a little practice, to translate written music into the program code. The computer translates this into Spectrum BEEPs and displays these on the screen.

As each BEEP is calculated the note is sounded and the pitch and duration values displayed. The tune is played at normal speed and the in formation redisplayed. You can then replay the tune as many times as you wish without losing the informa tion. When you have copied the BEEP values for future use you can return to the menu to enter more music. You can have a printed copy by changing any PRINT commands in lines 1240 to 1280 to LPRINT. If you want a copy of your original str ing entry, then break into the pro gram (CAPS SHIFT and BREAK) and enter the direct addressing mode LPRINT or PRINT A$.

CONVERSIONS

To convert PLAY commands in Dragon listings you select the MUSICMAKER option and enter the strings in the listings. The only variables that are not found in this program will be the letter V follow ed by a number. This controls the volume. There is no volume control on the Spectrum and so these are ig nored.

The other sound command on the Dragon is SOUND. This plays a given note for a given duration and a short program (line 1340 onwards) to deal with this conversion has been included.

10 SECOND LIMIT

The only difficulty that will be en countered is that the Spectrum will not BEEP for longer than 10 seconds. If the second number after SOUND is larger than 160 then sub tract 160 from the value and enter as two separate values. SOUND 90,192 could be entered as SOUND 90,160: SOUND 90,132 translated as BEEP 10,0: BEEP 2,0. If you want to speed the program up by leaving out the BEEPs during the calculation then omit line 1070.

HOW IT RUNS

10-15	*put CAPS LOCK on, directs to graphics routine*
30-50	*MUSICMAKER or SOUND selection routine*
70-225	*menu of instructions, selection routine*
230-355	*details of variables and parameters*
360	*input point for string of variables*
370-440	*set default values for T,O,L*
450-580	*main loop to read input, allocate items in A$ to subroutines*
590-700	*calculation of tempo*
710-810	*calculation of octave start values*
820-930	*calculation of note length*
940-1020	*allocation of relative pitch values to note letters*
1030-1090	*calculation of pitch and duration; first print and sound*
1100-1190	*calculation of pause length*
1200-1230	*play tune at correct speed*
1240-1280	*list Spectrum BEEP values*
1290-1300	*play and list again or return to menu*
1340-1360	*instructions for Dragon SOUND conversion*
1370-1380	*input of Dragon SOUND values*
1390-1400	*conversion calculations*
1410-1420	*display Spectrum values and play sound*
1430-1460	*options for another SOUND conversion or return to menu*
1470-1490	*graphics POKE routine*
1500-1660	*DATA values for graphics characters*

VARIABLES USED

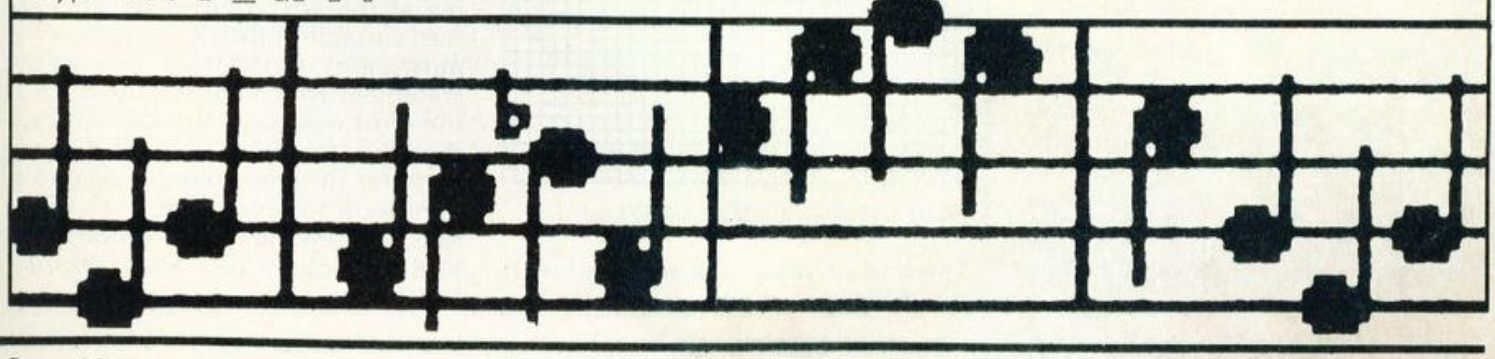
A$	*string holding input variables*
T	*tempo*
O	*octave*
L	*note length*
P	*pause length*
A to G	*note pitch values*
H(K)	*number array holding pitch values*
J(K)	*number array holding values*
K	*note counter*

```
                                                    "  10 CLS : POKE 23658,8: PRI
NT "  MUSICMAKER AND DRAGON SOUND      ****************************      CONVERSIO
N BY R.A.HOULTON       ************************                NOVEMBER 1984
            ************"
  15 GO SUB 1470
  20 PRINT : PRINT "     THE MUSICMAKER PROGRAMME IS SIMILAR IN FORMAT TO THE DRA
GON PLAY COMMAND."
  30 PRINT : PRINT "     A SHORT PROGRAMME TO CONVERTTHE DRAGON SOUND COMMAND FOR
 USEON THE SPECTRUM IS ALSO INCLUDED"
  40 PRINT : PRINT "PRESS ""M"" FOR MUSICMAKER OR ""S""     FOR DRAGON SOUND PROG
RAMME"
  50 IF INKEY$="" THEN  GO TO 50
  60 IF INKEY$="S" THEN  GO TO 1350
  70 CLS : PRINT "THIS PROGRAM ALLOWS YOU TO COPY OR COMPOSE MUSIC FOR YOUR 16K
 OR48K SPECTRUM USING ALPHABETIC   AND NUMERIC TERMS FOR THE NOTES,NOTE LENGTH, T
EMPO, OCTAVE AND  PAUSES ETC.": PRINT : PRINT "TO SEE THE VARIABLES AND THEIR  P
ARAMETERS PRESS Z, ANY OTHER   KEY TO WRITE MUSIC"
  80 IF INKEY$="" THEN  GO TO 80
  90 IF INKEY$<>"Z" THEN  GO TO 360
 140 CLS : PRINT TAB 4;"VARIABLES AND PARAMETERS": PRINT TAB 4;"**************
********": PRINT : PRINT "1. TEMPO": PRINT : PRINT "2. NOTE LENGTH": PRINT : PRI
NT "3. NOTES": PRINT : PRINT "4. OCTAVE": PRINT : PRINT "5. MODIFIERS": PRINT :
 PRINT "6. PAUSES": PRINT : PRINT "7. TO WRITE A TUN": PRINT : PRINT "PLEASE SELE
T BY NUMBER"
 150 IF INKEY$="" THEN  GO TO 150
 160 IF INKEY$="1" THEN  GO TO 230
 170 IF INKEY$="2" THEN  GO TO 240
 180 IF INKEY$="3" THEN  GO TO 250
 190 IF INKEY$="4" THEN  GO TO 330
 200 IF INKEY$="5" THEN  GO TO 340
 210 IF INKEY$="6" THEN  GO TO 350
 220 IF INKEY$="7" THEN  GO TO 360
 225 GO TO 150
 230 CLS : PRINT "TEMPO": PRINT "*****": PRINT : PRINT "TEMPO IS INDICATED BY TH
E LETTERT FOLLOWED BY A NUMBER": PRINT : PRINT "THE TEMPO IS THE SPEED AT WHICH
 THE PIECE IS PLAYED.  A VALUE   BETWEEN 1 AND 10 WILL MEET MOST NEEDS.": PRINT :
 PRINT "T2 IS A SPEED OF ONE BEAT PER   SECOND, T4 TWO BEATS PER SECOND": PRINT
: PRINT "IF YOU DO NOT ENTER A VALUE FOR T THE COMPUTER WILL ALLOCATE THEVALUE T
2": PRINT : PRINT "PRESS ANY KEY TO RETURN TO MENU": PAUSE 0: GO TO 140
 240 CLS : PRINT "NOTE LENGTH": PRINT "***********": PRINT : PRINT "    NOTE LEN
GTH IS INDICATED BY L FOLLOWED BY A NUMBER": PRINT : PRINT "MOST NOTE LENGTHS WI
LL BE IN THERANGE 1 TO 32": PRINT : PRINT "L1 IS A WHOLE NOTE, L2 A HALF   NOTE
 AND L4 A QUARTER NOTE": PRINT : PRINT "IF YOU DO NOT ENTER A VALUE FOR L THE COM
PUTER WILL ALLOCATE THEVALUE L4": PRINT : PRINT "L CAN BE MODIFIED BY (.) E.G.
 L2. = 1/2 + 1/4 NOTE = 3/4 NOTE": PRINT : PRINT "PRESS ANY KEY TO CONTINUE": PA
USE 0
 245 CLS : PRINT AT 0,6;"NOTE LENGTH VALUES";AT 1,6;"******************";AT 4,16
;"C ";AT 5,0;"A  = L1";AT 5,16;"B  = L2";AT 8,0;"C ";AT 8,16;"CE";AT 9,0;"D
= L
4";AT 9,16;"D  = L8";AT 12,0;"CF";AT 12,16;"CG";AT 13,0;"D  = L16";AT 13,16;"D
H
= L32";AT 16,0;"PRESS ANY KEY TO RETURN TO MENU": PAUSE 0: GO TO 140
 250 CLS : PRINT "NOTES": PRINT "*****": PRINT : PRINT "THE NOTES CAN BE REPRESE
NTED BY LETTERS AS SHOWN BELOW"
 260 PRINT AT 6,9;"C#";AT 6,12;"D#";AT 6,18;"F#";AT 6,21;"G#";AT 6,24;"A#"
 270 PRINT AT 7,9;"or";AT 7,12;"or";AT 7,18;"or";AT 7,21;"or";AT 7,24;"or"
 280 PRINT AT 8,9;"D-";AT 8,12;"E-";AT 8,18;"G-";AT 8,21;"A-";AT 8,24;"B-"
 290 PLOT 52,103: DRAW 168,0: PLOT 52,55: DRAW 168,0: FOR I=52 TO 220 STEP 24: P
LOT I,55: DRAW 0,48: NEXT I
 300 FOR I=9 TO 11: FOR J=9 TO 24 STEP 3: PRINT AT I,J;"■": NEXT J: NEXT I
 310 PRINT AT 13,8;"C";AT 13,11;"D";AT 13,14;"E";AT 13,17;"F";AT 13,20;"G";AT 13
,23;"A";AT 13,26;"B"
 320 PRINT AT 16,0;"SEQUENCES OF NOTES ARE ENTERED  BY LETTER E.G. GGACDBE";AT 1
9,0;"   PRESS ANY KEY TO CONTINUE": PAUSE 0
 325 CLS : PRINT AT 0,0;"THE POSITION OF THE NOTES ON    THE STAFF WHEN WRITTEN
IN THE   KEY OF ""G"" IS AS SHOWN BELOW": PRINT AT 5,23;"G";AT 6,0;"----------
---------F----------";AT 7,19;"E";AT 8,0;"-----------------D-------------";AT 9
,15;"C (04)";AT 10,0;"-------------B-------------------";AT 11,11;"A";AT 12,0;"--
--------G----------------------";AT 13,7;"F";AT 14,0;"-----E--------------------
```

```
----";AT 15,3;"D";AT 16,0;"-C-(03) MIDDLE ""C""";AT 20,0;"PRESS ANY KEY TO RETUR
N TO MENU": PAUSE 0: GO TO 140
 330 CLS : PRINT "OCTAVES": PRINT "*******": PRINT : PRINT "OCTAVES ARE INDICATE
D BY THE    LETTER O FOLLOWED BY A NUMBER": PRINT : PRINT "THERE IS A RANGE OF F
IVE OCTAVES": PRINT : PRINT "1 IS LOW  5 IS HIGH": PRINT : PRINT "MIDDLE C IS AT
 THE START OF THE THIRD OCTAVE": PRINT : PRINT "TO PLAY THE SCALE IN C YOU WOULD
": PRINT : PRINT "ENTER T203CDEFGAB04CC03BAGFEDC": PRINT : PRINT "PRESS ANY KEY
TO RETURN TO MENU": PAUSE 0: GO TO 140
 340 CLS : PRINT "MODIFIERS": PRINT "*********": PRINT : PRINT "THE NOTES CAN BE
 MODIFIED BY    # TO INDICATE SHARP AND BY - TO INDICATE FLAT": PRINT : PRINT "T
, O AND L CAN BE MODIFIED BY   THE FOLLOWING SUFFIXES": PRINT : PRINT "+ INCREAS
ES CURRENT VALUE BY 1": PRINT : PRINT "- DECREASES CURRENT VALUE BY 1": PRINT :
PRINT "> DOULES CURRENT VALUE": PRINT : PRINT "< HALVES CURRENT VALUE": PRINT :
PRINT ". INCREASES NOTE LENGTH BY HALF": PRINT : PRINT "PRESS ANY KEY TO RETURN
TO MENU": PAUSE 0: GO TO 140
 350 CLS : PRINT "PAUSES": PRINT "******": PRINT : PRINT "PAUSES ARE INDICATED B
Y LETTER  P FOLLOWED BY A NUMBER": PRINT : PRINT "VALUES OF P ABOVE 26 PRODUCE T
HESAME LENGTH OF DELAY": PRINT : PRINT "THE PAUSE HAS THE SAME VALUES ASTHE NOTE
 LENGTHS E.G. P4 WOULD 'BE A PAUSE LASTING FOR THE SAME LENGTH AS A QUARTER NOTE
": PRINT : PRINT "THE . MODIFIER CAN NOT BE USED  WITH THE PAUSE.  TO PAUSE L2.
  YOU WOULD USE P2P4": PRINT : PRINT "   PRESSANY KEY TO CONTINUE": PAUSE 0
 355 CLS : PRINT AT 0,6;"PAUSE LENGTH VALUES";AT 1,6;"******************";AT 5,
0;"I  = P1";AT 5,16;"J  = P2";AT 8,0;"O";AT 9,0;"P  = P4";AT 8,16;"K";AT 9,16;
"L
  = P8";AT 12,0;"M";AT 13,0;"L  = P16";AT 12,16;"M";AT 13,16;"N  = P32";AT 16,
0;
"PRESS ANY KEY TO RETURN TO MENU": PAUSE 0: GO TO 140
 360 CLS : PRINT "WRITING OR TRANSCRIBING MUSIC": PRINT "**********************
******": PRINT "HAVING READ THE INSTRUCTIONS YOUSHOULD NOW TRY ENTERING A TUNE.
FOR YOUR FIRST ATTEMPT TRY THE  EXAMPLE BELOW.": PRINT : PRINT "T403L2GB04CDL4CO
3BAGL2ADDL1A03L2GB04CDL4CO3BAGL2ADDL1GL2BGG04CO3L4BAGF#L2ADDL1A03L2GB04CDL4CO3BA
GL2ADDL1G": PRINT : PRINT "NOW TRY YOUR OWN TUNE"
 361 PRINT AT 14,0;"      C     C     CE    CF     ";AT 15,0;"A =L1 B =L2 D =
L4
D =L8 D =L16 ";AT 17,0;"              O     K     M     ";AT 18,0;"I =P1 J =P2 P
=
P4 L =P8 L =P16 ": INPUT " ENTER YOUR TUNE STRING PLEASE  "; LINE A$: LET N=1
 370 IF A$( TO N)="T" THEN  GO TO 390
 380 IF A$( TO N)<>"T" THEN  LET A$="T2"+A$: LET N=3: GO TO 410
 390 IF CODE A$(N+1 TO N+1)>=48 AND CODE A$(N+1 TO N+1)<=57 THEN  LET N=N+1: GO
TO 390
 400 LET N=N+1
 410 IF A$(N TO N)<>"O" THEN  LET A$=A$( TO N-1)+"O2"+A$(N TO ): LET N=N+2: GO T
O 430
 420 IF A$(N TO N)="O" THEN  LET N=N+2
 430 IF A$(N TO N)<>"L" THEN  LET A$=A$( TO N-1)+"L4"+A$(N TO )
 440 CLS : PRINT "A$ NOW = ";A$
 450 PRINT "TO PLAY YOUR TUNE USE THE       FOLLOWING SPECTRUM BEEPS": PRINT
 460 LET A$=A$+"S   "
 470 DIM H(LEN A$): DIM J(LEN A$): LET K=1
 480 FOR N=1 TO LEN A$
 490 IF A$(N TO N)="S" THEN  PRINT : PRINT "THE TUNE WILL NOW BE PLAYED AT  THE
CORRECT SPEED, THEN LISTED": GO TO 1200
 500 IF A$(N TO N)=";" OR A$(N TO N)="." OR A$(N TO N)="+" OR A$(N TO N)="-" OR
A$(N TO N)="<" OR A$(N TO N)=">" OR A$(N TO N)="#" THEN  NEXT N
 510 IF CODE A$(N TO N)>=48 AND CODE A$(N TO N)<=57 THEN  NEXT N
 520 IF A$(N TO N)="T" THEN  GO SUB 590
 530 IF A$(N TO N)="O" THEN  GO SUB 710
 540 IF A$(N TO N)="L" THEN  GO SUB 820
 550 IF A$(N TO N)="A" OR A$(N TO N)="B" OR A$(N TO N)="C" OR A$(N TO N)="D" OR
A$(N TO N)="E" OR A$(N TO N)="F" OR A$(N TO N)="G" THEN  GO SUB 940
 560 IF A$(N TO N)="P" THEN  GO SUB 1100
 570 IF A$(N TO N)="V" THEN  NEXT N
 580 NEXT N
 590 IF CODE A$(N+1 TO N+1)>=48 AND CODE A$(N+1 TO N+1)<=57 THEN  LET X=VAL A$(N
+1 TO N+1)
 600 IF CODE A$(N+1 TO N+1)<48 OR CODE A$(N+1 TO N+1)>57 THEN  GO TO 650
 610 IF CODE A$(N+2 TO N+2)>=48 AND CODE A$(N+2 TO N+2)<=57 THEN  LET X=10*X+VAL
 A$(N+2 TO N+2)
```

```
 620 IF CODE A$(N+2 TO N+2)<48 OR CODE A$(N+2 TO N+2)>57 THEN  GO TO 650
 630 IF CODE A$(N+3 TO N+3)<48 OR CODE A$(N+3 TO N+3)>57 THEN  GO TO 650
 640 IF CODE A$(N+3 TO N+3)>=48 AND CODE A$(N+3 TO N+3)<=57 THEN  LET X=10*X+VAL
 A$(N+3 TO N+3)
 650 IF A$(N+1 TO N+1)="+" THEN  LET X=X+1
 660 IF A$(N+1 TO N+1)="-" THEN  LET X=X-1
 670 IF A$(N+1 TO N+1)=">" THEN  LET X=2*X
 680 IF A$(N+1 TO N+1)="<" THEN  LET X=X/2
 690 LET T=2/X
 700 RETURN
 710 IF CODE A$(N+1 TO N+1)>=48 AND CODE A$(N+1 TO N+1)<=57 THEN  LET Y=VAL A$(N
+1 TO N+1)
 720 IF A$(N+1 TO N+1)="+" THEN  LET Y=Y+1
 730 IF A$(N+1 TO N+1)="-" THEN  LET Y=Y-1
 740 IF A$(N+1 TO N+1)=">" THEN  LET Y=2*Y
 750 IF A$(N+1 TO N+1)="<" THEN  LET Y=Y/2
 760 IF Y=1 THEN  LET S=-25
 770 IF Y=2 THEN  LET S=-13
 780 IF Y=3 THEN  LET S=-1
 790 IF Y=4 THEN  LET S=11
 800 IF Y=5 THEN  LET S=23
 810 RETURN
 820 IF CODE A$(N+1 TO N+1)>=48 AND CODE A$(N+1 TO N+1)<=57 THEN  LET Z=VAL A$(N
+1 TO N+1)
 830 IF A$(N+2 TO N+2)="." THEN  LET Z=(2*Z)/3
 840 IF CODE A$(N+2 TO N+2)>=48 AND CODE A$(N+2 TO N+2)<=57 THEN  LET Z=10*Z+VAL
 A$(N+2 TO N+2)
 850 IF A$(N+3 TO N+3)="." THEN  LET Z=(2*Z)/3
 860 IF CODE A$(N+3 TO N+3)>=48 AND CODE A$(N+3 TO N+3)<=57 THEN  LET Z=10*Z+VAL
 A$(N+3 TO N+3)
 870 IF A$(N+4 TO N+4)="." THEN  LET Z=(2*Z)/3
 880 IF A$(N+1 TO N+1)="+" THEN  LET Z=Z+1
 890 IF A$(N+1 TO N+1)="-" THEN  LET Z=Z-1
 900 IF A$(N+1 TO N+1)=">" THEN  LET Z=2*Z
 910 IF A$(N+1 TO N+1)="<" THEN  LET Z=Z/2
 920 LET L=1/Z
 930 RETURN
 940 IF A$(N TO N)="C" THEN  LET W=1
 950 IF A$(N TO N)="D" THEN  LET W=3
 960 IF A$(N TO N)="E" THEN  LET W=5
 970 IF A$(N TO N)="F" THEN  LET W=6
 980 IF A$(N TO N)="G" THEN  LET W=8
 990 IF A$(N TO N)="A" THEN  LET W=10
1000 IF A$(N TO N)="B" THEN  LET W=12
1010 IF A$(N+1 TO N+1)="#" THEN  LET W=W+1
1020 IF A$(N+1 TO N+1)="-" THEN  LET W=W-1
1030 LET DURATION=INT (1000*(T*L+0.0005))/1000
1040 LET PITCH=S+W
1050 LET H(K)=DURATION: LET J(K)=PITCH
1060 PRINT "BEEP ";DURATION;",";PITCH;" : ";
1070 BEEP DURATION,PITCH
1080 LET K=K+1
1090 RETURN
1100 IF CODE A$(N+1 TO N+1)>=48 AND CODE A$(N+1 TO N+1)<=57 THEN  LET U=VAL A$(N
+1 TO N+1)
1110 IF CODE A$(N+2 TO N+2)<48 OR CODE A$(N+2 TO N+2)>57 THEN  GO TO 1150
1120 IF CODE A$(N+2 TO N+2)>=48 AND CODE A$(N+2 TO N+2)<=57 THEN  LET U=10*U+VAL
 A$(N+2 TO N+2)
1130 IF CODE A$(N+3 TO N+3)<48 OR CODE A$(N+3 TO N+3)>57 THEN  GO TO 1150
1140 IF CODE A$(N+3 TO N+3)>=48 AND CODE A$(N+3 TO N+3)<=57 THEN  LET U=10*U+VAL
 A$(N+3 TO N+3)
1150 LET PAUSE=INT ((50*T)/U)
1160 IF PAUSE<1 THEN  LET PAUSE=1
1170 PRINT "PAUSE ";PAUSE;" : ";
1180 LET H(K)=0: LET J(K)=PAUSE: LET K=K+1
1190 RETURN
1200 FOR I=1 TO K-1
1210 IF H(I)=0 THEN  PAUSE J(I): NEXT I
1220 BEEP H(I),J(I)
```

```
1230 NEXT I
1240 CLS : PRINT "TO INCLUDE YOUR TUNE IN PROGRAM TYPE (LINE No) ";
1250 FOR I=1 TO K-1
1260 IF H(I)=0 THEN  PRINT "PAUSE ";J(I);" : ";: NEXT I
1270 PRINT "BEEP ";H(I);",";J(I);" : ";
1280 NEXT I
1290 PRINT TAB 0;"PRESS ""M"" FOR RETURN TO MENU,   PRESS ""P"" TO PLAY AGAIN AN
D LIST"
1300 IF INKEY$="" THEN  GO TO 1300
1310 IF INKEY$="M" THEN  GO TO 140
1320 IF INKEY$="P" THEN  GO TO 1200
1330 GO TO 1300
1340 REM PROGRAM TO CONVERT THE DRAGON COMPUTER SOUND COMMAND TOSPECTRUM BEEPS.
  BY R.A.HOULTONMARCH 1984 UPDATED NOVEMBER 1984
1350 CLS : PRINT : PRINT "    THE DRAGON SOUND COMMAND IS VERY SIMILAR TO THE SP
ECTRUM    BEEP COMMAND AND CAN BE COPIED  BY THIS PROGRAM"
1360 PRINT "THE SOUND COMMAND HAS THE FORM": PRINT : PRINT "SOUND P,D": PRINT :
PRINT "RANGE OF P 1(LOW) TO 255(HIGH)": PRINT "(MIDDLE C = 90)": PRINT : PRINT "
RANGE OF D 1(SHORT) TO 255(LONG)": PRINT "(1 SECOND = 16)": PRINT : PRINT "PLEAS
E ENTER P AND D"
1370 INPUT "P FIRST PLEASE ";P: IF P<1 OR P>255 THEN  GO TO 1370
1380 INPUT "AND NOW D ";D: IF D<1 OR D>255 THEN  GO TO 1380
1390 LET PITCH=INT (P/3)-30: LET E=D: IF E>160 THEN  LET E=160
1400 LET DURATION=INT ((E/16)*1000+0.0005)/1000
1410 PRINT : PRINT "FOR SOUND ";P;",";D: PRINT : PRINT "USE BEEP ";DURATION;",";
PITCH
1420 PRINT : PRINT "TO HEAR SOUND PRESS ANY KEY": PAUSE 0: BEEP DURATION,PITCH
1430 PRINT : PRINT "FOR ANOTHER SOUND PRESS S": PRINT : PRINT "ANY OTHER KEY TO
RETURN TO MENU"
1440 IF INKEY$="" THEN  GO TO 1440
1450 IF INKEY$="S" OR INKEY$="s" THEN  GO TO 1350
1460 RESTORE 1500: GO TO 10
1470 FOR I=USR "A" TO USR "P"+7
1480 READ A: POKE I,A
1490 NEXT I
1500 DATA 0,60,66,129,129,66,60,0
1510 DATA 1,61,67,129,129,66,60,0
1520 DATA 1,1,1,1,1,1,1,1
1530 DATA 1,61,127,255,255,126,60,0
1540 DATA 128,64,32,16,0,0,0,0
1550 DATA 128,64,32,144,64,32,16,0
1560 DATA 128,64,32,144,64,32,144,64
1570 DATA 32,16,0,0,0,0,0,0
1580 DATA 0,0,255,60,60,0,0,0
1590 DATA 0,0,60,60,255,0,0,0
1600 DATA 58,58,60,4,4,8,8,8
1610 DATA 8,16,16,16,16,32,32,32
1620 DATA 58,58,60,4,116,120,120,8
1630 DATA 232,240,240,16,16,32,32,32
1640 DATA 64,32,16,8,4,6,15,30
1650 DATA 60,248,112,32,112,248,252,132
1660 RETURN
1665 REM (Enter lines 1680 and       1700 in ordinary capitals       lines 1690
and 1710 in          graphics<CAP SHIFT>+<9>.)
1670 REM GRAPHICS CHARACTERS
1680 REM A B C D E F G H I J K L
1690 REM A B C D E F G H I J K L
1700 REM M N O P
1710 REM M N O P
Type UDG'S J as G J
```



B. TWISSELL

Error Handling

Sinclair really shouldn't have left the provision of an ON ERROR facility to us.

This article describes a short machine code routine which adds an ON ERROR facility to Spectrum BASIC.

There are many ways in which a program can terminate:

1) Legitimately, by reaching the end of the program or by a STOP statement.
2) By a bug in the program: NEXT without FOR, Out of DATA, etc.
3) By an unforeseen event: Out of memory, Number too big, etc.
4) By an I/O error: STOP in INPUT, Tape loading error, End of file, etc.
5) By the user typing BREAK.

When one of these events occurs, we may want the program to continue running, and possibly take some special action. There are many things that can be done:

a) terminate.
b) ignore the error and carry on regardless.
c) repeat the statement which caused the error.
d) jump to some other statement.
e) perform some statement (or routine) and then return to the interrupt statement.
f) hang up or self destruct.

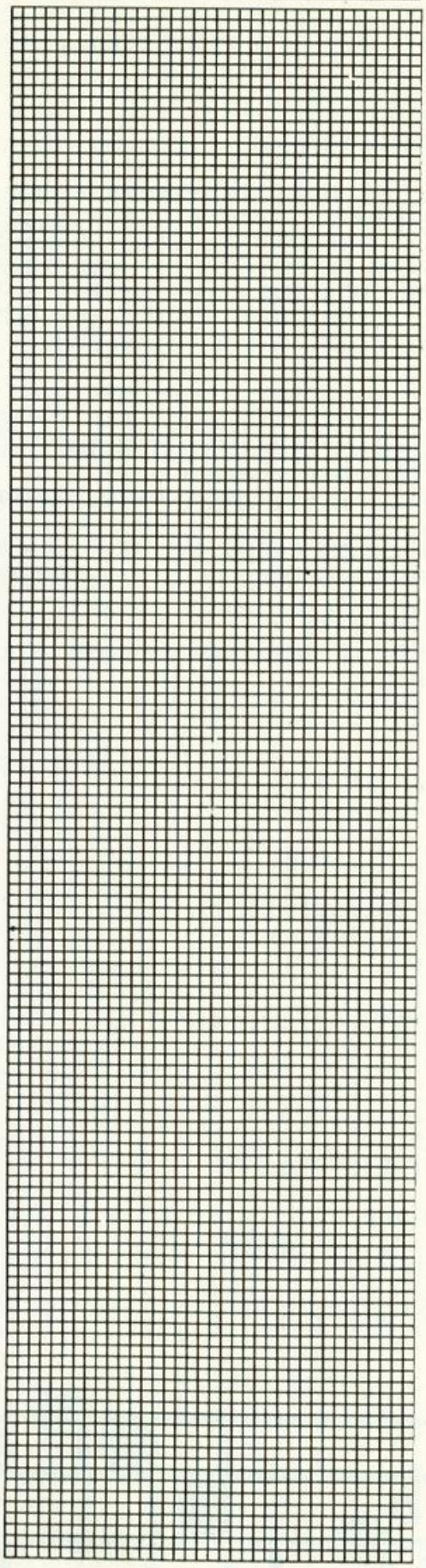
We might want different actions for different errors; eg (b) is suitable for (5) but (d) is more suitable for (3). The only action provided by Spectrum BASIC is of course (a).

THE COMPETITION

Other microcomputers have ON ERROR and ON BREAK statements for specifying the actions to be taken when errors occur. In principle, Interface 1 owners could write a machine code routine to add such statements to the Spectrum. However, the facility described here is accessed via USR, PEEK and POKE; it can be used whether Interface 1 is present or not. The new facility will handle all the errors which have a report number — thus things like "L BREAK" are handled, but note that "Microdrive not present", etc are not.

PROGRAM 1

Program 1 loads a machine code routine into memory. If the checksum is correct you should save the code using:



SAVE — — — CODE 30000,64

The dashes should be replaced with the filename of your choice. The routine is actually relocatable, which is a posh way of saying that it can be placed anywhere in memory. Thus you can load it again by:

CLEAR 59999
LOAD — — — CODE 60000

or whatever.

To turn the error processing on, you simply call the routine, using:

RANDOMIZE USR 30000

30000 is of course replaced by the appropriate address if you've loaded it elsewhere. This would normally be the first statement in the program, for the obvious reason that error processing is turned on before any errors can occur. This implies that you should load the machine code routine before loading the BASIC program.

When error processing is on, the BASIC program is allowed to terminate 'normally', either with "OK" or by a STOP statement. The error processing will be turned off at this point, so if you re-enter the program you'll have to turn the error processing on as before.

An attempt to terminate with any other error report will be caught, so that the BASIC program continues running.

DESIGN CONSIDERATIONS

The routine has been designed to keep the interface with BASIC simple, and also to keep the routine itself simple. It would have been possible — by some suitable POKE statements — to tell the routine such things as the line number to jump to, but this was rejected for simplicity's sake.

Perhaps the most obvious thing which the routine could do is an automatic CONTINUE whenever an error occurs. This would mean for example that BREAK would be ignored, but if you wanted to take some action you could detect that a BREAK had occured either by looking at the keyboard or by PEEKing the variable ERRNR. However, in most cases, CONTINUE repeats the statement which caused the error. If the error was, say, 'Number too big', then the statement would fail again, and the program would hang in a tight loop for ever. And if a VERIFY fails, there's no point redoing the VERIFY unless you also redo the SAVE first.

PROGRAM 1

Implements an ON ERROR facility in Spectrum BASIC. See text.

```
20      CLEAR 29999
30      address=30000
40      LET h$="210f000922b05ceb2a3d5c732372c93a
        3a5c3c2802fe09ca031321445ccb7e280b3a475c3-
        c772a4 55c22425c2100007c32715c220b5c2ab05ce52a4
        25cc39elb"
50      LET sum=0
60      FOR i=1 TO LEN h$ STEP 2
70      LET byte=16*FN d(h$(i))+FN d(h$(i+1))
80      POKE address+(i-1)/2,byte
90      LET sum=sum+byte
100     NEXT i
110     IF sum  5087 THEN PRINT FLASH 1'" "Line 40
        is wrong!!!!"
120     DEF FN d(c$)=CODE c$-(CODE "0" AND
        c$ ="9")-(CODE "a"-10 AND c$>="a")
```

PROGRAM 2

Example of ON ERROR facility. This assumes that the necessary machine code routine is in memory at address 30000.

```
20      RANDOMIZE USR 30000:
        LET errnr=23610:
        LET ok=255:
        LET number too big=5:
        LET stop in input=16:
        LET invalid file name=14:
        LET tape loading error=26
30      PRINT "Setting up array."' "You can't BREAK this"'
        "(try it and see!!)"
40      DIM n(300)
50      FOR i=1 TO 300
60      POKE errnr,ok
70      LET n(i)=1/INT (10*RND)
80      IF PEEK errnr=number too big THEN PRINT
        "overflow"
90      NEXT i
100     REM LOOP
110     POKE errnr,ok
120     INPUT "Enter file name."' "STOP (shift-6)is"' "inef-
        fective..."' LINE f$
130     IF PEEK errnr=stop in input THEN GO TO 110
140     REM END LOOP
150     REM LOOP
160     SAVE f$ DATA n( )
170     IF PEEK errnr=invalid file name THEN STOP
180     PRINT "Now to verify."' "If it fails, the SAVE"'
        "will be repeated,"' "BREAK will get you out."
190     POKE errnr, ok
200     VERIFY f$ DATA n( )
210     IF PEEK errnr=tape loading error THEN GO TO 160
220     REM END LOOP
230     PRINT "Finished"
240     STOP
```

Rather than have a lot of special cases to try and cope with all eventualities, the solution chosen has no special cases — after an error, execution is always resumed at the next statement. For example in:

```
100 LET n=4
110 LET n=1/0
120 PRINT n
```

the division will result in Number too big. The program will not crash, but will resume execution at the PRINT statement. Since statement 110 is not completed, n will still hold its old value, and so 4 will be printed.

WHAT NEXT?

After an error has occured, the system variable ERRNR (one byte at 23610) will hold a value other than 255. ERRNR corresponds to error reports as follows:

ERRNR	REPORT
255	0(OK)
0,1,...,8	1,2,....,9
9,10,....,26	A,B,....,r

ERRNR will hold such a value until the next error occurs, at which point it will be overwritten with the new error number.

If you want to detect when an error occurs, you should first clear any previous error condition by POKEing 255 in the ERRNR. Beware that executing the last statement does not set ERRNR to 255, because it is assumed to still contain the 255 which was put there when the program began execution. The upshot of this is that if some error has occurred and you haven't cleared ERRNR, the program may not be allowed to finish. You are advised to use a STOP statement whenever you want the program to finish since this sets ERRNR to 8 and will always work.

Thus all error types (2) to (5) mentioned earlier are caught, and any of the actions (a) to (f) can be taken by appropriate code in BASIC.

Program 2 gives some examples.

ON ERROR MACHINE CODE

Adds on ERROR facility to Spectrum BASIC. See text.

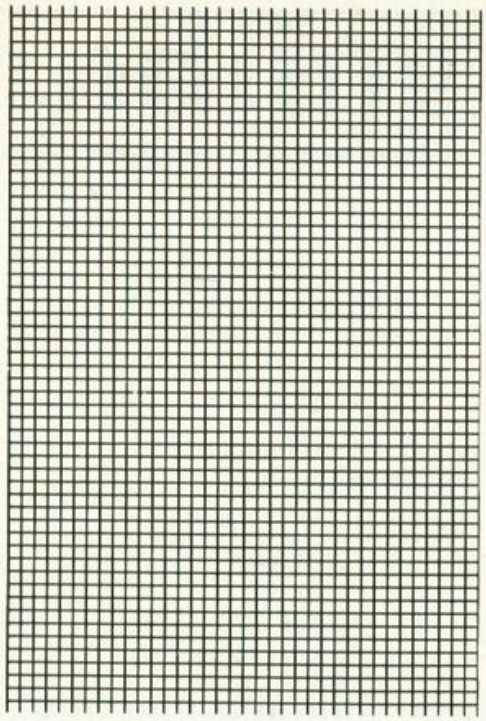
First of all there is a short piece of code to turn on the error processing. This is a matter of working out the address of the <on> label — when this is invoked by USR in BASIC, BC will have the value of <turnon>, and <on> is 15 bytes away — then the address of <on> is stored in two places: an unused location at ▷

UTILITY

23728; and in to the location pointed to by ERRSP...

```
turnon  LD      HL,15
        ADD     HL,BC
        LD      (23728),HL
        EX      DE,HL
        LD      HL,(ERRSP)
        LD      (HL),E
        INC     HL
        LD      (HL),D
        RET
```

The BASIC interpreter may detect any error at all sorts of places. However, the action is always the same (except that is, for errors from Interface 1). The interpreter clears all the junk from the Z80's stack. Since the length of the stack depends on where the error occurred, ERRSP always points to the 2nd value left on the top is an address. Normally it is 1303H; the code at this address prints an error message and prepares to receive a command. The error processing is turned on (above) by changing this address on the stack. Thus when an error occurs, the stack is cleared and our code is involed. For error numbers 0(ok) and 9 (STOP statement) we allow the program to terminate by jumping to the code which was supposed to be executed...



```
on      LD      A,(ERRNR)
        INC     A
        JR      Z,L1
        CP      9
L1      JP      Z,1303H
```

if a jump is about to take place, bit 7 of NSPPC will be 0 — we do nothing. Otherwise, we set up NEWPPC and NSPPC with the line number and statement which is to be executed next (this is one more than the current statement)...

```
        LD      HL,NSPPC
        BIT     7,(HL)
        JR      Z,L2
        LD      A,(SUBPPC)
        INC     A
        LD      (HL),A
        LD      HL,(PPC)
        LD      (NEWPPC),HL
```

now we clear a couple of things which would ordinarily be cleared..

```
L2      LD      HL,0
        LD      A,H
        LD      (FLAGX),A
        LD      (DEFADD),HL
```

finally, the address of our error will have been popped from the stack, so we put it back (remember, we stored it in an unused location)...

```
        LD      HL,(23728)
        PUSH    HL
```

now we jump back in to the interpreter, as if CONTINUE had just been entered, but after the check for BREAK...

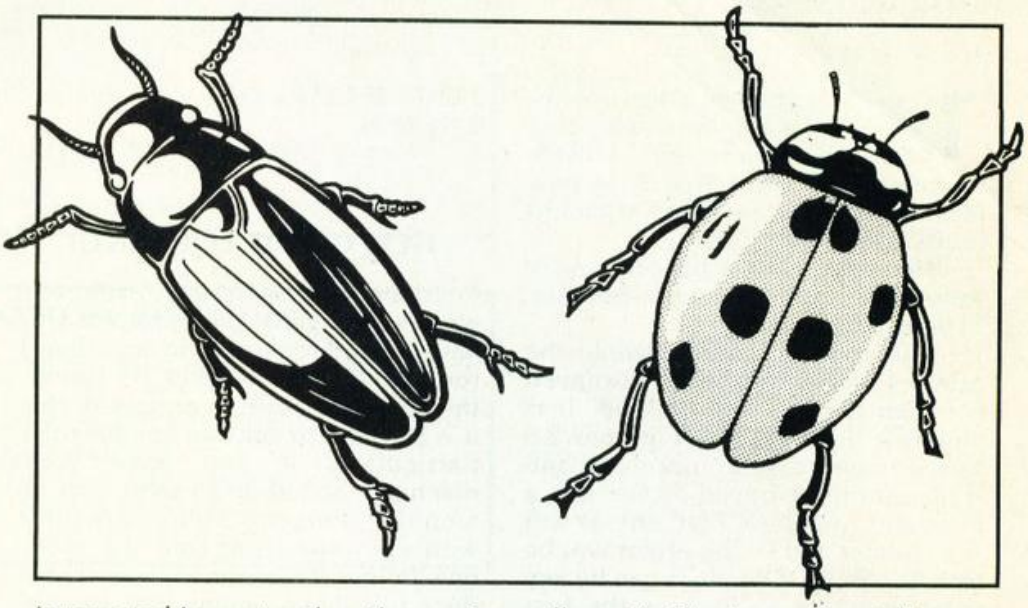
```
        LD      HL,(NEWPPC)
        JP      1B9EH
```

Bugbuster

Careful typist or not, you can still be infested. Who you gonna call? Bugbuster!

Typing in a program is a useful exercise. Apart from the patience required, techniques learned and the end program to be used, probably the most educational part of it is tracking down the bugs introduced by yourself or occasionally by our publication system.

In debugging you gain a much deeper insight and understanding on how the program actually works than by merely typing it in, but tracking down these errors is an art in itself and needs some skill. So here are some tips to help you in your efforts when faced with that cryptic error report!



1 NEXT WITHOUT FOR

Look back through the program. Either the loop has not been set up — no related FOR 'letter' = No1 TO No2 line, or the letter has been reused as an ordinary variable within the loop with a LET 'letter' = No.

2 VARIABLE NOT FOUND

This is one of the most common errors. Again, the problem may not lie in the line where the error was detected and reported. If there is only one variable, which may be one or more letters or a string($) variable, then that is the problem. There may be more than one variable in the line section reported and you will have to identify the offending one. In a line PRINT AT Y,X;A$ the culprit could be Y or X or A$. To find out which of them is causing the problem (it may be more than one) type in turn as a direct command:

PRINT Y Enter/Newline
PRINT X Enter/Newline
PRINT A$ Enter/Newline

Note which produces the error report. Now look back through the program printout for the line which sets it up — usually a LET or FOR command. Did you leave it out? Does the program get there or has a GOTO/GOSUB been wrongly addressed?

3 SUBSCRIPT WRONG

Connected with DIM A(No) or DIM A$(No). If the number in the brackets on the line where the error is reported is greater than the one in the original DIM statement, is not an integer or is less than 1, then this report is generated. If the subscript — number in brackets — is a number then check and change, however, if it is a variable then follow the procedure for tracing variables. It has probably exceeded the limits; look for lines with the variable being altered with + − */: if necessary add limiting code. For example:

IF X >10 THEN LET X=10

4 OUT OF MEMORY

As well as for programs which are too big, it may happen if the previous program set RAMtop. Before despairing, enter CLEAR USR "a"−1.

7 RETURN WITHOUT GOSUB

Somehow the computer has reached a RETURN command other than via a GOSUB instruction. Check a GOTO hasn't been entered in place of a GOSUB. Check for a missing GOSUB.

8 INTEGER OUT OF RANGE

An integer (whole number) either as a number or variable is too big or small and you are attempting to do something like PRINT AT 0,33 — not allowed! Check any variables involved as per report 2 and trace it back through the program looking for adjustments to it by + − */: Add limiting code if needed — see report 3.

9 OUT OF DATA

Check the number of DATA items match the number of READs; usually one (or more) has been missed out. Attempting to reread a DATA list without first using a RESTORE command will cause this and it can happen on an auto start program (saved with a LINE number). Good programming usually RESTOREs to the correct line number before using READ.

10 FOR WITHOUT NEXT

See report 1 but this time the NEXT is missing!

Note that the letters I have used for examples could by ANY letters not just A$,X,Y etc and depend on the particular choice of the programmer.

This is by no means a comprehensive list but I have tried to cover many of the most common error reports. Personally, I get almost as much satisfaction from debugging as I do from programming. I do assure you, however, that there is absolutely no truth in the rumour that we deliberately inject bugs into our listings in order to introduce you to the dubious delights of debugging!

PHILIP CRAWLEY

Supercharger

Hyper short routines to produce mega sound, border and protection tricks for your own crucial programs.

The ZX Spectrum, when released, was at the fore regarding colour, sound and HIRES. However it is clear that it is now lacking in some facilities standard on its contemporaries.

However, with a bit of trickery some nice results are quite feasible. Type in Program 1.

This routine is useful while the rules of a game are being displayed or when a game has finished. It is vital that the code is left at lines 2-6 as the timing is very important, and if the routine is moved deeper into a program, the GOTO (at line 4) will take longer and so the effect will be lost. The BORDER values can be any valid numbers so long as the first two and last two are all equal (1 = blue in this case). Another good border effect can be achieved by changing line 4 to:

```
4 PAUSE 1:IF INKEY$= ""THEN GOTO 3
```

For an explosion using the border try this little routine:

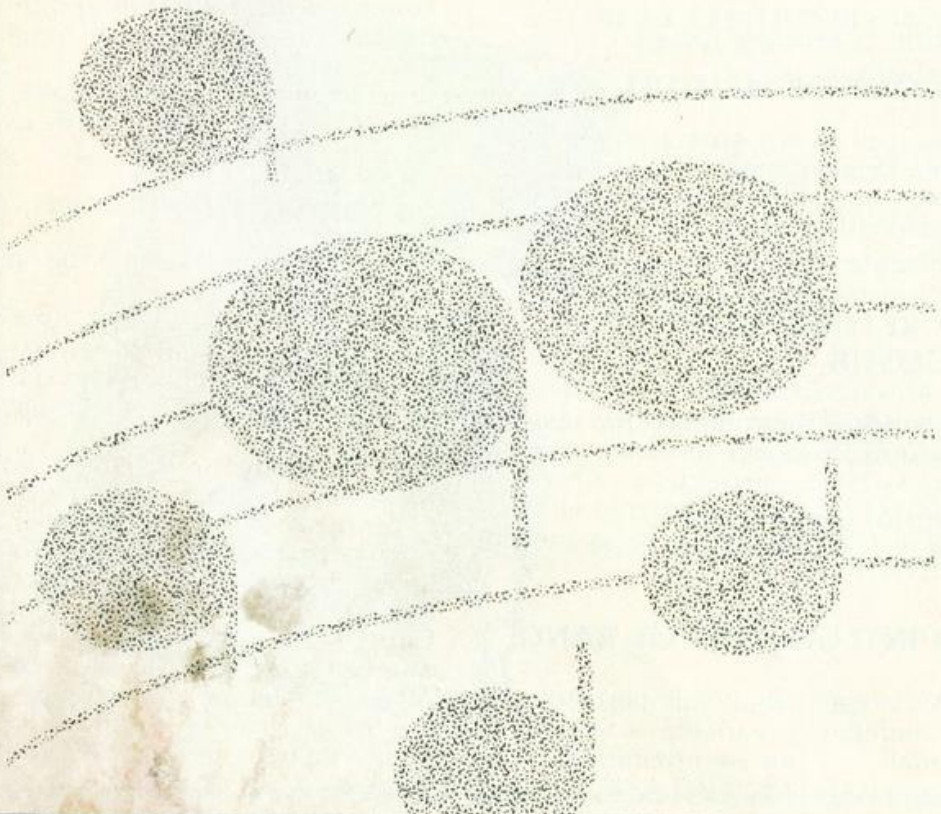
```
1 FOR N=0 TO 100
2 OUT 254,254*RND
3 NEXT N
```

HEY, GOOD LOOKING!

Combine this with any explosion visuals your game contains and you have a good looking (and sounding!) routine. On the subject of sound, the Spectrum is often criticised. But it is possible to achieve good results particularly if you have your machine hooked up to some sort of amplifier. Program 2 plays a rhythm with one tone rising and the other one falling. Programs 3 and 4 produce telephone ringing and a jingle respectively.

Since the BASIC interpreter is lacking in some aspects, several facilities may be implemented via the system or by calls to ROM routines as shown in Tables 1 and 2.

Getting back to the subject of borders try typing in the following line. LET T=USR XXXX where XXXX is an even number between 1310 and 1320. Try all the values as they give different colours.



ANTI-PIRACY DEVICE

If you want to protect your software from piracy the following method works extremely well. Without the use of colour cards or expensive 'dongles' it is possible to make a 48K program copyproof.

Spectrum tape software is usually made up of three distinct parts; short BASIC loader programs, SCREEN$ to keep you amused while the game loads and then machine code. So, to start with lower RAMTOP, load the machine code then enter Program 5. Leave out the normal loader program. It is important that the software is fully de-bugged. Now clear the screen and load your SCREEN$. Then making sure you don't lose the screen, type GOTO 1 and set your tape drive to record.

The principle behind the operation is as follows. No matter what the adverts state, a copier program does require some memory to run it. This may be the printer buffer or the UDG area, but no matter how devious the copier is it is in there somewhere. So if you make a file a full 48K long, it will overwrite the copier program and make it impossible to copy by software means.

The program must be loaded by LOAD ""PROG""CODE and will autorun. Also, the program is altered such that during LOADing, pressing the BREAK key will cause a system reset. Line 3 diables the LIST command so if your program is in BASIC, a measure of extra software protection is provided.

Since the whole of the memory is saved, when the program re-loads, it will take on exactly the same state as it had just after the save procedure. Line 5 disables the break key altogether so it won't crash if the BREAK key is used after loading is complete. Incidentally, if your software is written in BASIC, then Lines 1 and 6 can be omitted. They are, however, vital if your software contains machine code.

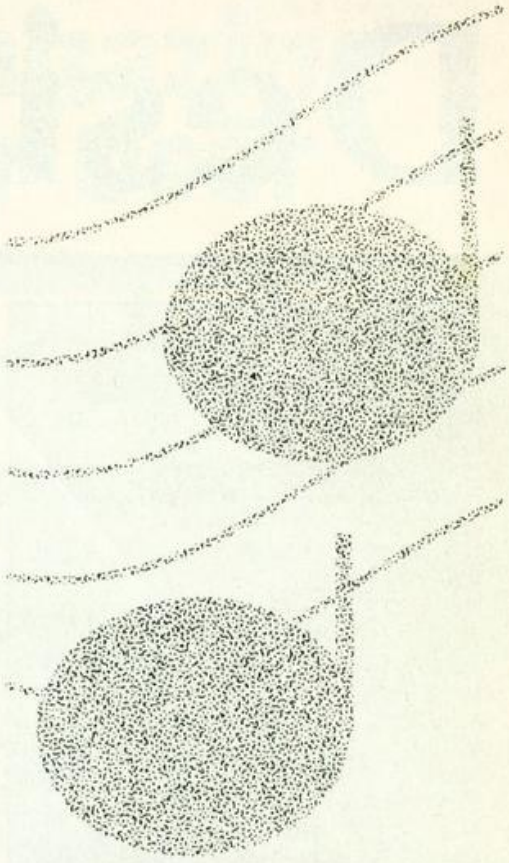
Although this method does increase loading time to five minutes, it is surely worth it when you consider that the software cannot spawn illegal copies. Do your bit for anti-piracy!

TABLE 1 SYSTEM VARIABLES

POKE 237546,0	*— Make line zero out of line 1 — un-editable*
POKE 23658,8	*— CAPS LOCK*
POKE 23658,0	*— CAPS LOCK OFF*
POKE 23755,100	*— Disable LIST command*
POKE 23755,0	*— Re-enables LIST command*
POKE 23659,0	*— 24 line screen, POKE 23659,2 before program finishes*
POKE 23736,181	*— Stops the 'Start tape. . . etc.' prompt*
POKE 23794,181	*— Same, if Interface One is fitted*
POKE 23693,56	*— Resets screen attributes*
POKE 23613,	*— Disable BREAK key*
PEEK (23730)−5	
POKE 23613,0	*— Reset error vector, protects program during LOAD*
POKE 23692,255	*— Disable 'Scroll' prompt*

TABLE 2 CALLS TO ROM ROUTINES

LET T=USR 7997:	*— T is the number of frames before keypress*
LET T=7997-T	
LET T=USR 3582	*— Simulates ZX81 SCROLL function*
LET T=USR 3213	*— Stops and asks 'Scroll'*
LET T=USR 4757	*— Prints Sinclair copyright message*
LET T=65536-USR 7962	*— Number of free bytes*



```
FOR a=1 TO 2:FOR b=11.6 TO 12 STEP 0.1:BEEP .018,b:FOR
d=1 TO 22:NEXT d:NEXT a:RETURN
```

```
FOR b=62 TO 24 STEP -6:BEEP .022,b:NEXT b:RETURN
```

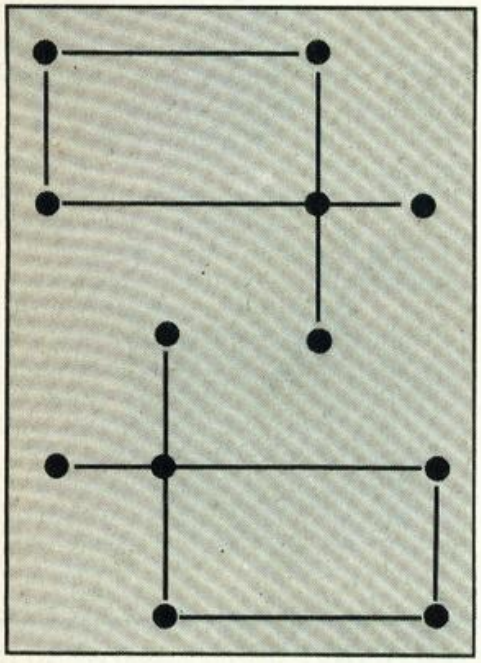
```
1 CLEAR XXXX:REM Your value
2 POKE 23613,0
3 POKE 23755,100
4 SAVE "PROG",CODE 16384,49152
5 POKE 23613, PEEK(23730)-5
6 RANDOMIZE USR XXXX+1
```

```
1 GO TO 100
2:
3 BORDER 1:BORDER 1:BORDER 6:BORDER 0:BORDER 4:BORDER
2:BORDER 3:BORDER 5:BORDER 1:BORDER 1
4 IF INKEY$="" THEN GO TO 3
5 RETURN
6:
100 REM display instructions
105 BORDER 7:PAPER 7:CLS
110 PRINT 1,10; INVERSE 1;"BORDER ROLL"
120 PRINT #1;TAB 4;FLASH 1;"PRESS ANY KEY TO CONTINUE"
130 GO SUB 2
140:
150 REM rest of program
```

```
10 FOR n=0 TO 20
20  BEEP 0.01,n: BEEP 0.01,20-n
30  NEXT n
40  PAUSE 0: RUN
```

Designer

Make the most of your designing ability with this short program.



This program allows you to create designs on your Spectrum using the PLOT, DRAW, and OVER commands.

After specifying the BORDER, PAPER and INK colours, you are invited, through user friendly instructions, to either plot a point, draw a line, or plot and draw over a line (in order to remove any mistakes). It is worth noting that DRAW always draws from the last point plotted, so that if you want a line separate from the last point drawn you must plot a point first.

DRAWBACKS

On this last point, one of the major drawbacks (pun intended) of the DRAW command is that the co-ordinate of the end of the line are offset of the last point plotted. However, it is quite easy to program around this by using the two system variable, bytes 23677 (which holds the y co-ordinate of the last point plotted).

The functions defined in lines 10 and 20 subtract those held in these bytes from the absolute screen co-ordinates you wish to draw to and gives the offset values required by the DRAW command. This method has ramifications for graphics work far beyond the scope of this relatively simple, if enjoyable, program and I hope that you will be able to make good use of it.

```
  10 DEF FN x(a)=a-PEEK 23677
  20 DEF FN y(a)=a-PEEK 23678
  30 BORDER 1: PAPER 1: INK 7: C
LS
  40 GO TO 3000
 100 INPUT "Enter 1 to plot, 2 t
o draw ";a
 110 IF a<1 OR a>2 THEN GO TO 10
0
 120 GO TO 1000*a
 999 REM **plot coords**
1000 INPUT "Enter 0 to plot, 1 t
o plot OVER ";a
1010 IF a<0 OR a>1 THEN GO TO 10
00
1020 INPUT "Enter plot coords ";
b;" ";c
1030 IF b<0 OR b>255 OR c<0 OR c
>175 THEN GO TO 1000
1040 PLOT OVER a;b,c
1050 GO TO 100
1999 REM **draw coords**
2000 INPUT "Enter 0 to draw, 1 t
o draw OVER ";a
2010 IF a<0 OR a>1 THEN GO TO 20
00
2020 INPUT "Enter draw coords ";
b;" ";c
2030 IF b<0 OR b>255 OR c<0 OR c
>175 THEN GO TO 2020
2040 DRAW OVER a;FN x(b),FN y(c)
2050 GO TO 100
2999 REM **instructions**
3000 FOR m=1 TO 11: FOR n=0 TO 1
3010 PRINT TAB RND*7; PAPER 2; I
NK 7; FLASH n;"ZX Designer"
3015 BEEP .025,m*2
3020 NEXT n: NEXT m
3030 PRINT AT 9,20;"Press any ";
AT 11,20;"letter to";AT 13,20;"c
ontinue"
3040 BEEP .025,0: PAUSE 3: IF IN
KEY$="" THEN GO TO 3040
3050 CLS
3060 PRINT AT 5,3;"This program
allows you to "
3070 PRINT "create designs using
PLOT, DRAW and OVER commands. Th
e DRAW command works in absolute
 screen coordinates."
3080 PRINT '"   Please enter BOR
DER colour     (0 to 7) ";
3090 INPUT a: IF a<0 OR a>7 THEN
 GO TO 3090
3100 PRINT a
3110 PRINT '"   Now enter PAPER
colour ";
3120 INPUT b: IF b<0 OR b>7 THEN
 GO TO 3120
3130 PRINT b
3140 PRINT '"   And INK colour "
;
3150 INPUT c: IF c<0 OR c>7 THEN
 GO TO 3150
3160 PRINT c
3170 PRINT '"   Further instruct
ions are     given in the INPUT
requests. Press any letter to co
ntinue."
3180 PAUSE 0
3190 BORDER a: PAPER b: INK c: C
LS
3200 GO TO 100
```

Britain's Best Magazine For The Sinclair User

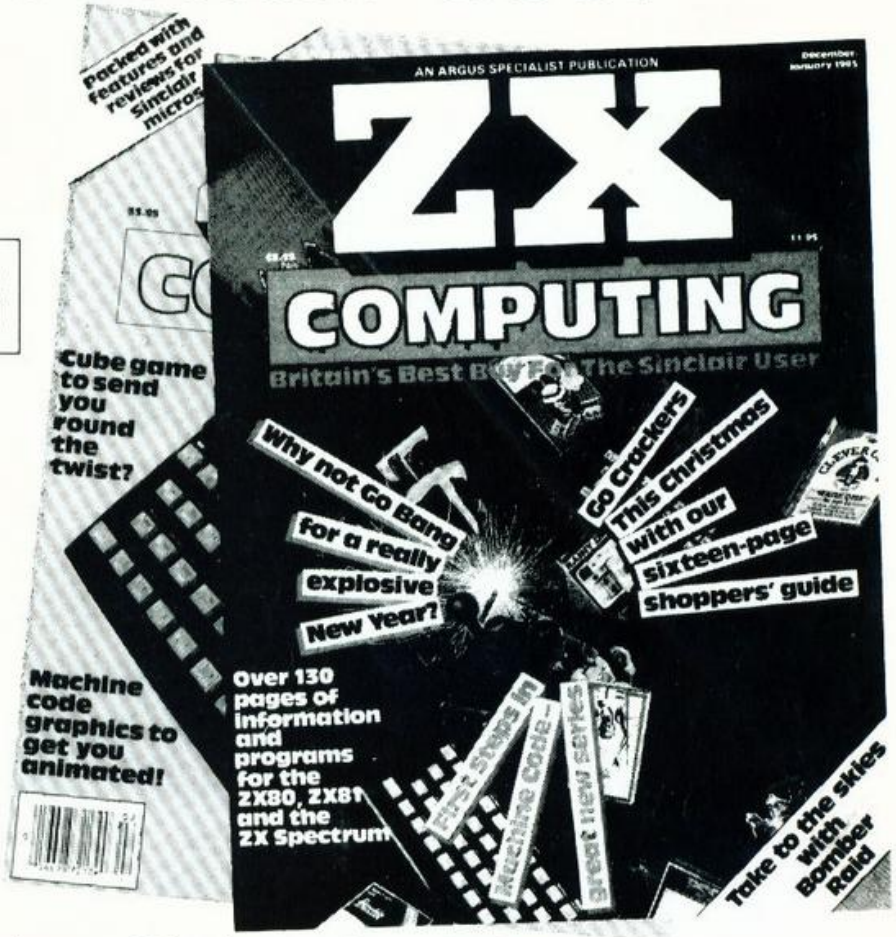
ZX COMPUTING

Don't be disappointed — make sure you get every issue of *ZX Computing!*

Just £13.50 will ensure that the next six issues will be lovingly wrapped and posted to you. Just fill in the form below, cut it out and send it with your cheque or postal order (made payable to ASP Ltd) to:

ZX Computing Subscriptions,
Infonet Ltd,
Times House,
179 The Marlowes,
Hemel Hempstead,
Herts HP1 1BB,

Alternatively you can pay with your Access or Barclaycard: simply fill in your card number, sign the form and send it off. Do NOT send your card!



Make the most of your ZX Computer with *ZX Computing* — bi-monthly!

Cut out and SEND TO:

ZX COMPUTING

ZX Computing Subscriptions,
Infonet Ltd,
Times House,
179 The Marlowes,
Hemel Hempstead,
Herts HP1 1BB,

SUBSCRIPTION RATES

(tick ☐ as appropriate)

£13.50 for six issues	☐
£15.00 for six issues	☐
£31.00 for six issues	☐
$19.50 for six issues USA surface mail.	☐

Subscription Order Form

Please commence my subscription to ZX Computing with the very next issue.

I am enclosing my (delete as necessary) cheque/Postal Order/International Money Order for £: . . . . . (made payable to ASP Ltd)
OR
Debit my Access/Barclaycard (*delete as necessary)*

We welcome Access

VISA

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Please use BLOCK CAPITALS and include post codes.

NAME (Mr/Mrs/Miss) . . . . . .

ADDRESS . . . . . .

. . . . . .

. . . . . **POSTCODE** . . . . . .

Signature . . . . . .

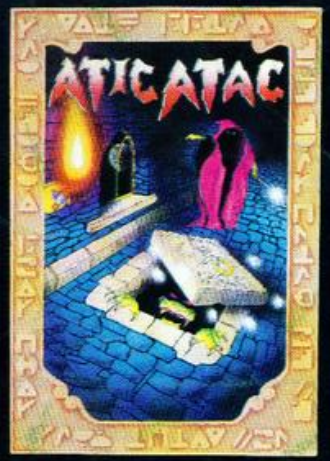
Date . . . . . .

ULTIMATE
PLAY THE GAME

48K SINCLAIR SPECTRUM
AMSTRAD
BBC MODEL B

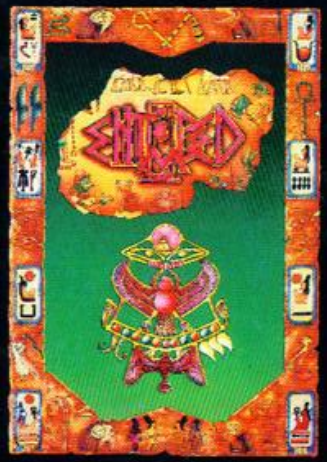
48K SINCLAIR SPECTRUM
BBC MODEL B

48K SINCLAIR SPECTRUM
BBC MODEL B

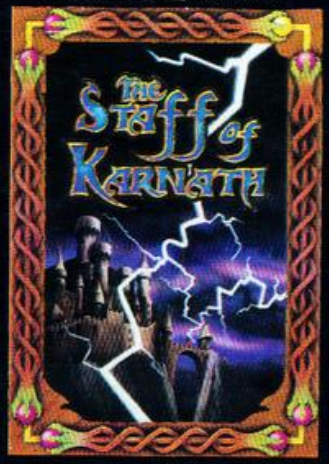


48K SINCLAIR SPECTRUM
AMSTRAD
BBC MODEL B

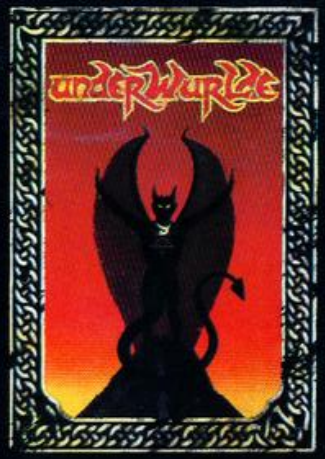
COMMODORE 64



COMMODORE 64



48K SINCLAIR SPECTRUM



AMSTRAD CPC 464

"ALIEN 8", "KNIGHTLORE", "UNDERWURLDE", "SABRE WULF", "ENTOMBED", & "STAFF OF KARNATH" recommended retail price £9.95 inc VAT. "ATIC ATAC" recommended retail price £7.95 inc VAT. Available from W.H.SMITHS, BOOTS, J.MENZIES, WOOLWORTHS and all good software retail outlets. Also available from ULTIMATE PLAY THE GAME, The Green, Ashby-de-la-Zouch, Leicestershire LE6 5JU (P&P are included) Tel: 0530 411485