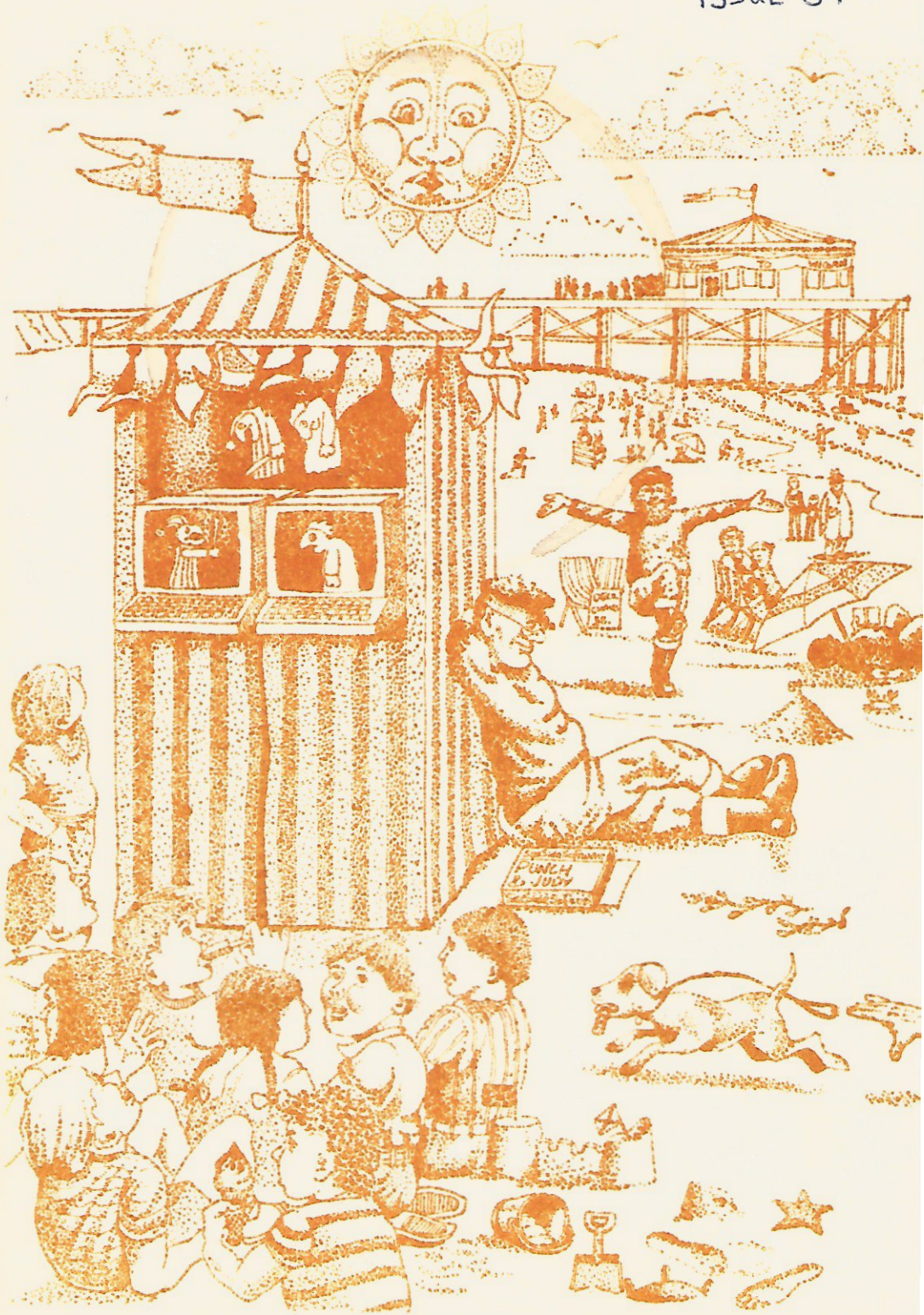
ISSUE 8.



C O N T E N T S

Illustrations as always by our very own John Trippick

SHARPSOFT USER NOTES

Issue No. 8

Last year we introduced the language FORTH to readers via these User Notes. PASCAL is likely to be the language which achieves the greatest usage growth by microcomputer hobbyists during 1983. Why does a language suddenly become fashionable? The answer is probably because it becomes available at a reasonable cost to the small computer user. A new language must, of course, also offer to the user something which existing software lacks. PASCAL offers a modern structured approach to programming which coupled with significantly higher execution speed, when compared to interpretive BASIC, represents an impressive step forward for the home computer user.

Our intention has always been to introduce a tape based compiled version of PASCAL for use by our User Notes readers. Indeed we even developed a small PASCAL subset compiler running from disk. A few months ago one of our readers wrote to us pointing out that a British Software House called Hisoft had already developed a good quality low cost tape/disk compiler for use on Z80 microcomputers.

On learning of the Hisoft PASCAL compilers we evaluated tne version of the product which runs on the MZ-80K, MZ-80A and MZ-80B computes. Our overall impression is that these tape and disk compilers are an outstanding achievement. They represent a significant advance in system software for the Sharp computer range. One of the main articles in this issue, describes the different implementations of PASCAL which are available on Sharp computers. We hope that this article will supply you with background information and help those who are interested in PASCAL, decide on which PASCAL to choose for your system.

Computer graphics is a topic which has universal appeal. Mr. John Simpson has written for us a superb article on using the MZ-80K graphics. This article includes a number of original ideas which to our knowledge have not been published before. We have also asked John if he could write a second article on the user of the PEEK statement in computer graphics - this we hope to publish in our next issue. We would like to make it clear to readers that the information and charts contained in the graphics article are copyright by Mr. J. Simpson.

This issue also includes the regular columns on FORTH, the MZ-80B, CP/M and your articles, letters and programs.

Issue 9 of the User Notes will be published towards the end of November, 1983.

MIKE BRINSON

Editor

FORTH - Bugs, comments and programs

by

F.W. Gair

I have derived a great deal of pleasure, as beginner, from my attempts to master the MZ-80K FORTH you kindly sent to me last year. I wonder if SUN readers would be interested in some aids I have derived to help the understanding of how this fascinating language works.

The kit, which is on four screens loads from DECIMAL 25 LOAD and gives the following facilities--

' (Name) SORT Example 1

gives type of word

' (Name) DEF Example 2

gives type of word and if the word is a Colon definition, disassembles the compiled form.

Addr. IDT Example 3

identifies a word from its Code Field Address.

' (Name) NFACODE Example 4

lists memory from the correspondiong Name Field Address together with ASCII. symbols. Any address followed by ADCODE has a similar result. A Hex base produces the neatest output.

Two further words, NAMAD and DICLIST list N.F.A.'s and the dictionary respectively. All alpha-numeric characters are reproduced.

Colon definitions that end ;S conclude with END. FOUND NAME at the end of a disassembled list means that the word has terminated at another word e.g. like COLD.

The nameless word X, used to exit from the text interpreter, can be revealed by 1D3D DEF.

Some difficulty exists with the word QUIT since it is not possible to distinguish between embedded or closing useages. The literal 212B, which happens to be the C.F.A. of QUIT, can also produce confusion. If disassembly stops affter QUIT it can be restarted by (last addres + 2)DC. Garbage indicates that the QUIT was terminal. If QUIT occurs without printing the word QUIT, i.e. flashing cursor-no OK, restart by (last address) + 4 DC. All listing is stopped by Shift Break.

SKIPOVER causes the printer to skip four lines over the paper perforations during long listings.

```
SCR # 25
  0 ( FORTH ANALYSIS KIT)
  1 : ?LBIT DUP 224 AND 128 - IF DUP . SPACE THEN ;
  2 : CATCH DUP 32 - 0< IF DROP SPACE ELSE DUP 127 =
  3 IF DROP SPACE ELSE EMIT THEN THEN ;
  4 ( ?WORD FROM CFA.)
  5 : START BEGIN 1 - DUP 3 - C@ 128 - 0< 0= UNTIL 3 - ;
  6 : NAME START DUP DUP 1+ SWAP C@ ?LBIT 31 AND 1 - TYPE ;
  7 : FINAL DUP C@ 31 AND + C@ 128 - EMIT  ;
  8 ( FIND NFA'S) : Q NAME FINAL 2 SPACES ;
  9 : ?NAME DUP C@ 31 AND + 1+ @ DUP ROT - 0= IF CR DROP
 10 DROP DROP ." FOUND NAME " SP! QUIT THEN OVER SWAP DUP 0= ;
 11 : FINDWORD LATEST BEGIN ?NAME UNTIL DROP ;
 12 ( M/C'S) : Z80 1 - BEGIN 1+ DUP C@ DUP . 195 - 0= UNTIL
 13 DUP 1+ C@ . 2+ C@ . ; : IBA CR DUP U. 8 0 DO DUP I + C@
 14 3 .R LOOP ; : ABA 2 SPACES 8 0 DO DUP I + C@ CATCH LOOP 8 + ;
 15 : ADCODE BEGIN IBA ABA ?TERMINAL UNTIL DROP CR ; -->
OK


26 LIST
SCR # 26
  0 ( SORT TYPE OF WORD)
  1 : ALTER 2 - DUP @ DUP ;
  2 : SORT ALTER 5905 - 0= IF ." COLON " DROP DROP ELSE
  3 DUP 5971 - 0= IF ." CONSTANT " DROP DROP ELSE
  4 DUP 5997 - 0= IF ." VARIABLE " DROP DROP ELSE
  5 DUP 6015 - 0= IF ." USER VARIABLE " DROP DROP ELSE
  6 SWAP - 2 - 0= IF ." PRIMITIVE " ELSE ." OTHER TYPE ? "
  7 THEN THEN THEN THEN THEN ;
  8 : DESCR DUP @ 2 + 2 SPACES 24 4465 C! SORT ;
  9 : NFACODE NFA ADCODE ; ( SPECIAL WORDS)
 10 : DECODE . SPACE @ Q DESCR 2+ CR ." M/C " Z80 QUIT ;
 11 : OUTPUT ." M/C E1 C5 4D CD 0A 28 C1 C3 45 12 " DROP DROP ;
 12 : WRITER 7300 - 0= IF . SPACE 7259 Q 2 SPACES 24 4465 C!
 13 ." COLON " CR DROP 2+ DUP DUP THEN ;
 14 : HEAT 8668 - 0= IF DROP DROP CR ." M/C " Z80 QUIT THEN ;
 15 ( DISCOMPILER ' WORD DEF) -->
OK


27 LIST
SCR # 27
  0 : WITHDATA DUP 2+ ROT ROT . SPACE @ Q @ . DESCR 2+ ;
  1 : DC CR BEGIN CR DUP DUP DUP WRITER DUP HEAT
  2 DUP @ 4694 - 0= IF WITHDATA ELSE
  3 DUP @ 4730 - 0= IF WITHDATA ELSE
  4 DUP @ 4754 - 0= IF WITHDATA ELSE
  5 DUP @ 5447 - 0= IF . SPACE @ Q DESCR CR ." END " SP! QUIT
  6 ELSE DUP @ 7259 - 0= IF . SPACE DUP @ Q SPACE DESCR SPACE
  7 DUP 3 + SWAP 2+ C@ DUP . SPACE SWAP OVER TYPE 1+ + ELSE
  8 DUP @ 4776 - 0= IF WITHDATA ELSE DUP @ 7042 - 0= IF DUP 7070
  9 - 0= IF DUP . SPACE ." (;CODE)              COLON " ELSE DECODE
 10 THEN ELSE DUP @ 10447 - 0= IF OUTPUT ELSE
 11 FINDWORD . SPACE @ Q DESCR THEN THEN THEN THEN THEN THEN
 12 THEN THEN DUP @ 8491 - 0= IF DROP QUIT THEN 2+
 13 ?TERMINAL UNTIL SP! ;
 14 ( DEFINE WORD) : DEF DUP 2 - @ 5905 - IF SORT
 15 ELSE DC THEN ; -->
OK
```

```
28 LIST
SCR # 28
  0 ( FIND NAME / CFA)
  1 : IDT LATEST BEGIN 1 TRAVERSE DUP 3 + ROT DUP ROT - 0=
  2 IF SWAP 3 + @ DROP QUIT THEN SWAP 1+ @ DUP 0= UNTIL
  3 DROP DROP ." NOT A C.F.A.  " ;
  4 CREATE PRINT 62 , 4120 , 54257 , 65279 , 51712 , 4677 , 32830 ,
  5 65235 , 318 , 6424 , 22489 , 1 , 56064 , 59134 , 47629 , 800 ,
  6 6361 , 7905 , 7440 , 64800 , 30731 , 8369 , 55788 , 343 ,
  7 0 , 65243 , 3558 , 8378 , 55558 , 54191 , 6398 , 7873 , 7440 ,
  8 64800 , 30731 , 8369 , 233 , SMUDGE
  9 : SKIPOVER 0 3328 1024 19968 6912 PRINT ;
 10 CREATE BEEP 205 C, 62 , 195 C, 4677 , SMUDGE
 11 : NOISE 12 0 DO BEEP LOOP ; : NAMAD LATEST 3 SPACES BEGIN DUP
 12 C@ 31 AND + 1+ @ DUP . DUP 0= IF DROP QUIT THEN ?TERMINAL
 13 UNTIL DROP ; : DICLIST LATEST CR BEGIN CR DUP DUP C@ 31 AND
 14 + 1+ SWAP . SPACE DUP DUP 2+ @ 3 SPACES 4 + SORT @ DUP 0=
 15 IF DROP QUIT THEN ?TERMINAL UNTIL DROP ; NOISE ;S
OK
```

```
' DENSITY SORT  VARIABLE OK
' COLD SORT  COLON OK
' SWAP SORT  PRIMITIVE OK
' FIRST DEF  CONSTANT OK
' R/W DEF
```

Ex 1

```
26E9  USE     VARIABLE
26EB  @     PRIMITIVE
26ED  >R     PRIMITIVE
26EF  SWAP     PRIMITIVE
26F1  SEC/BLK     CONSTANT
26F3  *     COLON
26F5  ROT     PRIMITIVE
26F7  USE     VARIABLE
26F9  !     PRIMITIVE
26FB  SEC/BLK     CONSTANT
26FD  0     CONSTANT
26FF  (DO)     PRIMITIVE
2701  OVER     PRIMITIVE
2703  OVER     PRIMITIVE
2705  T&SCALC     PRIMITIVE
2707  SET-IO     PRIMITIVE
2709  OBRANCH  8   PRIMITIVE
270D  SEC-READ     PRIMITIVE
270F  BRANCH  3A   PRIMITIVE
2713  CR     PRIMITIVE
2715  (.")      COLON  12  PRESS STOP THEN CR
272A  KEY     PRIMITIVE
272C  DROP     PRIMITIVE
272E  SEC-WRITE     PRIMITIVE
2730  CR     PRIMITIVE
2732  (.")      COLON  12  PRESS STOP THEN CR
2747  KEY     PRIMITIVE
```

Ex 2

```
2749  DROP     PRIMITIVE
274B  1+     COLON
274D  LIT  80    PRIMITIVE
2751  USE     VARIABLE
2753  +!     PRIMITIVE
2755  (LOOP)  -56    PRIMITIVE
2759  DROP     PRIMITIVE
275B  DROP     PRIMITIVE
275D  R>     PRIMITIVE
275F  USE     VARIABLE
2761  !     PRIMITIVE
2763  ;S     PRIMITIVE
END
```

```
 ' DLITERAL DEF

2012  STATE     USER VARIABLE
2014  ∋     PRIMITIVE
2016  OBRANCH  8    PRIMITIVE
201A  SWAP     PRIMITIVE
201C  C7  LITERAL     COLON
201E  C7  LITERAL     COLON
2020  ;S     PRIMITIVE
END
```

— length byte C7

```
 24DD IDT  EMPTY-BUFFERS
24DF IDT  NOT A C.F.A.  OK

1403 IDT  ?TERMINAL
1404 IDT  NOT A C.F.A.  OK
```

Ex 3

```
   LIMIT NFACODE
17CA  85 4C 49 4D 49 D4 BE 17  ⌴LIMI╢¥
17D2  53 17  0 90 85 42 2F 42  S  ₧⌴B/B
17DA  55 C6 CA 17 53 17  0  4  U→ ‼ S
17E2  85 42 2F 53 43 D2 D6 17  ⌴B/SC┬ ‼
17EA  53 17  1  0 87 2B 4F 52  S    ¯+OR
17F2  49 47 49 CE E2 17 11 17  IGI┐ ‡
17FA  56 12  0 12 C9 15 47 15  V    ℙ G
1802  82 53 B0 EE 17 7F 17  6  ¯Sn⁄
180A   0 82 52 B0  2 18 7F 17   ¯Rn
1812   8  0 83 54 49 C2  B 18    ⌐TI▬
181A  7F 17  A 85 57 49 44 54      ⌴WIDT
1822  C8 14 18 7F 17  C 87 57  ▒      ¯W
182A  41 52 4E 49 4E C7 1D 18  ARNIN ‡
1832  7F 17  E 85 46 45 4E 43      ⌴FENC
183A  C5 28 18 7F 17 10 82 44  ‡ (     ¯D
1842  D0 35 18 7F 17 12 88 56  ⌐5     ¯V
184A  4F 43 2D 4C 49 4E CB 40  OC-LIN ╦@
1852  18 7F 17 14  0 83 42 4C        ⌐BL
185A  CB 48 18 7F 17 16 82 49  ╦H     ¯I
1862  CE 57 18 7F 17 18 83 4F  ┐W     ⌐O
OK
```

Ex 4

```
NAMAD      330D 32F3 32E4 32C1 326D 321A 31F6 3091 306A 3042 2FFE 2FC8 2FA2 2F
2F6C 2EA8 2E92 2E77 2E4D 2E25 2DF3 2DDA 2D90 2D80 2D5E 2D39 2D0F 2CD7 2CB3 2C
2C83 2C3D 2C05 2BBB 2BB0 2B66 2B59 2B4D 2B41 2B32 2B21 2AFD 2AE6 2ABC 2AA3 2A
2A7D 2A5C 2A4C 2A2B 2A16 29F9 29E3 29D7 29C1 29AB 2996 2985 2978 295C 294A 29
28FA 28E6 27B6 2783 2765 26E1 26C5 26AA 2679 2648 262B 260F 25AE 251D 24FB 24
24CD 24AE 2487 2476 2468 245C 244E 2443 2439 242D 2423 2417 2404 23F0 23B3 23
OK
```

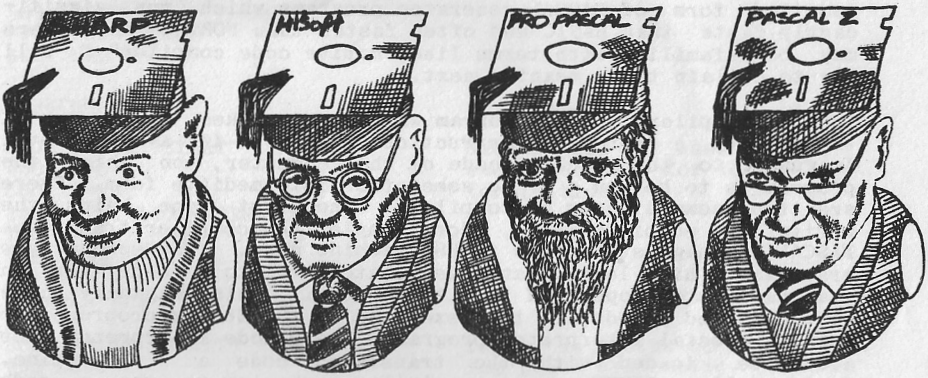
```
DICLIST

3347  DICLIST      COLON
330D  NAMAD       COLON
32F3  NOISE       COLON
32E4  BEEP      PRIMITIVE
32C1  SKIPOVER      COLON
326D  PRINT     PRIMITIVE
321A  IDT       COLON
31F6  DEF       COLON
3091  DC      COLON
306A  WITHDATA      COLON
3042  HEAT      COLON
2FFE  WRITER      COLON
2FC8  OUTPUT      COLON
2FA2  DECODE      COLON
2F90  NFACODE     COLON
2F6C  DESCR       COLON
2EA8  SORT        COLON
2E92  ALTER       COLON
2E77  ADCODE      COLON
2E4D  ABA       COLON
2E25  IBA       COLON
2DF3  Z80       COLON
2DDA  FINDWORD      COLON
2D90  ?NAME       COLON OK
```

PASCAL for the SHARP MZ-80K, MZ-80A and MZ-80B

by

Mike Brinson



In the past very little space has been set aside in these User Notes for the programming language PASCAL. Last year we introduced FORTH - 1983 looks like the year when PASCAL will become more widely known and used by microcomputer enthusiasts.

PASCAL has of course been available to Sharp users for well over a year; since Sharp introduced their tape based PASCAL interpreter. Last October a British Software House called Hisoft released Version 4 of their PASCAL compiler - this is now available as a tape or disk based product for Sharp computers. Before launching into the details of the Hisoft PASCAL compiler some background comments regarding PASCAL and compilers seems appropriate.

Throughout the lifetime of the Sharp microcomputer products BASIC has held its place as the most used high-level programming language. A high percentage of our readers have had their first introduction to computing via BASIC. Today's BASIC is a highly developed interactive programming environment which is suitable for both beginners and the more experienced practioners of the programming art.

In spite of all its good points interpretive BASIC has a number of deficiencies, the obvious being its slow operating speed and poor subroutine structure. Speed may not be important to the hobbyist - Does it worry you if your computer takes 60 seconds rather than 6 seconds to complete a task? There are however, some types of program where speed is important - games are an obvious case. If you have to sit and wait for a long time while the computer calculates its move then the impact of the game is lost.

BASIC's lack of speed is not a property of the language but rather a function of how the language has been implemented on a microcomputer. Unfortunately, to our knowledge, a tape compiled form of BASIC has not been developed for the Sharp range of computers. If any readers know of a tape BASIC compiler we would appreciate a letter with the details.

FORTH overcomes the speed deficiency of BASIC but is much more difficult to learn and use than BASIC. New programmers usually find FORTH very hard to learn and soon become frustrated with the language. PASCAL offers an alternative language - although more difficult to learn than BASIC it is easier for beginners to understand than FORTH. Also the native code compiled form of PASCAL generates programs which run significantly faste than BASIC and often faster than FORTH. As readers may be unfamiliar with terms like "native code compiler" I will try to explain their meaning next.

A compiler is a program which undertakes the task of translating a set of instructions written in a high-level language to the machine code of the computer, on which the program is to be run, or to some other intermediate form. There are two common forms of compiler. The first type being the native code compiler where, for example, the compiler translates PASCAL language statements to Z80 machine code. The second type translates high-level language statements to a code for an imaginary stack operated 16 bit computer. This imaginary code (often called p code) is then executed by the Z80 microprocessor using a special interpreter program. The p code interpreter must also be loaded with the translated code at run time. Incidentally, FORTH uses a similar mechanism to the p code concept.

In general the program generated by a native code compiler will execute MUCH faster than that produced by a p-code compiler. However, a p code compiler is usually more portable than a native code compiler because only the run time interpreter has to be rewritten if the compiler is moved to a microcomputer which uses a different microprocessor. A very well known form of PASCAL uses the p code approach. Most readers who are interested in PASCAL will probably have met the term UCSD PASCAL from the popular magazines. UCSD PASCAL is a p code compiled form of PASCAL which has been implemented on most popular 8 and 16 bit microprocessor based computers - the SAGE II 68000 computer has received wide publicity recently.

Program development using a compiler involves more stages than using interpretive BASIC. The program must first be entered into the computer using an EDITOR. This editor will be similar in structure and editing commands to the editor which is used to prepare an assembly language program. The editor will also allow you to save your program on tape or disk. The saved program is called a source program or source file. It is this source file which is input to the compiler for translation to machine code. The translation process may involve a number of different and unique stages. Eventually, a machine code program equivalent to the original high-level program will be generated by the compiler for exececution or storage on tape or disk. If during the translation from PASCAL to machine code an error is found by the compiler then the editor must be used to correct the error. Tracing and removing errors can be tedious! However, once a program compiles, it can be saved as an object code file and run simply by loading it and executing.

When composing the execution speed of a high-level language it is useful to have a relative series of objective tests. One test that has become quite fashionable is to test the numeric operations of the BASIC language. The following benchmarks are regularly used by reviewers of machines writing for "Personal Computer World". These benchmarks test most of the fundamental BASIC constructions and a number of the numeric functions.

Program BM1

```
300  PRINT "START"
400  FOR K=1 TO 1000
500  NEXT K
700  PRINT "END"
800  END
```

notes

Tests a simple FOR-NEXT loop

Program BM2

```
300  PRINT "START"
400  K=0
500  K=K+1
600  IF K<1000 THEN 500
700  PRINT "END"
800  END
```

Same as BM1 but uses an IF statement test for loop termination.

Program BM3

```
300  PRINT "START"
400  K=0
500  K=K+1
510  A=K/K*K+K-K
600  IF K<100 THEN 500
700  PRINT "END"
800  END
```

Tests the inclusion of floating point arithmetic. Note - using defined variable K in statement 510.

Program BM4

```
300  PRINT "START"
400  K=0
500  K=K+1
510  A=K/2*3+4-5
600  IF K<1000 THEN 500
700  PRINT "END"
800  END
```

Same as in BM3 but the interpreter has to convert the decimal numbers in statement 510 each time the loop is traversed.

Program BM5

```
300  PRINT "START"
400  K=0
500  K=K+1
510  A=K/2*3+4-5
520  GOSUB 820
600  IF K<1000 THEN 500
700  PRINT "END"
800  END
820  RETURN
```

Lines 520 and 820 added to test the speed of GOSUB and RETURN statements.

Program BM6

```
300  PRINT "START"
400  K=0
430  DIM M(5)
500  K=K+1
510  A=K/2*3+4-5
520  GOSUB 820
530  FOR L=1 TO 5
540  NEXT L
600  IF K<1000 THEN 500
700  PRINT "END"
800  END
820  RETURN
```

notes

Lines 430, 530 and 540 added to create array M add an extra FOR-NEXT loop.
Note the FOR-NEXT loop is executed on each pass of the outer loop.

Program BM7

```
300  PRINT "START"
400  K=0
430  DIM M(5)
500  K=K+1
510  A=K/2*3+4-5
520  GOSUB 820
530  FOR L=1 TO 5
535  M(L)=A
540  NEXT L
600  IF K<1000 THEN 500
700  PRINT "END"
800  END
820  RETURN
```

Same as BM6 but statement 535 added to test the time it takes to assign a value to an array element.

Program BM8

```
300  PRINT "START"
400  K=0
500  K=K+1
530  A=K 2
540  B=LN(K)
550  B=SIN(K)
600  IF K<100 THEN 500
700  PRINT "END"
800  END
```

Tests a number of the BASIC numeric functions. NOTE - the loop termiantes at K=100 NOT 1000.

Typical benchmark test results, in seconds, for the SHARP computer range running SHARBASIC (tape versions).

Model	BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8
MZ-80K (Z80 2MHz clock)	1.4	9.4	16.3	22.4	25.4	36.8	51.1	10.2
MZ-80A (Z80A 2MHz clock)	1.5	9.2	16.4	22.8	25.6	37.7	55.0	10.1
MZ-80B (Z80A 4MHz clock)	0.6	4.5	8.5	11.5	13.0	19.0	27.5	5.0
PC3201 (Z80A 2.5MHz clock?)	4.0	13.5	35.5	35.5	38.5	67.0	108.00	25.00

These execution times indicate that Sharp BASIC is a highly optimised interpretive form of BASIC. Looking through the magazines one finds that these benchmark times compare favourably with other Z80 and Z80A microcomputers. An interpreter executes programs slower than object code generated by a compiler. the reasons why this statement is true may not be clear to some readers. The Sharp BASIC interpreter NEVER translates the BASIC program statements into machine code but, as the name implies, interprets the action required from the structure of the statements. Hence, for example, during the execution of a program loop the BASIC program statements are re-interpreted on each pass through the loop. This process adds a considerable time overhead slowing down the execution of the program. The execution of the program is also slowed down due to the fact that all numbers are held internally in floating point format - numbers in 16 bit integer format have not been implemented by Sharp. The benchmark times are therefore dependent on the precision of the floating point package which is included in a BASIC interpreter. Note the PC3201 BASIC includes a higher precision floating package than the MZ-80K, A or B BASIC and hence is slower, although the computer's clock rate is higher than the MZ-80K and MZ-80A.

The important point to remember about benchmark tests is that the results on their own do not tell us much. For comparison purposes we need to know the test conditions, for example the Z80 clock rate and the precision of the floating point package. Benchmarks BM1 to BM8 do however give us a starting datum when we compare the relative speeds of PASCAL and BASIC. A second series of benchmark tests have been developed by "Personal Computer World" for testing features of the PASCAL language which are not found in BASIC. These programs are given below.

```
PROGRAM MAGNIFIER;
VAR K : INTEGER;
BEGIN
  WRITELN('S');
  FOR K:=1 TO 10000 DO;
  WRITELN('F')
END.
```

```
PROGRAM FORLOOP;
VAR J, K : INTEGER;
BEGIN
  WRITELN('S');
  FOR K:=1 TO 10000 DO
     FOR J:=1 TO 10 DO;
  WRITELN('F')
END.
```

```
PROGRAM WHILELOOP;
VAR J,K:INTEGER;
BEGIN
  WRITELN('S');
  FOR K:=1 TO 10000 DO
  BEGIN
     J:=1;
     WHILE J<=10 DO J:=J+1
  END;
  WRITELN('F')
END.
```

```
PROGRAM LITERALASSIGN;
VAR J,K,L:INTEGER;
BEGIN
  WRITELN('S');
  FOR K:=1 TO 10000 DO
     FOR J:=1 TO 10 DO L:=0;
  WRITELN('F')
END.
```

```
PROGRAM MEMORYACCESS;
VAR J,K,L:INTEGER;
BEGIN
  WRITELN('S');
  FOR K:=1 TO 10000 DO
     FOR J:=1 TO 10 DO L:=J;
  WRITELN('F')
END.
```

```
PROGRAM REALARITHMETIC;
VAR K:INTEGER;
    X:REAL;
BEGIN
  WRITELN('S');
     FOR K:=1 TO 10000 DO
     X:=K/2*3+4-5;
  WRITELN('F')
END.
```

```
PROGRAM REPEATLOOP;
VAR J,K:INTEGER;
BEGIN
  WRITELN('S');
  FOR K:= 1 TO 10000 DO
  BEGIN
     J:=1;
     REPEAT
       J:=J+1
     UNTIL J>10;
  END;
  WRITELN('F')
END.

PROGRAM EQUALIF;
VAR J,K,L:INTEGER;
BEGIN
  WRITELN('S');
  FOR K:=1 TO 10000 DO
     FOR J:=1 TO 10 DO
        IF J<6 THEN L:=1
               ELSE L:=0;
  WRITELN('F')
END.

PROGRAM NOPARAMETERS;
VAR J,K : INTEGER;
PROCEDURE NONE5;
BEGIN
  J:=1
END;
PROCEDURE NONE4;
BEGIN
  NONE5
END;
PROCEDURE NONE3;
BEGIN
  NONE4
END;
PROCEDURE NONE1;
BEGIN
  NONE2
END
BEGIN
WRITELN('S');
  J:=0;
  FOR K:=1 TO 1000 DO NONE1;
WRITELN('F')
END.
```

```
PROGRAM REALALGEBRA;
VAR K:INTEGER;
    X:REAL;
BEGIN
  WRITELN('S');
  FOR K:=1 TO 10000 DO
     X:=K/K*K+K-K;
  WRITELN('F')
END.

PROGRAM UNEQUALIF;
VAR J,K,L:INTEGER;
BEGIN
  WRITELN('S');
  FOR K:=1 TO 10000 DO
     FOR J:=1 TO 10 DO
     IF J<2 THEN L:=1
     ELSE L:=0;
  WRITELN('F')
END.

PROGRAM VALUE;
VAR J,K : INTEGER;
PROCEDURE VALUE5(I:INTEGER);
BEGIN
  I:=1
END;
PROCEDURE VALUE4(I:INTEGER);
BEGIN
  VALUE5(I)
END;
PROCEDURE VALUE3(I:INTEGER);
BEGIN
  VALUE4(I)
END;
PROCEDURE VALUE2(I:INTEGER);
BEGIN
  VALUE3(I)
END;
PROCEDURE VALUE1(I:INTEGER);
BEGIN
  VALUE2(I)
END;
BEGIN
WRITELN('S')
  J:=0;
  FOR K:=1 TO 10000 DO
  VALUE1(J);
WRITELN('F')
END.
```

```
PROGRAM REFERENCE;
VAR J,K:INTEGER;
PROCEDURE REFER5(VAR I:INTEGER);
BEGIN
  I:=1
END;
PROCEDURE REFER4(VAR I:INTEGER);
BEGIN
  REFER5(I)
END;
PROCEDURE REFER3(VAR I:INTEGER);
BEGIN
  REFER4(I)
END;
PROCEDURE REFER2(VARI:INTEGER);
BEGIN
REFER3(I)
END;
BEGIN
  WRITELN('S');
  J:=0;
  FOR K:=1 TO 10000 DO
  REFER1(J);
  WRITELN ('F');
END.
```

Even if you are unfamiliar with PASCAL I am sure you will be able to spot the similarity between a number of the BASIC and PASCAL benchmark programs. The following benchmark programs are roughly equivalent:-

BASIC	**PASCAL**
BM1	MAGNIFIER
BM3	REALALGEBRA
BM4	REALARITHMETIC

Notice however, that the PASCAL loop has been increased to 10000. This is done to allow accurate measurement of the benchmark times using a stopwatch.

GROUP ONE PASCAL Compilers and interpreters for the SHARP Computers

Name	Source	Machine	Type	Tape/disk
PASCAL	SHARP	MZ-80K/A & MZ-80B	Interpreter	Tape
HP4T	HISOFT	MZ-80K/A & MZ-80B*	Compiler (Native-code)	Tape
HP4D	HISOFT	MZ-80K/A & MZ-80B	Compiler (Native-code)	Disk CP/M
PROPASCAL	PROSPERO	MZ-80B	Compiler (Native-code)	Disk CP/M
PASCAL/Z	ITHACA INTERSYSTEMS	MZ-80B	Compiler (Native-code)	Disk CP/M

* By the time this edition of the User Notes reaches you a tape version of HP4T should be available for the MZ-80B. Write tc SHARPSOFT for the latest information on the release date.

GROUP TWO

Other PASCAL translators

Name	Source	Type	Tape/disk
PASCAL MT+	Digital Research	Compiler (Native code)	disk CP/M
UCSD PASCAL		Compiler (p code)	disk own operating system
JRT PASCAL	JRT	Compiler (p code)	disk CP/M
PASCAL/M		Compiler (p code)	disk CP/M

Typical Benchmark test results, in seconds, for PASCAL running on Z80 systems

Benchmark* Program	Hisoft Pascal4	Propascal	Pascal/Z	PASCAL MT+
Magnifier	0.2	0.27	2.4	0.2
Forloop	2.7	2.45	29.3	4.7
Whileloop	4.3	5.3	29.9	7.8
Repeatloop	3.6	4.2	29.3	6.9
Literalassign	3.4	4.6	30.3	5.5
Memoryaccess	3.5	4.3	31.4	5.7
Realarithmetic	14.8	13.0	192.9	59
Realalgebra	15.3	25.3	127.9	45
Vector	8.0	9.4	51.6	10.8
Equalif	5.9	6.0	33.9	11.2
Unequalif	5.9	5.9	33.4	11.2
Nonparameters	3.2	8.0	13.7	0.9
Value	3.8	8.7	14.2	3.4
Reference	3.8	8.8	15.0	3.4

* 4MHz clock

One observation can be made concerning these figures - namely that in general the code produced by a native code PASCAL compiler executes approximately TEN times faster than intepretive BASIC. One further benchmark test yields figures which allow comparison between assembly code, BASIC and PASCAL. This test is used by BYTE magazine and is based on the well know Eratothenes sieve technique for calculating Prime numbers.

```
PROGRAM prime;

(* Eratosthenes Sieve prime number program *)

CONST
     Size = 8190;
VAR
     flags : ARRAY [0..size] OF BOOLEAN;
     i, prime, k, count, item : INTEGER;
BEGIN
  WRITELN('10 iterations');
  FOR item := 1 TO 10 DO
  BEGIN
     count :=0;
     FOR i:=0 TO size DO flags [i]:=TRUE;
     FOR i:=0 TO size DO
     IF flags [i] THEN
  BEGIN
  prime : = i+i+3;
  K:=i + prime;
  WHILE K<=size DO
  BEGIN
  flags [K]:=FALSE;
  K:=K+prime
  END;
  count:=count+1
  END
END
WRITELN (count,'primes')
END.
```

Z80 Benchmark Times, in seconds, for the Sieve Program

1. PASCAL	Time	
PASCAL MT+ (V5.5)	22.7	
PASCAL/Z (V4.0)	31.4	
Prospero PASCAL (V1.4)	13.7	
Hisoft PASCAL (HP4D14)	19.5	
UCSD PASCAL (IV.03)	156.0	
JRT PASCAL (V2.0)	383.0	
PASCAL/M	450.0	
2. Other languages	**Time**	
Microsoft MBASIC (intepreter)	1476.0	(MBASIC is similar in speed to SHARP BASIC)
FORTH (fig)	84.0	
BDS C (V1.46)	39.9	
Z80 Assembler code	6.8	

From these figures we can conclude that a well written PASCAL compiler could be expected to generate Z80 code which is only two to three times slower than "hand" optimised assembly code. Notice also that interpretive BASIC is roughly two orders of magnitude slower than the code produced by the faster PASCAL compilers.

Choosing a PASCAL Compiler

We regularly get requests from Sharp computer users for advice regarding which software to use for a particular application. It is always difficult to give a quick answer to questions of this form. Normally the only way to answer such a question with a satisfactory reply is by firstly studying the application the user has in mind and secondly by evaluating the hardware on which the software is to be used. The following notes may help you decide if PASCAL will be useful for your programming applications. If you decide to try PASCAL then the next step should be to choose a compiler.

Learning PASCAL takes more effort than BASIC. However, if you have mastered BASIC then PASCAL is a rewarding challenge. Programmers with no knowledge of PASCAL will find the Sharp PASCAL interpreter has a lot to offer. If you follow Dr. Richard Meadows' new book on Sharp PASCAL, learning PASCAL will be both fast and relatively painless. Incidentally, Richard Meadows' book is structured around a series of introductory PASCAL lessons which make it ideal for beginners.

Readers who have a basic knowledge of PASCAL should consider the Hisoft tape PASCAL. In terms of value for money, it is outstanding. Software of this quality is rarely developed for tape based microcomputers. We are sure that it will become one of your most used software tools and in many cases will even replace assembly code programming. If your funds allow then start with Sharp PASCAL and move on to Hisoft PASCAL.

Readers with a disk based MZ-80A or MZ-80B computer and the CP/M operating system have a much wider choice when it comes to selecting a PASCAL compiler. Again for beginners we recommend the Hisoft disk version of PASCAL. It is fast, memory efficient and again outstanding value for money.

If you are either a professional programmer or a hobbyist using the MZ-80B and write large complex programs then you should evaluate PROPASCAL and PASCAL/Z. Which you decide on will largely depend on your application. If your need is for a fast "number crunching" PASCAL then PROPASCAL is a good choice. However, if you write "systems" programs where the code needs to be hand optimised or programs which include assembly language segments then PASCAL/Z is ideal.

In the end the choice must be made by you but I suspect it will be largely dictated by the hardware you use and the amount of money you have to invest in software.

Hisoft PASCAL has already achieved a high reputation among Sharp computer users. If you have written any programs using this compiler, or indeed any of the other PASCAL compilers which run on the Sharp computers, send them to us for publication in future User Notes. Your feedback and comments on using PASCAL on the Sharp computers will also interest our readers.

SHARP PASCAL

Sharp PASCAL is a tape interpreter version of PASCAL. Although only a subset of the standard PASCAL language it is ideal for learning basic PASCAL. The major omissions or differences to standard PASCAL are:

1. No procedure or function can be declared within another procedure of function declaration.

2. Structured data cannot be used.

3. Only value parameters allowed in procedure and function declarations and calls.

In this implementation of PASCAL a lot of the "spirit" of PASCAL is lost:-

NO CONST or TYPE declarations. NO RECORDS or POINTERS.

It does however include a number of hardware dependent extensions which can be used, for example, to control the music and graphics hardware.

SHARP PASCAL has an important place in the list of Z80 PASCALS because if you are just starting to learn PASCAL it is ideal as an introduction to the language. New PASCAL programmers will find the interactive environment an ideal one to learn the language.

HISOFT PASCAL

The Hisoft package is a true native code compiler for the Z80 microprocessor. The compiler has been adapted by Hisoft for use on a number of the popular Z80 microcomputers, for example the 48K SPECTRUM and the Sharp range of computers. The compiler is supplied as either a tape or disk product. The disk version runs under the CP/M operating system (MZ-80A and MZ-80B only). This is a mature product which has evolved from a subset compiler for the PASCAL language to a compiler which supports nearly al the major features of the language.

In general the tape and disk compilers have identical features. However, one area where there are noticeable differences is FILES. Computers which use tape as their medium for mass storage of data and programs tend to use different techniques for recording both data and programs. Hisoft have overcome the lack of tape standardisation by devising their own standard tape control routines. These are, of course, different from the PASCAL sequential FILE structures.

Hisoft PASCAL 4 is a fast, easy-to-use, powerful version of the PASCAL language as specified in the PASCAL User Manual and Report (Jensen/Wirth, Second Edition). The major omissions from this specification are as follows:-

Disk Version HP4D

1. Only FILES of CHAR are allowed.

2. A RECORD type may not have a VARIANT part.

3. PROCEDURES and FUNCTIONS are NOT valid as parameters.

Tape Version HP4T

1. FILES not supported - data stored and read from tape via special routines.

2. A RECORD type may not have a VARIANT part.

3. PROCEDURES and FUNCTIONS are NOT valid as parameters.

The compiler occupies approximately 12K of memory and the run-time routines an extra 4K. The tape version requires an additional 4K of memory when loaded. This extra storage is needed for a built-in editor.

Using the Hisoft compiler is very straightforward. Either the built-in editor (tape version) or a standard CP/M editor (ED, Wordmaster or Wordstar etc.) is used to enter and correct program text. With the tape version the complete package is held in memory at the same time allowing one to switch between editing, compiling and running a program without reloading different parts of the package from tape. This does tend to limit the size of a program which can be edited, compiled and run on the MZ-80K and MZ-80A. To overcome this limitation the Hisoft package allows segments of a program, usually separate PROCEDURES and/or FUNCTIONS, to be stored on tape. During compilation these segments can be read directly from tape and compiled. This operation is controlled by special commands embedded in the PASCAL comment statement. Once a program has been developed and debugged it can be written to tape as an object code file. The object code program can then be loaded as a stand-alone machine code program using the SHARP tape loader in ROM. Similar facilities are available in the CP/M disk version. However, the disk compiler operates entirely from disk and generates the final machine code program as a COM file. Overall Hisoft PASCAL provides the user with a very flexible, and very practical, interface between the user and his/her hardware.

Although PASCAL is a small language, in comparison to the size of a language like ADA, it is still a major task to implement the entire language within the memory capacity of todays 8-bit microprocesors. Tape mass storage also imparts restrictions on compiler writers when they attempt to implement the FILE section of the language. Hence, the majority of PASCAL compilers have been developed as a compromise between inclusion of the whole of PASCAL and memory size or mass storage medium. The "User Manual and Report" defines the PASCAL language. The environment in which the language is used is largely left to the harware designer and the compiler writer. All of the current PASCAL compilers for the Z80 microprocessor have had language extensions added by their authors. These extensions usually reflect firstly the environment in which the compiler runs and secondly an attempt to correct deficiencies in the language. Extensions normally take the form of predefined PROCEDURES or FUNCTIONS. Hisoft have added features like PEEK and POKE. Other compilers include, for example, a range of string handling PROCEDURES and FUNCTIONS which are similar to those found in BASIC.

The features of the PASCAL language implemented in the Hisoft PASCAL Compiler are more than adequate for the hobbyist programmer. Very sophisticated and powerful programs can be developed using this compiler. The following list gives a brief summary of the features supported.

Disk Version HP4D

1. Reserved Words.

AND	ARRAY	BEGIN	CASE	CONST	DIV	DO
DOWNTO	ELSE	END	FILE	FOR	FORWARD	
FUNCTION	GOTO	IF	IN	LABEL	MOD	NIL
NOT	OF	OR	PACKED	PROCEDURE	PROGRAM	
RECORD	REPEAT	SET	THEN	TO	TYPE	UNTIL
VAR	WHILE	WITH				

2. SPECIAL Symbols

+	-	*	/			
=	<>	<	<=	>=	>	
(	)	[	]	{	}	^
:=	.	,	;	:	'	..

3. Predefined Identifiers.

The following entities may be thought of as declared in a block surrounding the whole program and they are therefore available through the program unless re-defined by the programmer within an inner block.

```
CONST       MAXINT = 32767;

TYPE        BOOLEAN = (FALSE,TRUE);
            CHAR (* The expanded ASCII character set *);
            INTEGER = -MAXINT..MAXINT;
            REAL (* A subset of real numbers
                  - the largest real is 3.4E38,
                    while the smallest real is 5.9E-39*)

VAR         INPUT, OUTPUT : TEXT;

PROCEDURE   WRITE; WRITELN; READ; READLN; RESET; REWRITE; GET;
            PUT;  PAGE;  HALT;  OUT;  USER; POKE; INLINE; NEW;
            MARK; RELEASE;

FUNCTION    ABS;  SQR;  ODD;  RANDOM;  ORD; SUCC; PRED;  INCH;
            EOLN;  EOF; PEEK; CHR; SQRT; ENTIRE; ROUND; TRUNC;
            FRAC;  SIN;  COS; TAN; INP; ARCTAN; EXP; LN; ADDR;
            CPM;
```

Note Tape version HP4T is similal to HP4D but has disk routines replaced by special tape ccommands.

MORE POKES FOR THE MZ-80K

by

Dr. A K Black and Dr. G R Glover

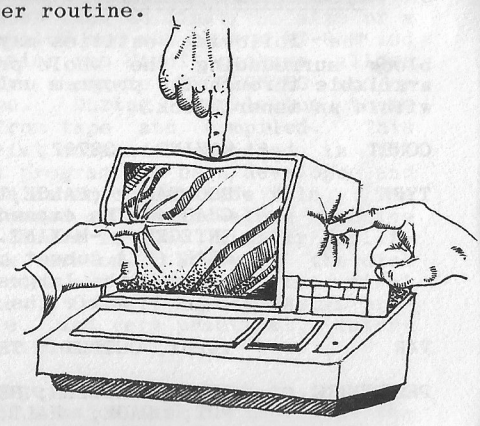
The Authors of our very popular ASTROLOGY Package

Here are some more POKES for the Sharp MZ-80K. Although the User-available memory on this machine is about 34K, after BASIC has been loaded, there are ways of effectively increasing this, without resource to floppy disks.

The techniques may also be applicable to other machines, with the right modifications.

Suppose you wish to analyse a fair amount of data. It is possible to use a program to calculate the necessary data, followed by a storage routine to POKE the data into the top area of RAM. This data can then be stored onto tape, and subsequently retrieved and analysed by another routine.

Suppose you wish to store items of data for 435 individuals. We needed to do just this, when we wanted to calculate 435 horoscopes, and then analyse the results. It takes over three and a half hours to calculate the horoscopes, and the data for the birth-dates occupies a fair amount of memory. We felt that it would be useful to devise a way of storing the data economically on tape, and then to be able to retrieve the data and perform various statistical tests upon it.



The technique used has wide applications. First decide how much memory space you need for the data. This may be much less than you think. If the data can be expressed in integers from 0 to 255, then you need only one byte per piece of information. If you can code the information into a number between 0 and 65535, then you need only two bytes per piece of information.

In our program, the information for 435 individuals could be coded into 30 bytes for each individual, or 13050 bytes in total. However, this is not the place to deal with how the horoscopes were calculated; we are more concerned with the problem of economical storage of the information.

On the Sharp MZ-80K (48K), the top address of memory is 53247. The necessary 13050 bytes of RAM can be saved with the LIMIT statement. Thus LIMIT 40197, spares the 13050 bytes space at the top of memory, (53247 - 13050 = 40197).

You then POKE the data systematically into the top of RAM, according to the needs of your program. Now type 'NEW'. This clears any program, but the data is still protected in the addresses above the LIMIT statement. Remember of course to press the 'CR' key after 'NEW'.

Now do the following POKES. POKE 4356,X:POKE 4357,Y:POKE 4354,A:POKE 4355,B

In this case X and Y are the low byte and the high byte of the start address, while A and B are the low byte and the high byte of the length of the data to be stored, (in bytes). In our example, the start address is 40198, and the length is 13050 bytes. Now 40198 = (157 x 256) + 6. This means that X = 6 and Y = 157, in our example, for the start address. Similarly, 13050= (50 x 256) + 250. Thus A = 250, and B =50.

So in our example, type in:

```
POKE 4356,6:POKE4357,157:POKE4354,250:POKE4355,50
```

For those unfamiliar with the high byte and the low byte idea, the calculation is simple. Simply find the value of INT(Z/256),where Z = the start address or the length. This is the high byte. Then find the value of Z - INT(Z/256); this is the low byte.

Next you need to record the data. Put a clear tape in, and type USR(33:USR(36). Remember to press 'CR'. When asked to 'RECORD.PLAY', do just that. Use the direct mode.

In our example, the whole sequence is as follows:

(1) Make sure your data is poked into the top of memory, and that there is a LIMIT statement.
(2) Type 'NEW'.
(3) Type 'POKE4356,6:POKE4357,157:POKE4354,250:POKE4355,50: USR(33):USR(36)' and 'CR' in direct mode.
(4) Press Record and Play when asked to do so.

You will now have a tape with your data poked into high memory.

In order to retrieve the data, you need to do the following. First of all LOAD BASIC. Now type 'BYE'. This will return you to Monitor Control. Next Load your tape containing your data in the normal way. A star will appear when loading is complete. Now type in GOTO $1260 and 'CR', followed by LIMIT 40197, (or whatever the LIMIT happens to be for your data). Hopefully, you will now see a 'READY' sign.

You may now load any progr4am to analyse your data. The data will be in the top of RAM, and if you have read this far it should be possible to access it.

The following sort of routine might be useful, for example:

```
10 DIM A$(30)
20 FOR J = 1 TO 30
30 A$(J) = STR$(PEEK(40197 + J))
40 NEXT J
```

This subroutine will place 30 elements into the A$ array. You may now if you wish erase those 30 elements from top RAM by resetting the limit.

Finally, if you are trying to protect your programs, beware of POKE 7409,0:POKE7410,0. If the would-be lister pokes these addresses before loading your program, he is likely to be able to list your program. Suggestions round this are welcome!

MZ-80K FULL STRING COMPARISONS for SHARP BASIC SP-5025

by

Alan Stevens

One of the shortcomings of Sharp BASIC SP-5025 is its limited string comparison ability - if can compare strings on the basis of equality only. The ability to test for string INequalities is often useful however, be it for alphabetical sorting of lists of a simple '...IF A$<>"YES" THEN...' test of user input. The following routine is my attempt to overcome this deficiency of SP-5025.

The routine compares strings as far as possible on a letter by letter basis. The letters are ranked as A<B<C<... etc. Lower case letters are taken to be equal to their upper case equivalents i.e. a=A,a<B,A<b,b=B...etc. (This may be changed if required so that all upper case letters are less than all lower case letters i.e. so that A<B<...<Y<Z<a<b<...<y<z. To do this simply POKE 8866,201). If the strings are equal as far as a letter by letter comparison is concerned the longer string is taken to be the greater e.g. SHARPSOFT>SHARP.

For those who have an assembler available I enclose the assembly language mnemonics of the routine (which was written and then printed by the ZEN assembler modified by ZEN MOD v7). For those with no assembler I have provided a BASIC program which READs the machine code (stored in DATA statements) and POKEs it into the appropriate addresses. There are no checksums in the BASIC program so take great care in typing it in.

The following points should be noted. Part of the routine is loaded into RAM from address 3DDC (hex) onwards. This is INSIDE SP-5025. This area is not used by SP-5025 itself; however, if you have a toolkit you might find that this area is occupied by the Toolkit. If this is so there are two options available.

The first is simply to use the LIMIT command to reserve sufficient space (79 bytes for the main routine) at the top of RAM, and change the ORG (assembler) and START (BASIC) parameters accordingly.

The second is to move the start of the BASIC text area (i.e. the place where your programs are stored) up by 79 bytes from 4806 (hex) to 4855 (hex), and then put the main routine here (change ORG to 4806H or START to 18438 decimal). This requires changes to pointers elsewhere within BASIC.

All the pointers which previously pointed to 4806H must now be made to point to 4855H. The location of some of these pointers will depend on just which Toolkit you have installed. It is quite easy to find out where they are RUNning the following BASIC program:

```
010  FOR I=4608 TO 17408
020  IF (PEEK(I)=6)*(PEEK(I+1)=72) THEN PRINT I;
030  NEXT I
```

All the locations so identified must then be POKEd with the number 85 decimal (55hex). If you are using the Knight´s Commander Toolkit the relevant addresses are:

17A9 1839 1AA6 1CF1 1FCF 29C8 2A13 2AAC3 2ADC 2AE8 2B49 2B61 3E6F 3ED1 3FE3 406E (all in hex).

The locations 4616 (1208H) and 4617)1209H) must be POKEd with 85 (55H) and 72 (48H) respectively in order to prevent the main routine being wiped out on cold-starting BASIC.

In all cases address 8805 (2265H) must be POKEd with 120 (78H) in order to make SP-5025´s Boolean expression evaluator skip a few (now unnecessary) instructions.

It must also be noted that the letter by letter comparison subroutine OVERWRITES part of the original SP-5025 Boolean expression evaluator routine. If you have a Toolkit you will need to check that it does not also require this space (Knight´s Commander does not).

You can make a permanent record of your modified BASIC (unless you have adopted the ´LIMIT´ approach) by POKEing the ASCII values of the letters of a suitable name (e.g. STRING BASIC) into locations 10F1 hext onwards (no more than 16 letters and end with 13 (ODH), POKE zero into 1104H and 1106H, and 18 (12H) into 1105H and 1106H. The file length must be put into 1102H and 1103H. This will depend upon where you have put the main routine. For the original version POKE4354,0 POKE4355,68 will do. For the modified start of BASIC text version POKE4354,85 POKE4355,72 is needed. Having done that, insert a blank tape into the cassette deck and type USR(33):USR(36) ´CR´ as a direct command. Press Record and Play as directed to save your modified BASIC.

```
1 REM ***MZ80K Full String Comparisons***
2 REM
3 REM    BASIC host program
4 REM
5 REM    by Alan Stevens
6 REM
7 REM ************************************
8 REM
10 DATA B9280D381579,B7282BCD862228261810
11 DATA B72816CD862228111806,B72805CD86223812
12 DATA 08E604281B1814,08FEB7280FE606280B180E
13 DATA 08FEB72809E60A2805,1119161803111E16
14 DATA CD1A18E1CD7B23C35B22
15 DATA 47CD0118EBCD0118,CD9E224FEBCD9E22
16 DATA EBB9C0231310F1C9,1ACDB90BE67FC9
100 REM
110 START=15836 : H=INT(START/256) : L=START-256*H
```

```
120 REM
130 N=9 : GOSUB 300
140 FOR I=0 TO 78:GOSUB 400:POKESTART+I,D:NEXT
150 REM
160 N=4 : GOSUB 300
170 FOR I=0 TO 30:GOSUB 400:POKE8838+I,D:NEXT
180 REM
190 POKE8805,120:POKE8835,195:POKE8836,L:POKE8837,H
200 END
300 A$="" : B$=""
310 FOR J=1 TO N:READ A$:B$=B$+A$:NEXT
320 RETURN
330 REM
400 HX$=MID$(B$,2*I+1,2) : D=0
410 FOR K=1 TO 2
420 A=ASC(MID$(HX$,K,1))-48:D=D*16+A+(A>9)*7
430 NEXT : RETURN
440 REM
450 REM *********************************
```

```
1                ;*** MZ80K String Inequality ***
2                ;
3                ;     by Alan Stevens
4                ;
5                ;*******************************
6
7                ;     *** Main routine ***
8
9                ;Takes over from SP-5025 at the
10               ;point where the two word lengths
11               ;have been found.
12
13               ZERO:     EQU  1619H
14               MINUS:    EQU  161EH
15               TWRK:     EQU  181AH
16               NMBR:     EQU  237BH
17               EVAL:     EQU  225BH
18
19                         ORG  3DDCH
20               ;Origin assumes no Toolkit present
21
22 3DDC B9       MAIN:     CP   C              ;compare word lengths
23 3DDD 280D               JR   Z,SAME         ;same length
24 3DDF 3815               JR   C,SHORT        ;1st word shorter
25               ;otherwise 1st word longer so
26 3DE1 79                 LD   A,C            ;put shorter length in A
27
28 3DE2 B7       LONG:     OR   A              ;zero length?
29 3DE3 282B               JR   Z,GRTR         ;if yes 1st word > 2nd
30 3DE5 CD8622             CALL COMP           ;else compare letters
31 3DE8 2826               JR   Z,GRTR         ;1st>2nd (it is longer)
32 3DEA 1810               JR   JMPS           ;goto further tests
33
34 3DEC B7       SAME:     OR   A              ;zero length?
35 3DED 2816               JR   Z,EQLTY        ;if yes 1st word=2nd
36 3DEF CD8622             CALL COMP           ;else compare letters
37 3DF2 2811               JR   Z,EQLTY        ;equal in all respects
38 3DF4 1806               JR   JMPS           ;further tests
```

```
39
40 3DF6 B7       SHORT:    OR   A                ;zero length?
41 3DF7 2805               JR   Z,LESS           ;if yes 1st word<2nd
42 3DF9 CD8622             CALL COMP             ;else compare letters
43
44 3DFC 3812     JMPS:     JR   C,GRTR
45               ;a return from the
46               ;comparison subroutine with the
47               ;carry set means 1st word > 2nd
48
49               ;The comparator token is stored
50               ;in the alternate A register,so
51
52 3DFE 08       LESS:     EX   AF,AF            ;get comparator
53 3DFF E604               AND  4                ;mask with 100 binary
54 3E01 281B               JR   Z,TRUE
55 3E03 1814               JR   FALSE
56
57 3E05 08       EQLTY:    EX   AF,AF            ;get comparator
58 3E06 FEB7               CP   0B7H             ;compare with '>' token
59 3E08 280F               JR   Z,FALSE
60 3E0A E606               AND  6                ;mask with 110 binary

61 3E0C 280B               JR   Z,FALSE
62 3E0E 180E               JR   TRUE
63
64 3E10 08       GRTR:     EX   AF,AF            ;get comparator
65 3E11 FEB7               CP   0B7H             ;compare with '>' token
66 3E13 2809               JR   Z,TRUE
67 3E15 E60A               AND  10               ;mask with 1010 binary
68 3E17 2805               JR   Z,TRUE
69
70               ;comparator doesn't match the
71               ;words' relationship so
72 3E19 111916   FALSE:    LD   DE,ZERO          ;address of '0'
73 3E1C 1803               JR   EXIT
74
75               ;comparator does match the
76               ;words' relationship so
77 3E1E 111E16   TRUE:     LD   DE,MINUS         ;address of '-1'
78
79 3E21 CD1A18   EXIT:     CALL TWRK             ;store number
80 3E24 E1                 POP  HL
81 3E25 CD7B23             CALL NMBR             ;pick up flags and
82 3E28 C35B22             JP   EVAL             ;goto Boolean evaluator.
83
84               ;*******************************
85
86               ;***Letter by letter comparison***
87               ;***subroutine. This overwrites***
88               ;***part of original SP-5025   ***
89               ;***Boolean evaluator routine. ***
90
91               FIND:     EQU  1801H
92               CNVT:     EQU  0BB9H
93
94                         ORG  2283H
95
96 2283 C3DC3D             JP   MAIN
97               ;Go from SP-5025 to main
98               ;comparison routine above.
99
```

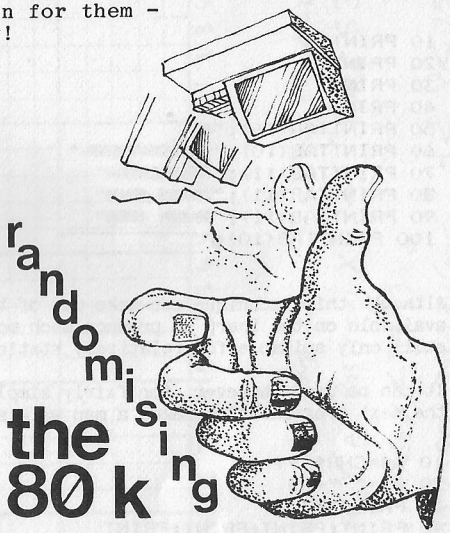
```
100                     ;Subroutine begins here
101
102 2286 47       COMP:     LD   B,A              ;short length for count
103 2287 CD0118             CALL FIND             ;find 2nd word
104 228A EB                 EX   DE,HL            ;interchange pointers
105 228B CD0118             CALL FIND             ;find 1st word
106
107                     ;Begin letter by letter comparison
108                     ;loop
109 228E CD9E22   NEXT:     CALL CODE             ;code 1st letter
110 2291 4F                 LD   C,A              ;save coded value
111 2292 EB                 EX   DE,HL            ;interchange pointers
112 2293 CD9E22             CALL CODE             ;code 2nd letter
113 2296 EB                 EX   DE,HL
114 2297 B9                 CP   C                ;compare letter codes
115 2298 C0                 RET  NZ               ;carry set means 1st>2nd
116 2299 23                 INC  HL               ;next letters
117 229A 13                 INC  DE
118 229B 10F1               DJNZ NEXT             ;loop until count=zero
119
120 229D C9                 RET                   ;return to main routine

121
122                     ;*********************************
123                     ;***ASCII to display code***
124                     ;***subroutine.          ***
125
126 229E 1A       CODE:     LD   A,(DE)           ;Ascii value in A
127 229F CDB90B             CALL CNVT             ;convert to display code
128 22A2 E67F               AND  7FH              ;mask lower case letters
129 22A4 C9                 RET
130
131                     ;*********************************
132
133                       END
```

RANDOMISING THE MZ-80K by J. SIMPSON

While developiong a program which relied heavily on the generation of random numbers I soon realised that things were getting repetitive. I'm no mathematician, but I gather that the MZ-80K produces 'pseudo-random' numbers, each generated from the 'seed' of the previous number. (Perhaps other more knowledgeable readers could give a more detailed explanation?). So it's quite possible to come across the same sequence just too often to make it seem truly random.

I wrote to the Technical Division of Sharp U.K., hoping they could suggest some brilliant formula to meet my need, but perhaps it was too tricky even for them - in any event, I never got a reply!



So for any other readers who may have a similar requirement I humbly offer my home-brewed remedy, until such time as someone produces an infallible solution in machine code! My method makes use of the Sharp's built in clock, which is accessed by TI$, and returns hours, minutes and seconds in the form HHMMSS. The clock is set to zero each time the machine is switched on, and I figures that no one, however quick on the draw, could switch on, load BASIC and load the program in the same amount of time (to the nearest second) every time. So there was my initial random factor - the value of TI$ since switch on.

It was easy enough to run the random number generator for the random number of times indicated by TI$ so that the sequence was always entered at a different point. Unfortunately, there was also the case to consider where the computer had been running for some considerable time before the program was loaded, and if, in the worst possible case, the clock happened to be on 999999, it would become more than a little tedious waiting for the random number generator to perform a million operations.!

It was necessary, therefore, to limit the exuberance of the machine in some way, and I did this by taking an arbitrary figure (5959 in the listing, which would mean the computer had been running for an hour since switch on) and ensuring that if the clock reading came above this, it was cut down by an arbitrary amount (in this case dividing by π)until it came within the specified limit. In fact this limit means that the user never has to wait more than 30 seconds while the routine runs, and it is quite easy to build in a few bells and screen flashes to avoid boredom.

```
10 REM ***RANDOMISER***
20 T = VAL(TI$)
30 IF T > 5959 THEN T = INT(T/π): GOTO 30
40 FOR J = 1 TO T
50 R = RND(1)
60 NEXT J
```

A SIMPLE INTRODUCTION TO GRAPHICS ON THE MZ-80K BY JOHN SIMPSON

There are several ways in which you can draw pictures on the MZ-80K. No doubt you have already discovered the first one, which is simply to use PRINT statements to build up a picture horizontally line by line. For example, the following program draws a crude picture of a house.

```
10 PRINT"■"
20 PRINT
30 PRINT
40 PRINT
50 PRINTTAB(16);"▒"
60 PRINTTAB(10);"◢▒▒▒▒▒▒▒▒▒▒▒▒◣"
70 PRINTTAB(11);"▒▒▒▒▒▒▒▒▒▒▒▒"
80 PRINTTAB(11);"▒H▒▒ ▒H▒"
90 PRINTTAB(11);"▒▒▒▒ ▒▒▒▒"
100 PRINTTAB(10);"▀▀▀▀▀▀▀▀▀▀▀▀"
```

A

Although this technique can make use of the excellent graphic symbols available on the Sharp to produce much more sophisticated results, it is still only suitable for relatively static subjects.

It can be used, however, for fairly simple movement as is demonstrated by the next program which makes a man walk across the screen.

```
10 A$=CHR$(99)
20 A$=" "+A$
30 PRINT"■"
40 PRINT:PRINT:PRINT:PRINT
50 FORI=0TO38
60 PRINT"■";TAB(I);A$
80 FORJ=1TO150
90 NEXTJ
100 NEXT I
```

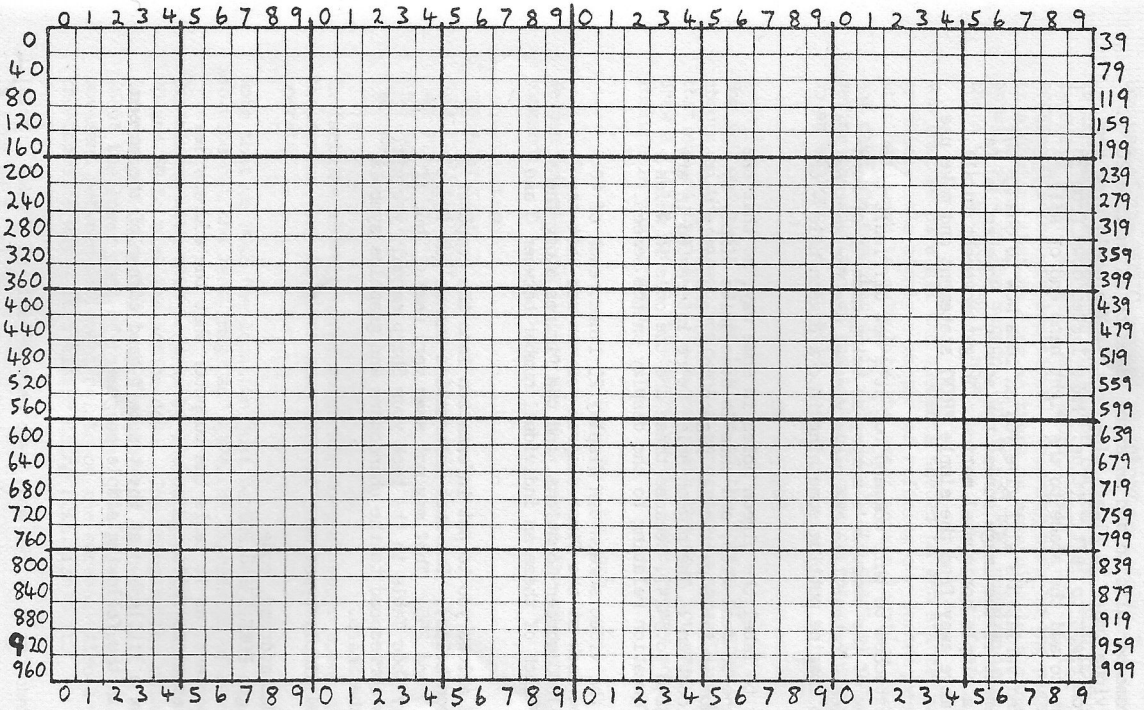
B

Since the program uses a graphic symbol which is not available from the keyboard we have to set it up by using CHR$() to put the man into A$.

We use a FOR:NEXT routine to change the TAB position to print at the different positions across the screen. However, we also need to build in a delay factor between the PRINT operations - otherwise the program would run far too fast - and this is done in lines 80 and 90 with another FOR:NEXT loop.

If we left it at that we would end up with a line of little men across the screen, so line 20 adds a space to A$ which ensures that as we PRINT each man the previous one is wiped out.

You can experiment with this program using different graphic symbols; changing the length of the delay loop by substituting a different value in line 80; or by changing the interval between the TAB positions - e.g.



SCREEN POSITIONS ARE HELD IN RAM FROM 53248 ONWARDS
THUS TO PLACE CHARACTER Y AT POSITION X — "POKE 53248+X, Y"
(X = SCREEN POSITION WHERE $0 \leq X \leq 999$)
(Y = CHARACTER CODE IN MZ-80K DISPLAY CODE $0 \leq Y \leq 255$)

DIAGRAM 1

change line 50 to read:-

```
50 FOR I = 0 TO 38 STEP 2
```

(But if you do this, remember also to add an extra space to A$ in line 20, to ensure that the previous image is always wiped out.) You could also try making the man move from right to left. (Clue: Make I step down from 38 to 0 - and don't forget to add the space to the right hand end of A$!)

Trying out the suggested variations will give you an idea how to change the apparent speed of movement of the image across the screen. However, so far all we've achieved is a simple horizontal motion. To get movement in all directions we need to move away from the simple PRINT statement and make use of the POKE facility.

A lot of mystique is attached to this command but it's not difficult to understand if you think of the computer's memory as being made up of rows and rows of pigeon holes - rather like the racks postmen use for sorting letters. It's easy to imagine putting something into a pigeon hole - or POKEing it in.

It's clearly necessary to label the pigeon holes in some way, so that the computer knows which one to deal with, and this is done quite simply by numbering them. The number for a particular pigeon hole is usually known as the 'address' or 'location'. The pigeon holes we're interested in are those numbered from 53248 to 54247, because these are the ones in which the computer holds information relating to the display on the screen.

You already know that the video screen can display 25 lines, each of 40 characters. If you now look at the first diagram you can see that the squares represent the 1000 screen locations, and the figures along the edges help you to give each of these an individual number between 0 and 999.

We have seen that POKE allows you to put information into the computer's memory locations (pigeon holes). But what information can we put there? Just a number between 0 and 255. That may not seem very useful, but if you look at the Display Code Table in the back your Sharp manual, you will see that these numbers correspond to the characters and graphics symbols which you can use on the Sharp.

To use the POKE command simply type:

```
POKE address, code
e.g.  POKE 53248,202
```

(The comma is very important, as it tells the computer that the address has ended and the code is about to follow.)

This example would draw a little man at the top left hand corner of the screen, and has been included in the following simple program.

```
10 PRINT"E"
20 POKE53248,202
30 MUSIC"R6"
40 POKE54247,202
50 MUSIC"R6"
60 POKE53248,0
70 MUSIC"R6"
80 POKE54247,0
```

(Note that when POKEing to the screen only the Sharp Display Codes can be used and NOT the ASCII codes which we used earlier with CHR$(). Note also the use of the MUSIC command to introduce a delay.)

In the above example we POKEd the bottom right hand corner of the screen (address 54247) as well, and subsequently wiped out the little men by POKEing 0 into both locations. Thinking back to the pigeon holes again, it's obvious that as long as there is something in a screen pigeon hole it will show on the screen, but if you POKE nothing into that pigeon hole there will be nothing to show.

Now that we have all the elements needed to get started with effective graphics, it's simply a matter of refining our techniques. For instance, it's hard work to calculate the exact screen location each time, so why not make the computer do it for us? At the start of our program we'll define the position of the top left hand corner of the screen by setting up an appropriate variable, e.g.:

```
SC = 53248
```

Now you can begin to see the use of the screen diagram. Just decide whereabouts on the screen you want a character to appear, read off the number of that square then POKE SC+ number with the code you want. For example, to draw a little man in the centre of the screen - say square 500 - we just include in our program:

```
POKE SC + 500,202
```

We can now make our little man walk across the screen by using the POKE command.

```
10 PRINT"E"
20 SC=53248
30 FORI=1TO999
40 POKESC+I-1,0
50 POKESC+I,202
60 NEXTI
```

In fact the above program makes him move at super speed across every line of the screen, but it gives some idea of just how fast the POKE command can operate.

Note that we POKE the location just behind our little man with 0, in order to wipe out the last image and give the impression of movement.

Experiment with this program by including a FOR:NEXT loop between lines 40 and 50 to slow the action down. Try changing the values in line 30; for example using 500 to 999 would mean that the little man only moved about the lower part of the screen. Use these values to control his movement between different points on the screen. Also try making him go the opposite way - it's easier using POKE, isn't it?

(Clue: Make I in line 30 step down from 999 to 0 and change the -1 in line 40 to +1.)

We're still only moving horizontally and it's time now to launch out across the screen in all directions. If we start with a character in the centre of the screen (i.e. at location 53248 + 500 = 53748) it's easy to see that

if we move him to the right, we have to increase his screen address by +1. Similarly to move left we change the address by -1. If we want him to move down the screen, we simply add 40 to his screen address. (Because each line on the screen is 40 characters long this puts him in the corresponding position one line lower down.) Conversely to move him up the screen, we subtract 40 from the screen address. So for continuous downward or upward movement we just keep adding or subtracting 40 as required.

We can use this concept in a program to make our little man perform on a trampoline.

```
10 PRINT"E"
20 SC=53248
30 POKESC+538,114:POKESC+539,112:POKESC+540,112
40 POKESC+541,112:POKESC+542,115
50 PM=SC+500
60 POKE PM,202
70 FORJ=1TO75:NEXTJ
80 PM=PM-40
90 POKE PM+40,0
100 IF PM=SC+60 THEN 120
110 GOTO 60
120 POKE PM,202
130 FORJ=1TO75:NEXTJ
140 PM=PM+40
150 POKE PM-40,0
160 IF PM=SC+500 THEN 60
170 GOTO 120
```

E

The POKEs in lines 30 and 40 draw the trampoline in the centre of the screen. In line 50 we set the start position of our man - note that making his position a variable (PM) makes handling the subsequent POKEs much easier. Having put him on the screen in line 60, we have a short delay loop in line 70, then we subtract 40 from his position in line 80 (to make him move upwards) and POKE a blank into his old position (which is PM + 40) in line 90.

Line 100 is very important - it checks to see if he has reached the upper limit of his jump, which we have set at SC + 60. If not, line 110 ensures that the upward movement continues by sending the program back to line 60. If he has reached the top of his jump line 100 sends the program to lines 120 to 150, which reverse the process and start him falling back towards the trampoline. Note that we add 40 to move down the screen.

Line 160 checks whether our performer has reached the trampoline and if so sends the program to line 60 to start him bouncing up again; if not, the program returns to line 120 to continue the falling movement. (The program constantly loops between these routines and so to stop it you have to press SHIFT/BREAK.)

You will be able to work out by looking at the screen chart that diagonal movement is achieved by adding or subtracting 39 or 41, depending which way you want to move. Let's use this in another program to make a flying

saucer move around in space. You'll see from the Display Code table that the appropriate code for the flying saucer symbol is 199.

```
10 PRINT"E"
20 SC=53248
30 PS=SC+500
40 POKE PS,199
50 FORJ=1TO75:NEXTJ
60 OP=PS
70 GOSUB100
80 POKE OP,0
90 GOTO 40
100 R=INT(RND(1)*8)+1
110 IF R=1 THEN X=1
120 IF R=2 THEN X=-1
130 IF R=3 THEN X=40
140 IF R=4 THEN X=-40
150 IF R=5 THEN X=41
160 IF R=6 THEN X=-41
170 IF R=7 THEN X=39
180 IF R=8 THEN X=-39
190 PS=PS+X
200 IF PS<SC THEN PS=PS+40
210 IF PS>SC+999 THEN PS=PS-40
220 RETURN
```

Here again we have used a variable, PS, for the position of the saucer. But we have introduced a further refinement by using another variable OP in which to store the old position of the saucer before we move it on to its new position. The subroutine in lines 100 to 220 performs two major tasks. First, it generates a random number in the range 1-8 (line 100) and uses this to determine in which direction the saucer will move - lines 110 to 180 give an appropriate value to X, which, when added to the present screen address, will give a movement of one position horizontally, diagonally, or vertically. Secondly, in lines 200 and 210 it checks whether the saucer has reached the top or bottom of the screen, and if so, adds or subtracts 40 from the current screen location to pull the saucer back towards the centre of the screen.

Routines based on this program can be written into your own programs to give a randomly moving alien, monster, etc.

So far we have only handled the movement of objects consisting of a single character, but once the principles have been grasped, it is not difficult to extend the same routines to handle larger objects. You can use the screen chart to design your plane, boat, tank, etc., trying out different combinations of graphics symbols until you have decided on the simplest and most effective design. (If you have access to a photocopier it is worth making several copies of this chart for your own use only, but please note that printing copies for other people, whether for sale or not, would constitute breach of copyright.)

Having produced our design and having looked up the display codes for the symbols we need the trick is to select a single square within the object and allocate the screen address for that square only. Then you can define the rest of your object in relation to that point.

DIAGRAM 2

-164	-163	-162	-161	-160	-159	-158	-157	-156
-124	-123	-122	-121	-120	-119	-118	-117	-116
-84	-83	-82	-81	-80	-79	-78	-77	-76
-44	-43	-42	-41	-40	-39	-38	-37	-36
-4	-3	-2	-1	X	+1	+2	+3	+4
+36	+37	+38	+39	+40	+41	+42	+43	+44
+76	+77	+78	+79	+80	+81	+82	+83	+84
+116	+117	+118	+119	+120	+121	+122	+123	+124
+156	+157	+158	+159	+160	+161	+162	+163	+164

POKE POSITIONS ROUND A POINT 'X'

POSITIONS ABOVE THE THICK LINE ARE EXPRESSED BY X−n, BELOW THE LINE BY X+n, where X is the Screen Address in the range 53248 to 54247 and n is the number shown in the relevant position on the chart. [Ensure that X−n > 53248 and X+n < 54247]

The main directions away from X can be expressed by:
NORTH X−40 (for each line moved); SOUTH X+40; WEST X−1; EAST X+1; N.E X−39; N.W. X−41; S.E. X+41; SW X+39.

This is easier to understand if you examine carefully the following program, which makes a tank move across the screen.

```
10 PRINT"E"
20 SC=53248
30 TP=SC+595
40 FORI=1TO25
50 POKE TP,74:POKE TP-40,58
60 POKE TP-2,81:POKE TP-1,74:POKE TP-41,56
70 POKE TP+1,74:POKE TP-39,0:POKE TP+2,74
80 POKE TP+3,87:POKE TP+4,0
90 FORJ=1TO150:NEXTJ
100 TP=TP-1
110 NEXTI
```

G

We define a position TP at which the centre of the tank will be located. In this example it happens to be one of the wheels. In line 50 we first POKE the tank position with the code for the wheel shape; then, as we want the turret to come above the centre of the tank, we POKE the code for the turret shape immediately above at TP-40. And so on for the rest of the tank. If you keep your Display Code table in front of you while examining lines 50 to 80 you will realise how the tank shape is built up. When you come to design complex shapes of your own you will find the second chart, which gives the amounts to add or subtract when defining positions around a point 'X', saves a lot of counting squares.

We can now show a further use of POKE by playing a dirty trick on our unsuspecting tank. Add the following lines to the program:

```
35 MP=SC+568:POKE MP,191
120 POKE TP-2,107:POKE TP-3,104:POKE TP-1,105
130 GOSUB 200
140 END
200 FOR K=1TO150
210 POKE 4514,K
220 USR(68)
230 NEXTK
240 RETURN
```

H

Line 35 plants a mine in the path of the tank. When the tank reaches it line 120 POKEs the explosion effect which blows away the front of the tank, while the Subroutine at lines 200 to 240 provides the sound effect. (The use of POKE in line 210 is rather different from the way we have used it for graphics. Unless you fully understand the right way to POKE other parts of the memory, do not POKE locations outside the Video RAM i.e. limit yourself to POKEing addresses 53248 to 54247.)

You could make the explosion sequence more devastating by POKEing more flying debris on to the screen, and calling the sound subroutine again.

Finally, a simple POKE routine which has a number of uses. The following program POKEs stars at random on to the screen:

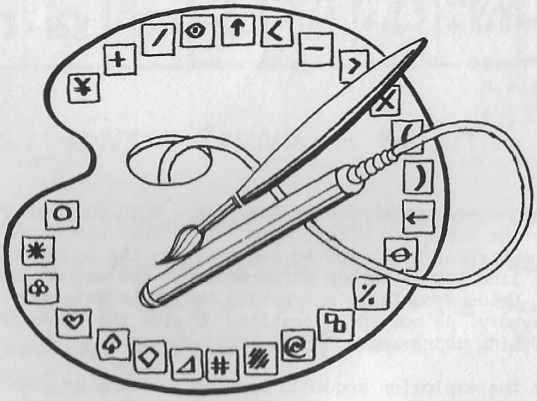
```
10 PRINT"E"
20 SC=53248
30 FORI=1TO75
40 R=INT(RND(1)*999)
50 POKE SC+R,107
60 NEXT I
```

This program can be used for creating the universe, as above, or by reducing the value in line 40 below 999, you could keep the stars to the top part of the screen and just have a starry sky. Change the display code in line 50 to 70, or 80 (either of which gives a fair impression of a tree) and you can put a wood on the screen. Increase the value of I in line 30 and you can have a forest as dense as you wish.

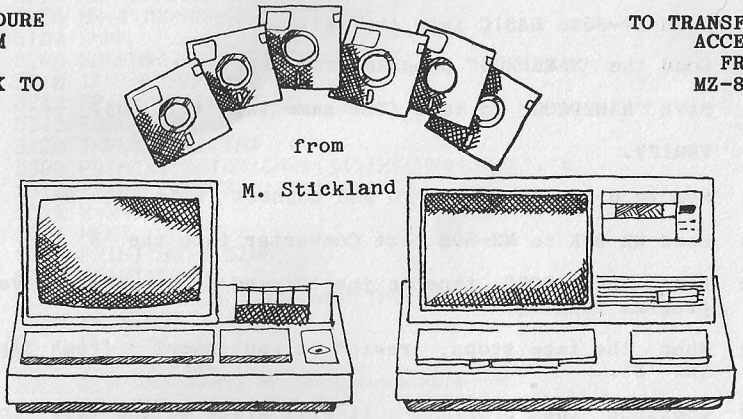
If you use this program to provide background 'scenery' and then put a moving character on to the screen, you will find that as the character passes over the background, the scenery is wiped out. You can get round this by using the PEEK command ..... but (if the Editor asks me!) that's another article!

JOHN SIMPSON

APRIL 1983



PROCEDURE TO TRANSFER RANDOM ACCESS FILES FROM MZ-80K TO MZ-80B



from

M. Stickland

Yes, it seemed such a good idea to upgrade from the MZ-80K to an MZ-80B, because it is so easy to transfer programs using the conversion program. Or is it? Two problems arise:

1) The conversion program only works with SP-5025 casette BASIC, not with SP-6015 or 6115 Disc BASICS.

2) The conversation program will not accept data tapes or transfer Random Access or Sequential Files from Disk.

As the majority of my work has been using Double Precision Disk BASIC and Random Access Files, imagine my dismay when trying to transfer data to the 'B' to discover that there was no way of doing it. Readers with a similar problem will therefore be interested to learn how the problem was resolved. The only way that came to mind was to read the data from the Random Access File, and to convert the data into program lines. Programs containing data that are to be converted to 'B' format MUST be within inverted commas.

The short Program listed here was therefore devised, and the records are read from Disk five at a time, listed on the screen, and the 'CR' has to tapped six times per five records to write new lines into the program. This probably sounds a little weird, but if you run the program all will become clear. This takes about 12 to 15 minutes per block of 500 records, a convenient number to do at one time. Then comes the boring bit.

The following procedure has to be followed:

1) Load Disk BASIC in the 'K' and type in the program. Save it on Disk. Substitute your Random Access File Name in lines 5200 and 6020.

2) Run the program and save the first 500 records as program lines.

3) SAVE/T in Disk BASIC. Use title such as "NAMEPROG 1-500" to indicate which record numbers it conains.

4) VERIFY. Do not leave this stage out, as it will save time in the long run by avoiding checksum errors (NOT possible with SP-6015).

5) Load SP-5025 BASIC into the 'K'.

6) Load the "NAMEPROG" program from tape.

7) SAVE "NAMEPROG" to tape (The same tape will do).

8) VERIFY.

9) Remove disks from MZ-80FD and connect it to the 'B'

10) Load MZ-80K to MZ-80B Text Converter into the 'B'

11) Load "NAMEPROG" tape in the 'B' and start the conversion program running.

12) When the tape stops, rewind it and insert a fresh tape in the 'B'.

13) Continue the program. It will write to the tape in 'B' format.

14) Load Disk BASIC SB-6510 or SB-6610.

15) Load "NAMEPROGR" tape again.

16) Save "NAMEPROG" onto a spare disk temporarily.

17) Insert appropriate Random Access File disk in Drive 2 of Disk Unit.

18) Run program by typing "GOTO 6000".

19) Repeat the above procedure, amending line 5050 so that MN is incremented by 500 each time the program is run.

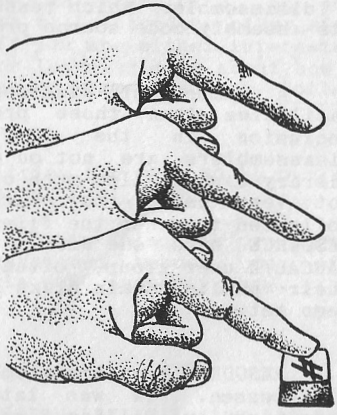
The above might seem a daunting task, but I have found that by doing a section or two only at a time, it is soon achieved. It also helps to have a coffee and a good Computer Magazine to read while all the saving and verifying is going on. If it is any consolation, I have never regretted the work involved, as the 'B' is such an improvement on the 'K'. The extra memory, the graphics and other facilities make it a delight to own.

For the transfer of ordinary programs, things are much easier, except that PEEKs, POKEs and USR numbers have to be altered (See also Issue 6 of User Notes). USR(62) on the 'K' should be replaced by USR(3774) on the 'B' (Bell).

I hope that some of the above will be of assistance. No doubt some Machine Code expert can provide a vastly more sophisticated method - How about it?

One final point. If you have records on Random Access File including ASCII 34 (Inverted Commas) this will cause an error message when finally running the program. If this happens, type PRINT N and this will tell you the line number in which (or adjacent to which) the error has occurred. The " can then be converted to a more innocuous symbol (how about a !) and a note made of the line number. It can then be reconverted in due course.

```
5000 REM PROGRAM TO TRANSFER MZ-80K
5010 REM RANDOM ACCESS FILES TO MZ-80B
5050 MN=1:MX=MN+499
5100 X=MN
5200 XOPEN#1,FD2,"FILENAME RAF"
5210 IFX>MXTHEN5500
5212 PRINT"■":PRINT
5215 FORC=0TO4
5220 INPUT#1(X),IN$
5300 PRINTX;"DATA";CHR$(34);X;CHR$(34);",";
5310 PRINT CHR$(34);IN$;CHR$(34)
5312 X=X+1
5315 NEXT
5330 PRINT"GOTO 5200"
5340 PRINT:PRINT
5350 PRINT"PRESS 'CR' OVER THE ABOVE"
5360 PRINT"■"
5370 CLOSE#1:END
5500 CLOSE#1:END
5999 REM ==============================
6000 REM TFR DATA-DISK RAF
6010 RESTORE
6020 XOPEN#1,FD2,"FILENAME RAF"
6100 READ N,IN$
6110 IFN=0THENCLOSE#1:END
6120 DATA 0,XXXXX
6130 PRINT#1(N),IN$
6140 GOTO6100
```



UTILITIES FOR THE MZ-80B

by

D. Triquell, Spain

KEY REPEAT

This short program allows the automatic repetition of any character.

```
10 FOR I=1 TO 21:READ A:POKE65336+I,A:NEXT I
20 FOR I=1 TO 3:READ A:POKE 2440+I,A;NEXT I
30 END
40 DATA 237,91,38,0,120,254,9,192,237,91,6,0,6,0,197,1,80,0,195,4,5
50 DATA 205,57,255
```

MODIFYING THE CURSOR SPEED

When we are working in CONSOLE C80 display mode, sometimes we should desire a little more speed in the cursor movement.

The speed control of the cursor is located in the address $725: the initial value is $40.

Modiying this value, we modify the cursor speed.

Decreasing this value between $01 and $3F we increase the cursor speed, and increasing this value between $41 and $FF we decrease the cursor speed.

CONTROL

PROGRAM

FOR

MICRO-COMPUTERS

The CP/M column in this issue is shorter than in previous User Notes - mainly because we still have not been able to find the time to download the STAGE2 software from 8" to 5 1/4" disks.

A good disassembler is an important tool if you write complex assembly code programs. The CP/M utility DDT includes a very basic disassembler which is useful for viewing small blocks of code on the V.D.U. Disassembling an entire program can be done using DDT but it is a tedious process. Ideally one requires a disassembler which reads a machine code program then reforms its assembly code source program on disk as an ASCII file.

A number of disassemblers which include more extensive facilities than those provided by DDT have been donated for inclusion in the CP/M library. Unfortunately, these disassemblers are not on one disk but on a number of different library disks. Also much of the other material on these disks is not very useful to Sharp computer owners. Hence we have collected together the files for a well known disassembler called RESOURCE onto one disk. Other user groups, for example the PASCAL/Z user group, often add programs from the CP/M library to their public domain disks so we believe a precedent has already been set.

RESOURCE is a disassembler for 8080 programs written by Ward Christensen. It was later extended for the Z80 with TDL mnemonics by William Earnest and with Zilog mnemonics by Hank Kee. RESOURCE is a professional quality program which will allow virtually any CP/M program to be disassembled. For this reason the documentation contains a warning that the program has been donated to the public domain for personal use and NOT for software PIRATES to "rip-off" commercial software. So if you use RESOURCE please use it in the "spirit" for which it was intended.

With RESOURCE it is possible to generate a fully commented and labelled assembly code source file on disk. It is also easy to use requiring only one or two trial runs before one becomes familiar with its operation.

CP/M library disassembler disk

		NOTES	LIBRARY
6K	RESOURCE.COM	8080 conversational disassembler	(CP/M Vol.42)
34K	REZ.ASM	RESOURCE ASM source code	(Sig/M Vol.10)
28K	REZ.DOC	User instructions	(Sig/M Vol.10)
8K	REZ.COM	RESOURCE-Z80 TDL op-codes	(Sig/M Vol.10)
38K	REZ.Z80	RESOURCE-Z80 Zilog op-codes	(Sig/M Vol.10)

We can supply this disk for the MZ-80K and MZ-80A/B CP/M formats at a cost of £6. Please specify which format you require when you order a copy.

NEW CP/M BOOKS

The RESOURCE disassembler is an example of an add-on transient utility. In the future we will introduce other similar utilities from the CP/M library. If you are either interested in writing your own utility programs or learning more about how CP/M works so that you can write better programs then the following books are a good read:

1. **A Programmer's Notebook:** Utilities for CP/M 80 by David E. Cortesi, Reston Publishing Company Ltd., 1983

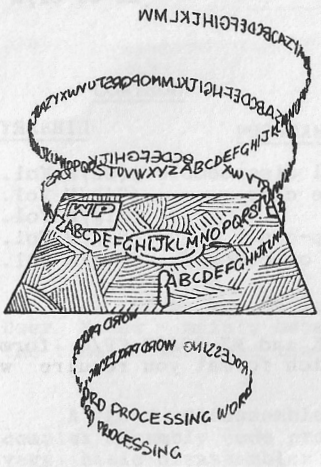
   If you use the CP/M operating system, this book provides you with the concepts and techniques needed to create professional quality programs in 8080 assembly code.

2. **Mastering CP/M** by Alan R. Miller, Sybex, 1983.

   This book explains techniques for using, altering and adding features to the CP/M operating system. It will give you a complete understanding of the CP/M modules, particularly the BIOS and BDOS.

3. **A Programmers' Guide to CP/M,** Edited by Sol Libes, Microsystems Press, 1982.

   Programmers' Guide to CP/M provides the knowledgeable programmer or software designer with a comprehensive in-depth look at CP/M and its capabilities.



FURTHER IMPROVEMENTS TO WP2

by

M.A. HAWES

Further to the listing of WP2 given in Issue 6 of S.U.N. I have made what I think are further improvements to an already useful program, and list them below for your consideration:

LINE 440 should end THEN GOTO500
(GOTO480 gives a NEXT error and in any case is wrong)

LINE 680 add:EF=0:
(This resets the namefile flag so that the file may be loaded again if needed)

LINE 780 insert :CLOSE: before GOTO170
(Without this mod, the namefile is closed only under error, in line 1930)

LINE 830 reverse co-ordinates if using BASIC EXTENSIONS
(The format given is for SPEED BASIC)

LINE 1020 change 'WP1' to 'WP2'.

LINE 1030 delete for clarity of display.

LINE 1110 change 'WP1' to 'WP2'.

LINE 1180 DELETE this line altogether
(It is made redundant by the new line 1190 and also causes filing problems)

LINE 1360 IF X>Y THEN L=L+1:GOTO170
(Without this mod, the programme exits from the reformatting routine with the line counter one too low, and any sebsequent new line overwrites the text)

```
LINE 1372 LL$=RIGHT$(A$(X),1):IF LL$=' 'THEN1376
LINE 1374 A$(X)=A$(X)+' '
LINE 1376 PP$=LEFT$(A$(X),2):IF PP$=' 'THEN1380
LINE 1378 FL$=LEFT$(A$(X),1:IF FL$=' 'THENA$(X)=RIGHT$(A$(X),
          LEN(A$(X))-1
```

(These lines improve formatting; without them, words from the ends of lines may be concatenated without spaces, or an unwanted space may appear at the start of a line)

LINE 1660 remove surplus brackets
(My attention was drawn to this whilst testing the program - I kept getting error conditions in this line, and removed the unwanted brackets for clarity.

```
LINE 1860 EF=0:IFJF=OTHENPRINT'INSERT NAMEFILE TAPE':GOTO1862
LINE 1861 GOTO1870
LINE 1862 PRINT:INPUT'NAMEFILE NUMBER ? ';X$:A$='NAMEFILE.'+X$:
          ROPENA$:JF=1
```

(These lines, with the two below, allow labelling of namefiles. This in turn prevents reading in a wrong file)

```
LINE 1972 INPUT'NAMEFILE NUMBER ? ';X$:A$='NAMEFILE.'+X$
LINE 1974 IFLEN(A$)>15THENPRINT'TOO LONG!!':GOTO1972
```

```
LINE 2320 FORM=XTOY:LL$=RIGHT$(A$(M),1):IFLL$=' 'THEN2322
LINE 2321 A$(M)=A$(M)+' '
LINE 2322 LS=LEN(A$(M)-1:LC=0
```

(These lines add a space at the end of each line if no already there, to cover theuse of FIND & REPLACE before any reformatting has been done)

LINE 2380 Delete altogether, or change the limit to 132 characters or more

LINE 2390 add extra statement :LS=LS+LN-LO
(This mod, was mentioned by P.L. Birch in his article on WP1 in Issue 2, but unfortunately it got misprinted. It prevents truncation on repeated insertions in the same line)

LINE 2400 change middle statement to :LC= LC+LN:
(Also given by Birch. Without this mod, any attempt to insert a new word in front of an existing word will cause the programme to go into a number of loops, adding the new word over and over again until the line is full)

LINES 3150- add comments covering use of bottom right hand key to stop/start listing or printing, and the use of 'R' thereafter to return to command level or any other key to continue.

With the above mods. WP2 is even more flexible than before, and especially so using FIND and REPLACE, which can now be used to insert anything anywhere, as well as to make corrections and deletions.

WP1 HITS DISC

from

M. Bellamy

I have done quite a lot of work on the WP1 Word Processor program, both in respect of its structure, and also in conversion of it to disc operation. Despite the use of BASIC for the input routine, it is fast enough for a two-fingered typist like me, and when compiled can run under FDOS, it would be fast enough for anyone.

```
100 REM.....WORDPRO.....
110 GOTO180
120 DATA33,0,208,17,163,17,213,1,40,0,197,205,166,13,237
130 DATA176,193,65,209,26,205,206,11,205,251,61,19,205,30,0
140 DATA200,16,242,205,18,61,17,232,211,183,237,82,25,56,214,201
150 RO=53200:LIMIT RO:RESTORE 120
160 FOR AD=RO TO RO+45:READ BY:POKE AD,BY:NEXT
170 WOPEN#1,USR(RO):PRINT#1:CLOSE#1:I$="":RETURN
180 CLR:PRINT"■"
190 I1$="                                                    MIKE BELLAMY,"
200 I2$="                                                  28 BEACH ROAD,"
210 I3$="                                                SHOREHAM-BY-SEA,"
220 I4$="                                                    SUSSEX."
230 DIM A$(255),P$(154),NM$(100):L=1:LH=0:NM$="NAMEFILE":TM$="TEMPFILE":CL=76
231 RESTORE 238:FOR P=0 TO 27:READ P$(P):NEXT
232 FOR P=32TO 39:READ P$(P):NEXT
233 FOR P=40TO 47:READ P$(P):NEXT
234 FOR P=99TO107:READ P$(P):NEXT
235 FOR P=129TO154:READ P$(P):NEXT
236 P$(81)="<":P$(82)="[":P$(85)="@":P$(87)=">":P$(97)="!":P$(73)="?"
237 P$(79)=":"
238 DATA" ",A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,£,0,1,2,3,4,5
239 DATA 6,7,8,9,-,=,;,/,.,",",#,$,%,&,',(,),+,*
240 DATA a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z
245 PRINT"TO PRINT INSTRUCTIONS PRESS P":PRINT"                    TO RUN PRESS R"
250 GET I$:IF I$=""THEN 250
255 ON ERROR GOTO 260
260 S=0:PRINT"■" OR × (LOWER RH KEY), FOLLOWED BY"
270 PRINT"A LETTER WILL SELECT THE EDIT COMMANDS. THEY ARE:-"
280 PRINT"L = LIST TEXT        P = PRINT TEXT"
290 PRINT"F = FIND/REP WORD    A = ARRANGE TEXT"
300 PRINT"C = CHANGE LINE      T = TITLE CHANGE"
310 PRINT"I = INSERT LINE      R = REPLACE LINE"
320 PRINT"G = GET TEXT         S = SAVE TEXT"
330 PRINT"K = KILL DISC FILE   D = DELETE LINE(S)"
340 PRINT"O = ORGANISE DISC    N = NAMES ON DISC"
350 IF I$="P"THEN GOSUB 120
360 PRINT"ENTER TEXT (OR USE EDIT COMMAND OPT) NOW"
370 FOR XX=0 TO 79:SET XX,22:NEXT
380 GOSUB 582:IFLEN(A$)<1 THEN A$="    ":A$(L)=A$:L=L+1:GOTO 380
390 IF(LEFT$(A$,1)="×")+(LEFT$(A$,1)="")THEN 420
400 A$(L)=A$:L=L+1:GOTO 380
410 REM.....INSTRUCTION RESPONSE
420 PRINT"■TEXT ";L-1;" LINES LONG. ";CL;" CHARS/LINE"
425 POKE 57347,5:POKE 4464,0
430 C$=MID$(A$,2,1)
440 IF(C$="L")+(C$="l")THEN 600
450 IF(C$="G")+(C$="g")THEN 1150
460 IF(C$="P")+(C$="p")THEN 690
470 IF(C$="S")+(C$="s")THEN 1370
480 IF(C$="R")+(C$="r")THEN 990
490 IF(C$="A")+(C$="a")THEN 1520
500 IF(C$="K")+(C$="k")THEN 2690
510 IF(C$="I")+(C$="i")THEN 1070
520 IF(C$="D")+(C$="d")THEN 1240
530 IF(C$="F")+(C$="f")THEN 2150
540 IF(C$="T")+(C$="t")THEN 2340
550 IF(C$="O")+(C$="o")THEN 2800
560 IF(C$="N")+(C$="n")THEN 2060
570 IF(C$="C")+(C$="c")THEN 2550
572 GOTO 380
```

```
580 REM.....................................................INPUT ROUTINE...
582 A$="":AA$="":D=12+(S-5*INT(S/5))*2:CURSOR 0,D:PRINT"?";SPC(75)
584 C=1:S=S+1:FOR A=1 TO 76
586 POKE 57347,(5-C):POKE 4464,C:GET AA$:IF AA$="" THEN 586
587 IF ASC(AA$)=99 THEN C=0:GOTO 586
588 IF ASC(AA$)=98 THEN C=1:GOTO 586
589 IF AA$=CHR$(102)THEN A=76:GOTO 598
590 IF ASC(AA$)=96 THEN A$=LEFT$(A$,LEN(A$)-1):A=A-2:GOTO 596
592 IF ASC(AA$)=17 THEN A$=A$+SPC(40):A=A+39:GOTO596
594 IF(LEN(A$)>66)*(AA$=" ")THEN A=76:GOTO 598
595 A$=A$+AA$
596 CURSOR 0,D:PRINT L;" ";A$;"?":IF LEN(A$)<>2 THEN 598
597 IF(LEFT$(A$,1)="x")+(LEFT$(A$,1)="")THEN A=76
598 NEXT A:RETURN
599 REM.....................................................................LIST TEXT.
600 INPUT"LIST FROM LINE # (A IF ALL)?";X$:IF X$="A"THEN X=1:Y=L-1:GOTO 640
610 X=VAL(X$):INPUT"TO LINE # (E FOR END)?";Z$:IF Z$="E"THEN Y=L-1:GOTO 640
630 Y=VAL(Z$):IF Y>L-1 THEN Y=L-1
640 S=0:PRINT"■":FOR I=X TO Y:PRINT I;" ";A$(I):GOSUB 650:NEXT I:GOTO 670
650 IF I-12*S=12 THEN S=S+1:INPUT"PRESS .+(CR) TO CONT ";I$:PRINT"■";SPC(39)
660 RETURN
670 PRINT: INPUT "END. PRESS .+(CR) TO CONT ";I$:GOTO 260
680 REM.....................................................................PRINT TEXT..
690 INPUT"PRINT FROM LINE # (A IF ALL)?";X$:IF X$="A"THEN X=1:Y=L-1:GOTO 730
700 X=VAL(X$):INPUT "TO LINE # (E FOR END)? ";Z$:IF Z$="E"THEN Y=L-1:GOTO 730
720 Y=VAL(Z$):IF Y>L-1 THEN Y=L-1
730 FOR AA=1 TO 300:NEXT:PRINT "INCL TITLE?(Y/N) "
740 GET T$:IF T$=""THEN 740
760 FOR AA=1 TO 300:NEXT:PRINT"HARD COPY?"
770 GET I$:IF I$="Y"THEN 840
775 IF I$="N"THEN 790
780 IF I$=""THEN 770
790 PRINT"■"
800 IF T$="N"THEN 820
810 PRINT I1$:PRINT I2$:PRINT I3$:PRINT I4$
820 S=0:FOR I=X TO Y:PRINT A$(I):GOSUB 650:NEXT
830 INPUT"END. PRESS .+(CR) TO CONTINUE. ";I$:GOTO 260
840 FOR AA=1 TO 300:NEXT:PRINT "NO. OF COPIES?"
850 GET NC$:IF NC$=""THEN 850
860 NC=VAL(NC$):PRINT"■■";NC;" COPIES        ":IF NC=0 THEN 260
880 FOR AA=1 TO 300:NEXT:PRINT"PRINT LINE NUMBERS?(Y/N) "
890 GET N$:IF N$=""THEN 890
895 INPUT"IS PRINTER AT STARTING POINT? ";II$
900 FOR P1=1 TO NC:IF T$="N"THEN 930
920 PRINT/P:PRINT/P:PRINT/P I1$:PRINT/P I2$:PRINT/P I3$:PRINT/P I4$:PRINT/P
930 FOR I=X TO Y:IF N$="Y"THEN 950
940 PRINT/P A$(I):NEXT I:GOTO 960
950 PRINT/P I;" ";A$(I):NEXT I
960 PRINT/P"■":NEXT P1:GOTO 260
980 REM.....................................................................REPLACE LINE
990 INPUT "REPLACE LINE #? ";I:IF I>L-1 THEN 260
1000 CURSOR 0,13:PRINT"REPLACE TEXT ——┐";SPC(39);"↓"
1010 IF I>1 THEN CURSOR 0,11:PRINT I-1;" ";A$(I-1)
1020 CURSOR 0,15:PRINT I;" ";A$(I):CURSOR 0,17:PRINT"WITH NEW TEXT:-"
1030 CURSOR 0,20:PRINT I+1;" ";A$(I+1)
1050 S=3:GOSUB 582:A$(I)=A$:GOTO 260
1060 REM.....................................................................INSERT LINE
1070 INPUT"INSERT AFTER LINE #? ";X:IF X>L THEN PRINT"PAST E O F ":GOTO 1070
1080 IF X>0 THEN CURSOR 0,13:PRINT X;" ";A$(X)
1090 CURSOR 0,20:PRINT X+1;" ";A$(X+1)
1100 CURSOR 0,16:PRINT"NEW TEXT ↓"
1110 S=2:GOSUB 582
1120 L=L+1:FOR Y=L TO X+2 STEP -1:A$(Y)=A$(Y-1):NEXT:A$(X+1)=A$:GOTO 260
1140 REM.....................................................................TEXT FROM DISC
```

```
1150 X=0:PRINT"TO GET FILE PUT FILE DISC IN DRIVE 2"
1160 INPUT"FILE NAME? ";X$:A$="WP1"+X$:ROPEN #1,FD2,"NAMEFILE"
1170 INPUT #1,I:FOR X=1 TO I:INPUT#1,NM$(X):NEXT X:CLOSE#1:FOR X=1 TO I
1180 IF NM$(X)=A$ THEN X=I+1
1185 NEXT X:IF X<>0 THEN 1210
1190 PRINT"FILE NOT PRESENT":FOR I=1 TO 1000:NEXT:GOTO 260
1210 ROPEN #1,FD2,A$:INPUT #1,I:FOR X=1 TO I:INPUT#1,A$(X):NEXT:CLOSE #1
1220 L=I+1:GOTO 260
1230 REM.....................................................DELETE LINE(S).
1240 PRINT"DELETE OLD TEXT "
1250 INPUT"FROM LINE #(A FOR ALL)? ";X$:IF X$="A"THEN A=1:I=L:GOTO 1300
1270 A=VAL(X$):INPUT"TO LINE #(E FOR END)? ";Z$:IF Z$="E"THEN I=L:GOTO 1320
1280 I=VAL(Z$):IF I>L-1 THEN I=L-1
1290 IF A=I THEN PRINT A;" ";A$(A)
1300 INPUT"OK TO DELETE? Y/N) ";I$:IF I$="Y"THEN 1330
1310 GOTO 260
1330 PRINT"DELELETING OLD TEXT":FOR X=A TO I:A$(X)="":NEXT X
1340 FOR X=A TO L:IF A$(X)=""THEN FOR Y=X+1 TO L:A$(Y-1)=A$(Y):A$(Y)="":NEXT Y
1350 NEXT X:L=L+A-I:GOTO 260
1360 REM.....................................................TEXT TO DISC
1370 PRINT"TO SAVE FILE PUT FILE DISC IN DRIVE 2"
1380 INPUT"FILE NAME?";X$:A$="WP1"+X$
1390 IF LEN(A$)>15 THEN PRINT"TOO LONG":GOTO 1380
1400 INPUT"FROM LINE #(A FOR ALL)? ";X$:IF X$="A"THEN A=1:I=L:GOTO 1440
1410 A=VAL(X$):INPUT"TO LINE #(E FOR END)? ";Z$:IF Z$="E"THEN I=L-1:GOTO 1440
1420 I=VAL(Z$):IF I>L-1 THEN I=L
1440 WOPEN#1,FD2,A$:PRINT#1,L:FOR X = A TO I:PRINT #1,A$(X):NEXT:CLOSE #1
1460 ROPEN#1,FD2,NM$:INPUT #1,I:FOR X=1 TO I:INPUT#1,NM$(X):NEXT:CLOSE #1
1470 I=I+1:NM$(I)=A$
1490 WOPEN#1,FD2,TM$:PRINT#1,I:FOR X=1 TO I:PRINT#1,NM$(X):NEXT:CLOSE #1
1500 DELETE FD2,NM$:RENAME FD2,TM$,NM$:GOTO 260
1510 REM.....................................................FORMAT
1520 PRINT"FORMAT TEXT"
1530 INPUT"CHARS/LINE (MAX 76)?";CL:IF(CL>76)+(CL<0)THEN 1530
1550 INPUT"FROM LINE #(A FOR ALL)? ";X$:IF X$="A"THEN X=1:Y=L:GOTO 1590
1560 X=VAL(X$):INPUT"TO LINE #(E FOR END)? ";Z$:IF Z$="E"THEN Y=L:GOTO 1590
1580 Y=VAL(Z$):IF Y>L THEN Y=L
1590 PRINT"CHANGING TEXT TO ";CL;" CHARACTERS/LINE"
1600 IF X>Y THEN 260
1610 GOSUB 1970
1620 IF LEN(A$(X))=CL THEN GOSUB 2010:X=X+1:GOTO 1600
1630 IF LEN(A$(X))<CL THEN 1760
1640 FOR I=LEN(A$(X))TO 1 STEP-1
1650 IF MID$(A$(X),I,1)="@"THEN NF=2
1660 IF(MID$(A$(X),I,1)=" ")*(I<=CL)THEN 1690
1670 NEXT I:PRINT"LINE ";X;" TOO LONG - NO SPACES":IF NF=2 THEN NF=1
1680 GOSUB 2010:PRINTA$(X):GOTO 260
1690 A$=RIGHT$(A$(X),LEN(A$(X))-I):A$(X)=LEFT$(A$(X),I)
1700 IF(LEFT$(A$(X+1),2)="  ")+(X=Y)THENGOSUB 1730:GOTO 1720
1710 A$(X+1)=A$+" "+A$(X+1):GOSUB 2010:X=X+1:GOTO 1600
1720 A$(X+1)=A$:GOSUB 2010:X=X+1:GOTO 1600
1730 L=L+1:Y=Y+1
1740 FOR I=L TO X+2 STEP-1:A$(I)=A$(I-1):NEXT I:IF FL=1 THEN FX=1:FL=0
1750 RETURN
1760 IF(LEFT$(A$(X+1),2)="  ")+(X=Y)THEN 1780
1770 GOTO 1790
1780 GOSUB 2010:X=X+1:GOTO 1600
1790 IF LEN(A$(X))+LEN(A$(X+1))=<CL THEN A$(X)=A$(X)+" "+A$(X+1):GOTO 1930
1800 FOR I=1 TO LEN(A$(X+1)):IF MID$(A$(X+1),I,1)="@"THEN 1820
1810 GOTO 1830
1820 A$(X+1)=RIGHT$(A$(X+1),LEN(A$(X+1))-1):FL=2
1830 IF ASC(MID$(A$(X+1),I,1))<>32 THEN 1850
1840 GOTO 1860
1850 NEXT I
1860 IF LEN(A$(X))+I>CL THEN 1880
1870 GOTO 1900
```

```
1880 IF FL=2 THEN FL=1
1890 GOSUB 2010:X=X+1:GOTO 1600
1900 A$(X)=A$(X)+" "+LEFT$(A$(X+1),I)
1910 A$(X+1)=RIGHT$(A$(X+1),(LEN(A$(X+1))-I)):GOSUB 2010:GOTO 1600
1920 NEXT I:GOSUB 2010:X=X+1:GOTO 1600
1930 L=L-1:Y=Y-1:IF FL=1 THEN FL=2
1940 GOSUB 2010
1950 FOR I=X+1 TO L:A$(I)=A$(I+1):NEXT I
1960 GOTO 1600
1970 NF=0:FL=0:FX=0
1980 IF LEFT$(A$(X),1)="¤"THEN A$(X)=RIGHT$(A$(X),LEN(A$(X))-1):NF=1
1990 IF LEFT$(A$(X+1),1)="¤"THEN A$(X+1)=RIGHT$(A$(X+1),LEN(A$(X+1))-1):FL=1
2000 RETURN
2010 IF(NF=1)+(FL=2)THEN A$(X)="¤"+A$(X)
2020 IF(NF=2)+(FL=1)THEN A$(X+1)="¤"+A$(X+1)
2030 IF FX=1 THEN A$(X+2)="¤"+A$(X+2)
2040 RETURN
2050 REM.....................................................LIST FILES ON DISC
2060 PRINT"IS DISC IN DRIVE 2? "
2065 GET I$:IF I$="Y"THEN 2080
2070 GOTO 2065
2080 ROPEN#1,FD2,NM$:INPUT #1,I:FOR X=1 TO I:INPUT#1,NM$(X):NEXT:CLOSE#1
2100 S=0:FOR X=1 TO I:PRINT RIGHT$(NM$(X),LEN(NM$(X))-3)
2110 IF X-12*S=12 THEN S=S+1:INPUT"PRESS C +(CR) TO CONT";I$
2120 NEXT X:INPUT"END. HARD COPY? Y/N ";I$:IF I$="Y"THEN 2130
2125 FOR X=1 TO I:NM$(X)="":NEXT:GOTO 260
2130 FOR X=1 TO I:PRINT/P RIGHT$(NM$(X),LEN(NM$(X))-3):NM$(X)="":NEXT:GOTO 260
2140 REM.....................................................FIND/REPLACE TEXT
2150 PRINT"FIND TEXT TO BE REPLACED"
2160 CURSOR 0,13:S=1:PRINT"OLD TEXT? ":GOSUB 582:O$=A$ :LO=LEN(O$)
2170 CURSOR 0,17:S=3:PRINT"NEW TEXT? ":GOSUB 582:N$=A$:LN=LEN(N$)
2180 INPUT"SEARCH FROM LINE #?(A FOR ALL)";X$:IF X$="A"THEN X=1:Y=L:GOTO 2220
2190 X=VAL(X$):INPUT"TO LINE #(E TO END)? ";Z$:IF Z$="E"THEN Y=L:GOTO 2220
2210 Y=VAL(Z$):IF Y>L-1 THEN Y=L-1
2220 INPUT"REPLACE (Y/N)? ";I$:IF I$="N"THEN 260
2240 FOR M=X TO Y:LS=LEN(A$(M)):LC=0
2250 FOR N=1 TO LO:P=LC+N:IF MID$(O$,N,1)=MID$(A$(M),P,1)THEN 2280
2260 LC=LC+1:IF LC+LO=<LS THEN 2250
2270 NEXT M:GOTO 260
2280 NEXT N
2290 A$=MID$(A$(M),1,LC)+N$+MID$(A$(M),LC+LO+1,LS-LC-LO)
2300 IF LEN(A$)>79 THEN PRINT"LINE ";M;" TOO LONG":GOTO 2270
2310 IF I$="Y"THEN A$(M)=A$
2320 PRINT M;" ";A$(M):LC=LC+1:GOTO 2250
2330 REM.....................................................AMEND ADDRESS/TITLE
2340 PRINT"REPLACE ADDRESS WITH TITLE":PRINT"NEW TEXT"
2350 CURSOR 0,12:PRINT"LINE 1 ":PRINT"    ";I1$
2360 CURSOR 0,13:PRINT"  .":CURSOR 0,13:INPUT I1$:I1$=RIGHT$(I1$,LEN(I1$)-1)
2370 CURSOR 0,15:PRINT"LINE 2 ":PRINT"    ";I2$
2380 CURSOR 0,16:PRINT"  .":CURSOR 0,16:INPUT I2$:I2$=RIGHT$(I2$,LEN(I2$)-1)
2390 CURSOR 0,18:PRINT"LINE 3 ":PRINT"    ";I3$
2400 CURSOR 0,19:PRINT"  .":CURSOR 0,19:INPUT I3$:I3$=RIGHT$(I3$,LEN(I3$)-1)
2410 CURSOR 0,21:PRINT"LINE 4 ":PRINT"    ";I4$
2420 CURSOR 0,22:PRINT"  .":CURSOR 0,22:INPUT I4$:I4$=RIGHT$(I4$,LEN(I4$)-1)
2430 GOTO 260
```

```
2540 REM.....................................................AMEND TEXT OF LINE
2550 INPUT"AMEND TEXT LINE #? ";X:IF X>L THEN 260
2560 CURSOR 0,13:IF X>1 THEN PRINT X-1;" ";A$(X-1)
2570 CURSOR 0,15:PRINT X:CURSOR 2,15:PRINT A$(X)
2600 IF X<L-1 THEN CURSOR 0,17:PRINT X+1;" ";A$(X+1)
2610 CURSOR 0,22:PRINT"ALTER LINE ";X
2620 CURSOR 0,15:INPUT A$:A$=""
2630 FOR I=0 TO 75:A$=A$+P$(PEEK(53850+I)):NEXT
2640 FOR I=0 TO 75:IF RIGHT$(A$,1)=" "THEN A$=LEFT$(A$,LEN(A$)-1):GOTO 2670
2650 I=75
2670 NEXT:A$(X)=A$:A$="":GOTO 260
2680 REM.....................................................FILE DELETION
2690 PRINT"IS FILE DISC IN DRIVE 2?(Y/N) "
2692 GET I$:IF I$="Y"THEN 2700
2694 GOTO 2692
2700 INPUT"NAME OF FILE TO BE DELETED? ";X$:A$="WP1"+X$:X$=""
2720 ROPEN#1,FD2,NM$:INPUT #1,I:FOR X=1 TO I:INPUT#1,NM$(X):NEXT:CLOSE #1
2730 FORX=1TOI:IF NM$(X)=A$ THENFOR Y=X TO I-1:NM$(Y)=NM$(Y+1):NEXTY:NM$(I)=""
2735 NEXT X:GOTO 2750
2740 PRINT"FILE NOT PRESENT":FOR I=1 TO 1000:NEXT:GOTO 260
2750 I=I-1:DELETE,FD2,A$
2770 WOPEN#1,FD2,TM$:PRINT#1,I:FOR X=1 TO I:PRINT#1,NM$(X):NEXT:CLOSE #1
2780 DELETE,FD2,NM$:RENAME,FD2,TM$,NM$
2790 PRINT"FILE ";A$;" DELETED":FOR I=1 TO 1000:NEXT I:GOTO 260
2800 PRINT"FORMAT DISC FOR USE"
2810 PRINT"IS DISC IN DRIVE 2? "
2812 GET I$:IF I$="Y"THEN 2820
2814 GOTO 2812
2820 WOPEN #1,FD2,NM$:PRINT#1,0:CLOSE #1:GOTO 260
5000 FOR I=0 TO 154:PRINTI;"  ";P$(I)
5010 NEXT
```

```
100 REM LUNAR LANDER
110 DIM FC(10),AC(10),PN$(30),PT(30)
120 FOR X=1TO9
130 READ FC(X),AC(X)
140 NEXT X
150 DATA 2,-40,3,-20,5,-10,7,-2,10,0,14,2,18,5,22,10,26,20
180 GOSUB 710
190 GETI$:IFI$=""THEN190
200 IFI$="A"THENAS$="N":GOTO230
210 IFI$="L"THEN230
220 GOTO190
230 NV=0:OV=NV:IS=5:H=50000:F=3000
240 TI$="000000":AC=0:I=5
250 GOSUB900
260 TI$="000000"
270 SC=VAL(RIGHT$(TI$,2)):MN=VAL(MID$(TI$,3,2)):TM=60*MN+SC
280 IFVAL(RIGHT$(TI$,2))=SCTHEN280
290 USR(62)
300 GETI:IFI=0THENI=IS
310 AC=AC+AC(I)
320 OV=NV:NV=NV+AC:OH=H:H=H+NV
330 F=F-FC(I):IS=I
340 IFF<1000THENPOKE53320,12:POKE53321,15:POKE53322,23
350 IFOH=HTHENPOKE53342,0
360 IFOH<HTHENPOKE53342,194
370 IFOH>HTHENPOKE53342,193
380 GOSUB980
390 IFF<=0THEN440
400 IFH<=0THEN440
410 IFH>50000THEN430
420 IFOH>50000THEN430
430 GOTO270
440 IFH<0THENH=0:GOSUB980
450 IFF<0THENF=0:GOSUB980
460 GOSUB 1210
470 IF(F=0)*(H<10)*(NV<-10)THENPOKE4466,12:PRINT"   HEAVY LANDING":GOTO540
480 IF(H=0)*(ABS(NV)<5)THENPOKE4466,12:PRINT"   GOOD LANDING":GOTO540
490 IF(OH<10)*(OH*ABS(NV)<81)THENPOKE4466,12:PRINT"  HEAVY LANDING":GOTO540
500 IF(H=0)*(ABS(NV)<10)THENPRINT"████████████   LANDED":GOTO540
510 PRINT"████████████    CRASHED":MUSIC"1_D1_D1_D1_D"
520 IFF=0THENPRINT"   NO MORE FUEL"
570 PRINT"█████████████████    AGAIN ?"
580 INPUT"███████████";I$
590 IFLEFT$(I$,1)="Y"THEN180
600 IFLEFT$(I$,1)="N"THENPRINT"█":END
610 MUSIC"G":GOTO580
620 GOTO460
700 END
710 REM INSTRUCTIONS
720 PRINT"█"
730 PRINT" THIS IS THE LUNAR LANDER. THE AIM IS"
740 PRINT"TO LAND THE LUNAR MODULE ON THE MOON"
750 PRINT"AT A SPEED OF LESS THAN 10 FEET PER"
760 PRINT"SECOND. YOU WILL BE LAUNCHED AT A"
770 PRINT"HEIGHT OF 50000 FEET WITH A THRUST OF"
780 PRINT"5 WHICH WILL MAINTAIN YOUR HEIGHT"
790 PRINT"UNTIL YOU CHANGE THE THRUST OF YOUR"
800 PRINT"ENGINE. THIS YOU DO BY OPERATING THE"
810 PRINT"NUMERAL KEYS. A THRUST OF 9 GIVES"
820 PRINT"MAXIMUM THRUST BUT USES MOST FUEL,"
830 PRINT"20 LBS PER SECOND. A THRUST OF 1"
840 PRINT"WILL CAUSE THE CRAFT TO FREE FALL."
```

```
850 PRINT"YOU START WITH 3000 LBS OF FUEL."
860 PRINT"YOU SHOULD LAND AS QUICKLY AS "
870 PRINT"POSSIBLE. ENTER L WHEN YOU ARE"
880 PRINT"READY TO LAUNCH."
890 RETURN
900 PRINT"▩    HEIGHT      SPEED        TIME     FUEL"
910 PRINT"▩▩▩▩▩";"E.T.A.";TAB(20);"FUEL TIME"
920 PRINT"▩▩";TAB(20);"▲"
930 PRINTTAB(19);"▲▩▲"
940 PRINTTAB(19);"▩▩▩"
950 PRINTTAB(19);"▲ ▲"
960 PRINT"▩▩▩▩▩▩▩▩▩▩~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
970 POKE53616,107:POKE53596,107:POKE53612,107:POKE53638,107:POKE53658,199
980 PRINT"▩▩▩";TAB(3);H;SPC(5);
990 PRINTTAB(15);ABS(NV);SPC(4);
1000 PRINTTAB(24);MID$(TI$,3,2);".";RIGHT$(TI$,2);
1010 PRINTTAB(31);F;SPC(2)
1020 PRINTTAB(3);NV;SPC(4);
1030 PRINTTAB(15);AC*-1;SPC(3);
1040 PRINTTAB(31);FC(I);SPC(2)
1050 PRINT"▩▩";:IFNV<0THENPRINTTAB(8);INT(H/ABS(NV));SPC(4);
1060 IFNV=>0THENPRINTTAB(8);SPC(6);
1070 PRINTTAB(30);INT(F/FC(I));SPC(2)
1080 FORP=53828TO54148STEP40
1090 POKEP,0:POKE P+1,0:NEXTP
1100 PRINT"▩▩▩▩▩▩▩▩▩"
1110 IFI>0THENPOKE53828,78:POKE53829,77
1120 IFI>1THENPOKE53868,78:POKE53869,77
1130 IFI>2THENPOKE53908,78:POKE53909,77
1140 IFI>3THENPOKE53948,78:POKE53949,77
1150 IFI>4THENPOKE53988,78:POKE53989,77
1160 IFI>5THENPOKE54028,78:POKE54029,77
1170 IFI>6THENPOKE54068,78:POKE54069,77
1180 IFI>7THENPOKE54108,78:POKE54109,77
1190 IFI>8THENPOKE54148,78:POKE54149,77
1200 RETURN
1210 REM LANDING ROUTNE
1220 FORPK=53627TO54110STEP40
1230 POKEPK,0
1240 POKEPK+1,0
1250 POKEPK+2,0
1260 POKEPK+3,0
1270 POKEPK+40,0
1280 POKEPK+41,216
1290 POKEPK+42,215
1300 POKEPK+43,0
1310 POKEPK+80,78
1320 POKEPK+81,67
1330 POKEPK+82,67
1340 POKEPK+83,77
1350 POKEPK+120,208
1360 POKEPK+121,208
1370 POKEPK+122,208
1380 POKEPK+123,208
1390 POKEPK+161,86
1400 POKEPK+162,66
1405 NEXTPK
1410 RETURN
1500 REM PROGRAM BY C S JOHNSON
1510 REM 12.1980
1520 REM COPYRIGHT (C'
```

```
100 REM ♦♦♦ LOGICALES ♦♦♦
110 PRINT"■"
120 LET TITLES = 6
130 DIM LS$(16),DE$(4,4)
140 DIM YE$(4),NO$(4)
150 DIM PR$(250)
160 LETFI=INT(TI*RND(1))+1
170 IFFI=1THEN220
180 LETFI=(FI-1)*24
190 FORIT=1TOFI
200 READTH$
210 NEXTIT
220 READSU$
230 READGR$
240 FORLI=2TO4
250 READYE$(LI)
260 NEXTLI
270 FORLI=2TO4
280 READNO$(LI)
290 NEXTLI
300 FORNA=1TO16
310 READLS$(NA)
320 NEXTNA
330 RESTORE
340 PRINTSU$
350 PRINT"========================================="
360 PRINT "*Below are the groups to be associated"
370 PRINT
380 PRINT"*There are 4 possibilities in each group"
390 PRINT"■";GR$;"■"
400 PRINT"─────────────────────────────────────────"
410 FOR DELAY = 1 TO 4000:NEXT
420 FORIT=1TO16
430 CO=CO+1
440 PRINTLS$(IT);", ";
450 IFCO<4THEN480
460 PRINT"■■"
470 CO=0
480 NEXTIT
490 PRINT"■ To start just clik any key"
500 GET KE$:IFKE$=""THEN500
510 PRINT "■"
520 FORRO=1TO4
530 LETST=1
540 FORCO=1TO4
550 LETPI=INT(4*RND(1))+ST
560 IFLS$(PI)=""THEN550
570 LETDE$(RO,CO)=LS$(PI)
580 LETLS$(PI)=""
590 LET STRT=STRT+4
600 NEXTCO
610 NEXTRO
620 FORCL=1TO250
630 LETX1=INT(4*RND(1)+1)
640 LETY1=INT(4*RND(1)+1)
650 LETX2=INT(4*RND(1)+1)
660 LETY2=INT(4*RND(1)+1)
670 REM──────────────────────
680 IFY2=Y1THEN660
690 IFY1>Y2GOSUB1270
700 IFX1=X2THENLI$=YE$(Y2):GOTO720
710 LETLI$=NO$(Y2)
720 LETTHI$=STR$(X1)+STR$(Y1)+STR$(X2) +STR$(Y2)
730 FORIT=0TOGI
```

```
740 IFTH$=PR$(IT)THEN630
750 NEXTIT
760 LETGI=IT+1
770 LETPR$(IT)=TH$
780 LETSU$=DE$(X1,Y1)
790 LETOB$=DE$(X2,Y2)
800 PRINTSU$;" ";LI$;" ";OB$
810 PRINT TAB(14);"Want another clue ? (Y,N)"
820 GETKE$:IFKE$=""THEN820
830 IFKE$="N"THEN850
840 NEXTCL
850 MUSIC"T A0A0 A0"
860 PRINT"■"
870 LETIT=1
880 FORRO=1TO4
890 PRINTDE$(RO,1);" ";VE$(2)
900 INPUTLS$(IT)
910 PRINTLS$(IT);" ";VE$(3)
920 IT=IT+1
930 INPUTLS$(IT)
940 PRINTLS$(IT);" ";VE$(4)
950 IT=IT+1
960 INPUTLS$(IT)
970 PRINTLS$(IT)
980 IT=IT+1
990 PRINT
1000 PRINT"═══════════════════════════════════════"
1010 NEXTRO
1020 LETIT=1
1030 FORRO=1TO4
1040 FORCO=2TO4
1050 IFDE$(RO,CO)=LS$(IT)THEN1070
1060 LETMI=1
1070 IT=IT+1
1080 NEXTCO,RO
1090 IFMI=0 GOSUB1310
1100 IFMI=1 GOSUB1370
1110 PRINT"■"
1120 FORRO=1TO4
1130 IT=2
1140 FORCO=1TO4
1150 IFIT=5THENIT=0
1160 PRINTDE$(RO,CO);" (";VE$(IT)
1170 IT=IT+1
1180 NEXTCO
1190 PRINT
1200 NEXTRO
1210 FORDL=1TO10000:NEXT
1220 PRINT TAB(8);"Want another game ?(Y,N)"
1230 GETKE$:IFKE$=""THEN1230
1240 IFKE$="N"THEN1470
1250 CLR:RUN
1260 STOP
1270 LETSW=Y1
1280 LETY1=Y2
1290 LETY2=SW
1300 RETURN
1310 PRINT"■"
1320 MUSIC"A1 D2A1R0A1 D2A1"
1330 PRINT
1340 PRINTTAB(7);"Well done sir! (or madam)
1350 PRINTTAB(7);"You've got them all correct*
1360 RETURN
1370 PRINT"■"
1380 FORT=4TO1STEP-1
1390 TEMPOT
```

```
1400 MUSIC"A0"
1410 NEXTT
1420 PRINT
1430 PRINTTAB(17);"WRONG!"
1440 PRINTTAB(6);"Perhaps you'd like something"
1450 PRINTTAB(6);"a little easier next time ?"
1460 RETURN
1470 MUSIC"-A0R0A0R0_A0"
1480 END
1490 DATA Title ♦♦♦WHO DOES WHAT ?♦♦♦
1500 DATA FORENAMES  SURNAMES  JOB CHARACTER
1510 DATA IS,IS THE,IS,IS NOT,IS NOT THE,IS NOT
1520 DATA  JACK,FRED,FRANK,GEORGE
1530 DATA  JONES,HODGES,PIKE,MAINWARING
1540 DATA BITCHER,GROCER,CLERK,BANKER
1550 DATA OLD,SOUR,STUPID,POMPOUS
1560 REM──────────────────────────
1570 DATA Title ♦♦♦ FOOTBALL TEAMS ♦♦♦
1580 DATA TEAM'S NAME  COLOURS  THEIR GROUND  HOME TOWN
1590 DATA WEAR,GROUND IS,TOWN IS
1600 DATA DONT WEAR,DONT PLAY IN,HOME TOWN IS NOT
1610 DATA RANGERS,ROVERS,WANDERERS,UNITED
1620 DATA BLUE,RED,GREEN,YELLOW
1630 DATA STADIUM,WOMBLEY,GASWORKS LANE,BOTTLE PARK
1640 DATA GLASGOW,LONDON,BIRMINGHAM,EDINBURGH
1650 REM──────────────────────────
1660 DATA TITLE ♦♦♦ Supermarket Shelves
1670 DATA PRODUCT  PRODUCT-NAME  TYPE-OF-PACKAGE  DISCOUNT
1680 DATA IS CALLED,COMES IN THE, NOW AT
1690 DATA IS NOT CALLED, DOESN'T COME IN THE,IS NOT SELLING AT
1700 DATA BUTTER,JAM,BISCUITS,SOUP
1710 DATA ACME,TOPPO,ICKY,CREAMO
1720 DATA TINFOIL,BOX,JAR,TIN
1730 DATA 20,60,35,50
1740 REM──────────────────────────┘
1750 DATA TITLE ♦♦♦ Fantasy ♦♦♦
1760 DATA PERSON  COLOUR OF CLOAK  TRAIT  WEAPON
1770 DATA WEARS THE,IS THE,CARRIES THE
1780 DATA DOESN'T WEAR A,IS'NT THE,DOESN'T USE A
1790 DATA WIZARD,ELF,DWARF,MAN
1800 DATA RED CLOAK,YELLOW CLOAK, BLUE CLOAK,GREEN CLOAK
1810 DATA TALLEST,SMALLEST,BRAVEST,STRONGEST
1820 DATA SILVER SWORD,BRONZE AXE,EWE BOW,LONG SPEAR
1830 REM──────────────────────────
1840 DATA TITLE ♦♦♦ Cars ♦♦♦
1850 DATA OWNER  MAKE  REG.NO  COLOUR
1860 DATA IS ,HAS NUMBER,IS
1870 DATA IS NOT,DOES NOT HAVE NUMBER,IS NOT
1880 DATA FREDS CAR,BERTS CAR,SIDS CAR,ALFS CAR
1890 DATA JAGUAR,ROLLS,MINI,BUBBLE CAR
1900 DATA SHA 3P,HMS IIR,UR 21,XL 5
1910 DATA YELLOW,SILVER,RED,GREEN
1920 REM──────────────────────────
1930 DATA TITLE ♦♦♦ Haulage ♦♦♦
1940 DATA COMPANY'S NAME  VEHICLE  TOWN  GOODS
1950 DATA USE,Co.IS IN,MOVES
1960 DATA DONT USE,Co ISN'T IN,DON'T MOVE
1970 DATA WILLIAMS ,SPEEDY,JUGGERNAUT,ROADLINERS
1980 DATA RED LORRIES,BLUE VANS,WHITE LORRIES,GREEN VANS
1990 DATA GLASGOW,BRADFORD,LIVERPOOL,ABERDEEN
2000 DATA FURNITURE,LIVESTOCK,STEEL,PARCELS
2010 REM──────────────────────────
```

Mr. R. Krishnan presents:

A pools plan checking program. This popular version is only useful if you do the pools and check your results against the Daily Express Newspaper Plan No. 37. It would be impossible to publish a program for every pools system, so, some of these ideas and methods could be used to form your own checking program according to whichever plan you use.

This program is NOT a pools Forecaster.

```
5 REM * THIS PROGRAM HAS BEEN WRITTEN USING   *
6 REM * SP-5060 AND SHOULD BE EASILY CONVERTED*
7 REM * TO ANY OTHER BASIC ON THE SHARP MZ80K *
8 REM * OR MZ80A.                             *
9 REM
```

```
10 REM--EXPRESS-PLAN-37 ** BY R.KRISHNAN
20 TI$="000000"
30 DIMR$(80,2)
40 DIMB(29,16),A(104)
50 DIML$(16)
60 DIM R(16),L(3)
70 REM-----------------------------------
80 PRINT"■■■■■ARE YOU TESTING POOLS PLA N GUARANTEES ?"
90 INPUT"■■■■■■■■■■■■■■■■■■■■";K$
100 IF K$="YES" THEN120
110 GOTO160
120 GOSUB1580
130 GOTO280
140 REM-----------------------------------
150 REM--ROUTINE FOR INPUTTING FORECAST RESULTS
160 PRINT"■"
170 PRINT"INPUT YOUR FORECAST RESULTS I N THE SAME ORDER AS IN YOUR COUPON"
180 PRINT:PRINT"1 FOR HOME WIN"
190 PRINT:PRINT"1.5 FOR AWAY WIN"
200 PRINT:PRINT"2 FOR NON-SCORDRAW"
210 PRINT:PRINT"3 FOR SCORE DRAW"
220 PRINT
230 FOR K= 1 TO 16
240 INPUT R(K)
250 NEXT K
260 REM-----------------------------------
270 REM--ROUTINE FOR PRINT-OUT OF FOREC AST RESULTS
280 PRINT"■"
290 FOR K= 1TO16:PRINTR(K);:NEXT K
300 REM-----------------------------------
310 REM--ROUTINE FOR ANALYSING & PRINTI NG-OUT FORECAST RESULTS
320 D=0:D1=0:H=0:A=0
330 FOR K= 1 TO 16
340 IF R(K)=1 THEN H=H+1
350 IF R(K)=1.5 THEN A=A+1
360 IF R(K)=2 THEN D=D+1
370 IF R(K)=3 THEN D1=D1+1
380 NEXT K
390 PRINT :PRINT:PRINT
400 PRINT
```

```
410 PRINT"         YOUR FORECAST CONTAIN S":PRINT :PRINT
420 PRINTTAB(11);"HOME WINS*****";H
430 PRINT:PRINTTAB(11);"AWAY WINS*****" ;A:PRINT
440 PRINTTAB(11);"NON-SCORDRAWS*";D:     PRINT
450 PRINTTAB(11);"SCORE-DRAWS***";D1
460 FOR 0= 1TO 2000:NEXT 0
470 PRINT"■"
480 REM-----------------------------------
490 REM--ROUTINE TO WARN OF THE TIME RE QUIRED TO CHECK AGAINST CHART
500 A0$="D.EXPRESS PLAN 37:14*2+1 BLOCKS"
510 A1$= "MAXIMUM PTS SCORED ON EACH BLOCK"
520 A2$="CHECKING IN PROGRESS"
530 A3$= "THIS TAKES ABOUT 5 MINUTES"
540 A0=INT((40-LEN(A0$))/2)
550 A1=INT((40-LEN(A1$))/2)
560 A2=INT((40-LEN(A2$))/2)
570 A3=INT((40-LEN(A3$))/2)
580 PRINT:PRINT:PRINT
590 PRINTTAB(A0);A0$
600 PRINTTAB(A0);"-----------------------------------------------"
610 PRINT :PRINT
620 PRINTTAB(A1);A1$
630 PRINT
640 PRINTTAB(A2);A2$
650 PRINT
660 PRINTTAB(A3);A3$
670 REM-----------------------------------
680 REM--ROUTINE TO CHECK THE PTS SCORE D ON EACH LINE OF THE CHART
690 C=0 :F=0
700 Z=0:Z1=0:N=0:N1=0:N2=0
710 C=C+1
720 IF C=1 THEN860
730 IF C=2 THEN880
740 IF C=3 THEN900
750 IF C=4 THEN920
760 IF C=6 THEN940
770 IF(C=8)+(C=10)+(C=14)+(C=16)+(C=18) +(C=25) THEN960
780 IF C=11 THEN980
790 IF C=12 THEN1000
800 IF(C=13)+(C=15)+(C=17)+(C=24)THEN1020
810 IF C=19  THEN1040
820 IF(C=21)+(C=23)+(C=27)+(C=29) THEN1060
830 IF C=30 THEN1380
840 N=4:N1=1:N2=8
850 GOTO1070
860 N=4:N1=1:N2=4
870 GOTO1070
880 N=16:N1=5:N2=16
890 GOTO1070
900 N=8:N1=1:N2=12
910 GOTO1070
920 N=4:N1=13:N2=16
930 GOTO1070
940 N=8:N1=9:N2=16
950 GOTO1070
960 N=4:N1=9:N2=16
970 GOTO1070
980 N=6:N1=1:N2=8
990 GOTO1070
1000 N=6:N1=9:N2=16
1010 GOTO1070
1020 N=5:N1=1:N2=8
1030 GOTO1070
1040 N=104:N1=1:N2=16
1050 GOTO1070
```

```
1060 N=5:N1=9:N2=16
1070 FOR Y=1 TO N
1080 S=0
1090 FOR X= N1 TO N2
1100 CURSOR 33,23:PRINTTI$
1110 READL$(X)
1120 IF L$(X) ="0" THEN1140
1130 S=S+R(X)
1140 NEXT Y
1150 IF C=19 THEN1180
1160 B(C,Y)=S
1170 GOTO1190
1180 A(Y)=S
1190 IFZ<S THEN Z=S
1200 NEXT Y
1210 REM-----------------------------------
1220 REM--ROUTINE TO STORE THE MAXIMUM P TS INEACH BLOCK
1230 K=Z
1240 FOR V= 1 TO N
1250 CURSOR 33,23:PRINTTI$
1260 IF C=19 THEN1290
1270 IF B(C,V)=K THEN1310
1280 GOTO1350
1290 IF A(V)=K THEN1310
1300 GOTO1350
1310 F=F+1
1320 R$(F,0)="■■■■■BLOCK-"+STR$(C):R$(F,1)=  " COLUMN-"+STR$(V)
1330 R$(F,2)="GIVES--"+STR$(K)
1340 PRINTR$(F,0);TAB(10);R$(F,1);TAB(2 3);R$(F,2)
1350 NEXT V
1360 GOTO700
1370 REM-----------------------------------
1380 PRINT"■":PRINTTI$
1390 PRINT"■■■■■■■"
1400 PRINT  "       READY TO PRINT OUT RESULTS"
1410 PRINT"■  IF YOU WANT TO PROCEED PRESS 'SPACE' "
1420 GET Q$:IF Q$=CHR$(32) THENUSR(62):GOTO1420
1430 REM-----------------------------------
1440 REM--ROUTINE TO PRINT-OUT THE MAXI MUM PTS INEACH BLOCK
1450 PRINT"■■■"
1460 FOR X=1 TO F
1470 PRINTR$(X,0);TAB(10);R$(X,1);TAB(2 3);R$(X,2)
1480 IFINT(X/9)=X/9  THEN1500
1490 GOTO1530
1500 USR(62):PRINT TAB(5);"■■PRESS SPACE TO CONTINUE"
1510 GETQ$:IF Q$=CHR$(32) THEN1510
1520 PRINT"■■■"
1530 NEXT X
1540 CURSOR 1,1:PRINTTI$
1550 END
1560 REM-----------------------------------
1570 REM--SUBROUTINE TO GENERATE 16 RAN DOM NOS (1<=R<=3)
1580 FOR X= 1TO 3:L(X)=0:NEXT X
1590 FOR X= 1 TO 16
1600 Y=INT(RND(1)*3+1)
1610 R(X)=Y
1620 L(Y)=L(Y)+1
1630 NEXT X
1640 IF(L(3)=8)*(L(2)=0) THEN1660
1650 GOTO1580
1660 RETURN
```

```
1670 REM----------------------------------
1680 DATA X,X,X,O
1690 DATA X,X,O,X
1700 DATA X,O,X,X
1710 DATA O,X,X,X
1720 DATA X,X,X,O,X,X,O,O,O,O,O,O
1730 DATA X,X,X,O,O,O,X,X,O,O,O,O
1740 DATA X,X,O,X,X,O,X,O,O,O,O,O
1750 DATA X,X,O,X,O,X,O,X,O,O,O,O
1760 DATA X,O,X,X,X,O,O,X,O,O,O,O
1770 DATA X,O,X,X,O,X,X,O,O,O,O,O
1780 DATA O,X,X,X,X,O,X,O,O,O,O,O
1790 DATA O,X,X,X,O,X,O,X,O,O,O,O
1800 DATA X,X,O,O,O,O,O,O,X,X,X,O
1810 DATA O,O,X,X,O,O,O,O,X,X,X,O
1820 DATA X,O,X,O,O,O,O,O,X,X,O,X
1830 DATA O,X,O,X,O,O,O,O,X,X,O,X
1840 DATA X,O,O,X,O,O,O,O,X,O,X,X
1850 DATA O,X,X,O,O,O,O,O,X,O,X,X
1860 DATA X,X,O,O,O,O,O,O,O,X,X,X
1870 DATA O,O,X,X,O,O,O,O,O,X,X,X
1880 DATA X,X,O,O,O,O,O,O,X,X,X,O
1890 DATA O,O,X,X,O,O,O,O,X,X,X,O
1900 DATA X,O,X,O,O,O,O,O,X,X,O,X
1910 DATA O,X,O,X,O,O,O,O,X,X,O,X
1920 DATA X,O,O,X,O,O,O,O,X,O,X,X
1930 DATA O,X,X,O,O,O,O,O,X,O,X,X
1940 DATA X,O,X,O,O,O,O,O,O,X,X,X
1950 DATA O,X,O,X,O,O,O,O,O,X,X,X
1960 DATA X,X,X,O
1970 DATA X,X,O,X
1980 DATA X,O,X,X
1990 DATA O,X,X,X
2000 DATA O,O,O,O,X,X,X,O
2010 DATA O,O,O,O,X,X,O,X
2020 DATA O,O,O,O,X,O,X,X
2030 DATA O,O,O,O,O,X,X,X
2040 DATA X,X,X,O,X,X,O,O
2050 DATA X,X,X,O,O,O,X,X
2060 DATA X,X,O,X,X,O,O,X
2070 DATA X,X,O,X,O,X,X,O
2080 DATA X,O,X,X,X,O,X,O
2090 DATA X,O,X,X,O,X,O,X
2100 DATA O,X,X,X,X,O,O,X
2110 DATA O,X,X,X,O,X,X,O
2120 DATA X,X,X,X,X,O,O,O
2130 DATA X,X,X,X,O,X,O,O
2140 DATA X,X,X,X,O,O,X,O
2150 DATA X,X,X,X,O,O,O,X
2160 DATA X,X,X,O,O,O,O,O
2170 DATA X,X,O,X,O,O,O,O
2180 DATA X,O,X,X,O,O,O,O
2190 DATA O,X,X,X,O,O,O,O
2200 DATA X,O,O,O,X,X,X,X
2210 DATA O,X,O,O,X,X,X,X
2220 DATA O,O,X,O,X,X,X,X
2230 DATA O,O,O,X,X,X,X,X
2240 DATA O,O,O,O,X,X,X,O
2250 DATA O,O,O,O,X,X,O,X
2260 DATA O,O,O,O,X,O,X,X
2270 DATA O,O,O,O,O,X,X,X
2280 DATA X,X,X,X,O,O,O,O
2290 DATA X,X,O,O,X,X,O,O
2300 DATA X,X,O,O,O,O,X,X
```

```
2310 DATA 0,0,X,X,X,X,0,0
2320 DATA 0,0,X,X,0,0,X,X
2330 DATA 0,0,0,0,X,X,X,X
2340 DATA X,X,X,X,0,0,0,0
2350 DATA X,X,0,0,X,X,0,0
2360 DATA X,X,0,0,0,0,X,X
2370 DATA 0,0,X,X,X,X,0,0
2380 DATA 0,0,X,X,0,0,X,X
2390 DATA 0,0,0,0,X,X,X,X
2400 DATA X,X,X,X,0,0,0,0
2410 DATA X,X,0,0,X,0,X,0
2420 DATA X,X,0,0,0,X,0,X
2430 DATA 0,0,X,X,X,0,X,0
2440 DATA 0,0,X,X,0,X,0,X
2450 DATA X,0,X,0,X,0,X,0
2460 DATA X,0,X,0,0,X,0,X
2470 DATA 0,X,0,X,X,0,X,0
2480 DATA 0,X,0,X,0,X,0,X
2490 DATA X,X,X,X,0,0,0,0
2500 DATA X,X,0,0,X,0,0,X
2510 DATA X,X,0,0,0,X,X,0
2520 DATA 0,0,X,X,X,0,0,X
2530 DATA 0,0,X,X,0,X,X,0
2540 DATA X,0,0,X,X,0,0,X
2550 DATA X,0,0,X,0,X,X,0
2560 DATA 0,X,X,0,X,0,0,X
2570 DATA 0,X,X,0,0,X,X,0
2580 DATA X,0,X,0,X,X,0,0
2590 DATA X,0,X,0,0,0,X,X
2600 DATA 0,X,0,X,X,X,0,0
2610 DATA 0,X,0,X,0,0,X,X
2620 DATA 0,0,0,0,0,X,X,X
2630 DATA X,0,X,0,X,0,0,X
2640 DATA X,0,X,0,0,X,X,0
2650 DATA 0,X,0,X,X,0,0,X
2660 DATA 0,X,0,X,0,X,X,0
2670 DATA X,X,X,X,X,X,X,X,0,0,0,0,0,0,0,0
2680 DATA X,X,X,X,X,X,0,0,X,X,0,0,0,0,0,0
2690 DATA X,X,X,X,X,X,0,0,0,0,X,X,0,0,0,0
2700 DATA X,X,X,X,X,X,0,0,0,0,0,0,X,X,0,0
2710 DATA X,X,X,X,X,X,0,0,0,0,0,0,0,0,X,X
2720 DATA X,X,X,X,0,0,X,X,X,X,0,0,0,0,0,0
2730 DATA X,X,X,X,0,0,X,X,0,0,X,X,0,0,0,0
2740 DATA X,X,X,X,0,0,X,X,0,0,0,0,X,X,0,0
2750 DATA X,X,X,X,0,0,X,X,0,0,0,0,0,0,X,X
2760 DATA X,X,X,X,X,X,0,X,0,X,0,X,0,0,0,0
2770 DATA X,X,X,X,X,0,X,0,0,X,0,X,0,0,0,0
2780 DATA X,X,X,X,X,0,X,0,0,0,0,0,X,0,X,0
2790 DATA X,X,X,X,X,0,X,0,0,0,0,0,0,X,0,X
2800 DATA X,X,X,X,0,X,0,X,X,X,0,X,0,0,0,0
2810 DATA X,X,X,X,0,X,0,X,0,X,0,X,0,0,0,0
2820 DATA X,X,X,X,0,X,0,X,0,0,0,0,X,0,X,0
2830 DATA X,X,X,X,0,X,0,X,0,0,0,0,0,X,0,X
2840 DATA X,X,X,X,X,0,0,X,X,0,0,X,0,0,0,0
2850 DATA X,X,X,X,X,0,0,X,0,X,X,0,0,0,0,0
2860 DATA X,X,X,X,X,0,0,X,0,0,0,0,X,0,0,X
2870 DATA X,X,X,X,X,0,0,X,0,0,0,0,0,X,X,0
2880 DATA X,X,X,X,0,X,X,0,X,0,0,X,0,0,0,0
2890 DATA X,X,X,X,0,X,X,0,0,0,X,X,0,0,0,0
2900 DATA X,X,X,X,0,X,X,0,0,0,0,0,X,0,0,X
2910 DATA X,X,X,X,0,X,X,0,0,0,0,0,0,X,X,0
2920 DATA X,X,0,0,X,X,X,X,X,X,0,0,0,0,0,0
2930 DATA X,X,0,0,X,X,X,X,0,0,X,X,0,0,0,0
```

```
2940 DATA X,X,0,0,X,X,X,X,0,0,0,0,X,X,0,0
2950 DATA X,X,0,0,X,X,X,X,0,0,0,0,0,0,X,X
2960 DATA 0,0,X,X,X,X,X,X,X,X,0,0,0,0,0,0
2970 DATA 0,0,X,X,X,X,X,X,X,0,0,X,X,0,0,0
2980 DATA 0,0,X,X,X,X,X,X,X,0,0,0,0,X,X,0
2990 DATA 0,0,X,X,X,X,X,X,X,0,0,0,0,0,0,X
3000 DATA X,X,0,0,0,0,0,0,X,X,X,X,X,X,0,0
3010 DATA X,X,0,0,0,0,0,0,X,X,X,X,X,0,0,X
3020 DATA X,X,0,0,0,0,0,0,X,X,0,0,X,X,X,X
3030 DATA X,X,0,0,0,0,0,0,0,0,0,X,X,X,X,X
3040 DATA X,0,X,0,X,X,X,X,X,X,0,X,0,0,0,0
3050 DATA X,0,X,0,X,X,X,X,X,0,X,0,X,0,0,0
3060 DATA X,0,X,0,X,X,X,X,X,0,0,0,0,X,0,0
3070 DATA X,0,X,0,X,X,X,X,X,0,0,0,0,0,X,X
3080 DATA X,0,X,0,0,0,0,0,0,X,0,0,X,X,X,X
3090 DATA X,0,X,0,0,0,0,0,0,X,X,0,X,X,X,X
3100 DATA X,0,X,0,0,0,0,0,0,X,X,X,X,X,0,X
3110 DATA X,0,X,0,0,0,0,0,0,X,X,X,X,0,X,X
3120 DATA X,0,0,X,X,X,X,X,X,X,0,0,X,0,0,0
3130 DATA X,0,0,X,X,X,X,X,X,0,X,X,0,0,0,0
3140 DATA X,0,0,X,X,X,X,X,X,0,0,0,0,X,0,X
3150 DATA X,0,0,X,X,X,X,X,X,0,0,0,0,0,X,0
3160 DATA X,0,0,X,0,0,0,0,0,X,X,X,X,X,0,0
3170 DATA X,0,0,X,0,0,0,0,0,X,X,X,X,0,X,X
3180 DATA X,0,0,X,0,0,0,0,0,X,0,X,0,X,X,X
3190 DATA X,0,0,X,0,0,0,0,0,0,X,0,X,X,X,X
3200 DATA X,0,X,0,X,0,X,0,X,0,X,0,X,0,X,0
3210 DATA X,0,X,0,0,X,0,X,0,X,0,X,0,X,0,X
3220 DATA X,0,X,0,X,0,0,X,0,X,0,X,0,X,0,X
3230 DATA X,0,X,0,X,0,X,0,X,0,X,0,X,0,0,X
3240 DATA X,0,0,X,0,X,0,X,0,X,0,X,0,X,0,X
3250 DATA 0,X,X,0,X,X,X,X,X,0,0,X,0,0,0,0
3260 DATA 0,X,X,0,X,X,X,X,X,0,X,X,0,0,0,0
3270 DATA 0,X,X,0,X,X,X,X,X,0,0,0,0,X,0,0
3280 DATA 0,X,X,0,X,X,X,X,X,0,0,0,0,0,X,X
3290 DATA 0,X,X,0,0,0,0,0,X,X,X,X,X,0,0,X
3300 DATA 0,X,X,0,0,0,0,0,0,X,X,X,X,0,X,X
3310 DATA 0,X,X,0,0,0,0,0,0,X,0,X,0,X,X,X
3320 DATA 0,X,X,0,0,0,0,0,0,0,X,0,X,X,X,X
3330 DATA 0,X,0,X,X,X,X,X,X,0,X,0,0,0,0,0
3340 DATA 0,X,0,X,X,X,X,X,X,0,X,0,X,0,0,0
3350 DATA 0,X,0,X,X,X,X,X,X,0,0,0,0,X,0,X
3360 DATA 0,X,0,X,X,X,X,X,X,0,0,0,0,0,X,X
3370 DATA 0,X,0,X,0,0,0,0,0,X,0,0,X,X,X,X
3380 DATA 0,X,0,X,0,0,0,0,0,0,X,X,0,X,X,X
3390 DATA 0,X,0,X,0,0,0,0,0,X,X,X,X,X,0,X
3400 DATA 0,X,0,X,0,0,0,0,0,X,X,X,X,0,X,X
3410 DATA 0,0,X,X,0,0,0,0,0,X,X,X,X,X,X,0
3420 DATA 0,0,X,X,0,0,0,0,0,X,X,X,X,0,0,X
3430 DATA 0,0,X,X,0,0,0,0,0,0,X,X,0,0,X,X
3440 DATA 0,0,X,X,0,0,0,0,0,0,0,0,X,X,X,X
3450 DATA 0,X,0,X,0,X,0,X,0,X,0,X,0,X,0,X
3460 DATA 0,0,0,0,X,X,0,0,X,X,X,X,X,X,0,0
3470 DATA 0,0,0,0,X,X,0,0,X,X,X,X,0,0,X,X
3480 DATA 0,0,0,0,X,X,0,0,0,X,X,0,0,X,X,X
3490 DATA 0,0,0,0,X,X,0,0,0,0,X,X,X,X,X,X
3500 DATA 0,0,0,0,X,0,X,0,X,X,X,X,X,0,0,X
3510 DATA 0,0,0,0,X,0,X,0,X,X,X,X,0,X,X,0
3520 DATA 0,0,0,0,X,0,X,0,X,0,0,0,X,X,X,X
3530 DATA 0,0,0,0,X,0,X,0,0,X,X,0,X,X,X,X
3540 DATA 0,0,0,0,X,0,0,X,X,X,X,X,X,0,X,0
3550 DATA 0,0,0,0,X,0,0,X,X,X,X,X,0,X,0,X
3560 DATA 0,0,0,0,X,0,0,X,X,0,X,0,X,X,X,X
3570 DATA 0,0,0,0,X,0,0,X,0,X,0,X,X,X,X,X
```

```
3580 DATA 0,0,0,0,0,X,X,0,X,X,X,X,X,0,X,0
3590 DATA 0,0,0,0,0,0,X,X,0,X,X,X,X,0,X,0,X
3600 DATA 0,0,0,0,0,0,X,X,0,X,0,X,0,X,X,X,X
3610 DATA 0,0,0,0,0,0,X,X,0,0,X,0,X,X,X,X,X
3620 DATA 0,0,0,0,0,0,X,0,X,X,X,X,X,X,0,0,X
3630 DATA 0,0,0,0,0,0,X,0,X,X,X,X,X,0,X,X,0
3640 DATA 0,0,0,0,0,0,X,0,X,X,X,0,0,X,X,X,X,X
3650 DATA 0,0,0,0,0,0,X,0,X,0,X,X,0,X,X,X,X
3660 DATA 0,0,0,0,0,0,0,X,X,X,X,X,X,X,X,0,0
3670 DATA 0,0,0,0,0,0,0,X,X,X,X,X,X,0,0,X,X
3680 DATA 0,0,0,0,0,0,0,X,X,X,X,0,0,X,X,X,X
3690 DATA 0,0,0,0,0,0,0,0,X,X,0,0,X,X,X,X,X
3700 DATA 0,0,0,0,0,0,0,0,0,X,X,X,X,X,X,X,X
3710 DATA X,0,X,0,X,0,X,0
3720 DATA X,0,X,0,0,X,0,X
3730 DATA 0,X,0,X,X,0,X,0
3740 DATA 0,X,0,X,0,X,0,X
3750 DATA X,0,0,X,X,X,0,0
3760 DATA X,0,0,X,0,0,X,X
3770 DATA 0,X,X,0,X,X,0,0
3780 DATA 0,X,X,0,0,0,X,X
3790 DATA 0,0,0,0,X,X,X,X
3800 DATA X,0,X,0,X,0,0,X
3810 DATA X,0,X,0,0,X,X,0
3820 DATA 0,X,0,X,X,0,0,X
3830 DATA 0,X,0,X,0,X,X,0
3840 DATA X,X,X,X,0,0,0,0
3850 DATA X,X,0,0,X,0,X,0
3860 DATA X,X,0,0,0,X,0,X
3870 DATA 0,0,X,X,X,0,X,0
3880 DATA 0,0,X,X,0,X,0,X
3890 DATA X,0,0,X,X,X,0,0
3900 DATA X,0,0,X,0,0,X,X
3910 DATA 0,X,X,0,X,X,0,0
3920 DATA 0,X,X,0,0,0,X,X
3930 DATA 0,0,0,0,X,X,X,X
3940 DATA X,0,0,X,X,0,X,0
3950 DATA X,0,0,X,0,X,0,X
3960 DATA 0,X,X,0,X,0,X,0
3970 DATA 0,X,X,0,0,X,0,X
3980 DATA X,0,0,X,X,0,X,0
3990 DATA X,0,0,X,0,X,0,X
4000 DATA 0,X,X,0,X,0,X,0
4010 DATA 0,X,X,0,0,X,0,X
4020 DATA X,X,X,X,0,0,0,0
4030 DATA X,X,0,0,X,0,0,X
4040 DATA X,X,0,0,0,X,X,0
4050 DATA 0,0,X,X,X,0,0,X
4060 DATA 0,0,X,X,0,X,X,0
4070 DATA X,0,0,X,X,0,0,X
4080 DATA X,0,0,X,0,X,X,0
4090 DATA 0,X,X,0,X,0,0,X
4100 DATA 0,X,X,0,0,X,X,0
4110 DATA X,0,X,0,X,X,0,0
4120 DATA X,0,X,0,0,0,X,X
4130 DATA 0,X,0,X,X,X,0,0
4140 DATA 0,X,0,X,0,0,X,X
4150 DATA 0,0,0,0,X,X,X,X
```

For MZ-80A and MZ-80K enthusiasts - both listings of SIMON

```
1 REM         **MZ-80A SIMON**
4 REM
5 REM
10 REM          written by
20 REM
30 REM      Tony Adams 1/JAN/81
35 REM
40 REM
45 REM
50 REM
55 DIMM$(4)
60 PRINT"▣";"↓↓↓";SPACE$(15);"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒":PRINTSPACE$(15);"▒";SPACE$(9);"▒"
80 PRINTSPACE$(15);"▒  SIMON  ▒":PRINTTAB(15);"▒         ▒":N$="'SIMON'"
90 PRINTSPACE$(15);"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
100 M1$="C2DE#FG7":M2$="‾C4DECE7":M3$="_C5":M4$="_E5":M5$="_G5":M6$="C5"
110 TEMPO7:MUSICM1$;M2$:M$(1)="F2":M$(2)="E2":M$(3)="D2":M$(4)="C2":HS=10
120 M7$="‾C":M8$="‾C8":PRINT"↓↓↓↓↓↓↓↓→→→→Do you want instructions (Y or N) ?"
130 GETA$:IFA$="Y"GOSUB2000
140 D=3:IFA$=""THEN130:REMD=No of goes
160 DIMRN(16),IN(16):REM      plus one
165 PRINT"▣↓↓↓↓↓↓↓↓G E T  R E A D Y. . . .":FORDE=1TO1000:NEXT
170 FORA=0TOD:RN(A)=INT(4*RND(1)+1):NEXT:GOSUB1000:G=0:A=2:FORDE=1TO800:NEXT
180 ONRN(G)GOSUB700,720,740,760
185 FORDE=1TO60:NEXT
189 IF(EF=1)*(G=A-2)THENEF=0:GOTO800
190 G=G+1:IFG=ATHENG=0:A=A+1:GOTO200
195 GOTO180
200 PRINT"↓↓↓↓↓↓↓↓↓↓↓→→→→→→→→→→→→→→→→→→→→NOW↓↓←←←COPY↓↓←←←←THAT":FORDE=1TO500:NEXT
210 GOSUB1000:I=0
220 GETI$:IF(I$="R")+(I$="Y")+(I$="G")+(I$="B")THEN230
223 T=T+1:IFT=100THENPRINT"▣↓↓↓↓↓↓YOU TOOK TOO LONG!!.":FORDE=1TO500:NEXT
224 IFT=100THENT=0:GOTO260
225 GOTO220
230 T=0:IFI$="G"THENIN(I)=1
235 IFI$="R"THENIN(I)=2
240 IFI$="Y"THENIN(I)=3
245 IFI$="B"THENIN(I)=4
250 IFIN(I)=RN(I)THEN280
260 MUSICM7$;M7$;M7$;M7$;M8$:PRINT"▣↓↓↓→→THE CORRECT SEQUENCE FOLLOWS...."
270 FORDE=1TO800:NEXT:GOSUB1000:EF=1:GOTO180
280 ONIN(I)GOSUB700,720,740,760
290 IFSC=433THENPRINT"▣YOU HAVE THE MAX.POSSIBLE SCORE!!!":FORDE=1TO500:NEXT
295 IFSC=433THEN800
300 SC=SC+1:IF(I=D)GOSUB500:I=0:GOTO170
320 IFI=A-2THENI=0:FORDE=1TO500:NEXT:GOTO180
340 I=I+1:GOTO220
500 PRINT"▣↓↓↓↓↓↓↓↓↓↓↓→→→→→→→→→→→→→→→→O.K NOW ↓"
510 PRINT"→→→→→→→→→→→→→→→→TRY THIS":D=D+2
520 FORI=1TO7:FORJ=255TO1STEP-20:POKE4514,I:POKE4513,J:USR(68):NEXTJ,I:USR(71)
530 FORDE=1TO600:NEXT:RETURN
699 END
```

```
700 P=53298:POKEP,119:POKEP+14,118:POKEP+118,52:POKEP+7,121:POKEP+136,52
710 MUSICM3$:POKEP,0:POKEP+14,0:POKEP+118,0:POKEP+7,0:POKEP+136,0:RETURN
720 POKE53495,119:POKE53417,121:POKE53694,52:POKE53935,118:POKE54017,121
730 MUSICM4$:POKE53495,0:POKE53417,0:POKE53694,0:POKE53935,0:POKE54017,0
735 RETURN
740 POKE53433,121:POKE53515,118:POKE53716,52:POKE53955,119:POKE54033,121
750 MUSICM5$:POKE53433,0:POKE53515,0:POKE53716,0:POKE53955,0:POKE54033,0
755 RETURN
760 POKE54099,118:POKE54017,52:POKE54145,121:POKE54033,52:POKE54111,119
770 MUSICM6$:POKE54099,0:POKE54017,0:POKE54145,0:POKE54033,0:POKE54111,0
775 RETURN
800 SP=53248:FORDE=1TO600:NEXT:PRINT"■":IFSC<=HSTHENEF=1:GOTO835
810 PRINT"■■■■■High Score, Enter Your Name■■":PRINTTAB(12);"(MAX 6 LETTERS)■■
820 PRINT"Then Press <CR>";:INPUTT$
830 HS=SC:IFLEN(T$)>6THEN810
832 IFHS>99THENSP=SP+1
833 IFHS>9THENSP=SP+1
835 PRINT"■":IFEFTHENPRINT"YOU SCORED ";SC;"■"
840 PRINT"■■■■■■■■HIGHEST SCORE:- ";HS;" by ";N$:N$=T$
841 IFEF=1THENEF=0:GOTO860
842 FORL=1TOLEN(T$):C=ASC(MID$(T$,L,1))-64:IFC<1THENC=0
844 FORJ=1TO21+L:POKESP+J,C:POKESP+J-1,0
845 POKESP+J+1,202:FORDE=1TO60:NEXTDE,J:FORI=1TO4:POKESP+J-1+(I*80),C
850 POKESP+J-1+(I*80)-80,0:MUSICM$(I):POKESP+J,0:NEXTI,L
855 FORI=1TO6:POKESP+341+LEN(T$)+I,0:NEXT
860 D=3:SC=0:PRINT"■■■■■■■■■■■■■■Another Game ?"
870 GETA$:IFA$="Y"THEN165
880 IFA$="N"THENPRINT"■■■■■C H I C K E N !!":END
890 GOTO870
1000 PRINT"■";SPACE$(12);"** SIMON **";:PRINT
1004 PRINT:PRINT"                  ┌──────────┐"
1005 PRINT"              |   GREEN  |"
1010 PRINT"              └──────────■"
1020 PRINT"          ▄▄▄                 ||||\"
1030 PRINT"          ▒▒▒                 ||||||"
1035 PRINT"          ▒▒▒                 ||Y||"
1040 PRINT"          ▒R▒                 ||E||"
1045 PRINT"          ▒E▒                 ||L||"
1050 PRINT"          ▒D▒                 ||L||"
1055 PRINT"          ▒▒▒                 ||O||"
1060 PRINT"          ▒▒▒                 ||W||"
1065 PRINT"          ▒▒▒                 ||||||"
1070 PRINT"          ▀▀▀                 ||||/■"
1080 PRINT"                ▄▄▄▄▄▄▄▄▄▄▄"
1085 PRINT"                ■■BLUE■■■■■"
1090 PRINT"                ▀▀▀▀▀▀▀▀▀▀▀■"
1095 RETURN
2000 PRINT"■";SPACE$(15);"═══════■"
2010 PRINTSPACE$(16);"S I M O N■"
2020 PRINTSPACE$(15);"═══════■"
2030 PRINT"  'SIMON' is a game which tests your■"
2040 PRINT"ability to memorise a sequence of■"
2045 PRINT"flashing lights & sounds.■"
2050 PRINT"■It will start off with a small■"
2052 PRINT"sequence. If you copy correctly,then■":PRINT"   it gets harder.■"
2055 PRINT"The further you go, the higher your■"
2057 PRINT"score.";SPACE$(16);"■■W A I T..."
2070 FORDE=1TO12000:NEXT
```

```
2090 PRINT"■";TAB(12);"KEYBOARD LAYOUT■■■■■■"
2095 PRINT"                ┌─┐ ┌─┐ ┌─┐
2100 PRINT"       red      |R| |  | |Y| yellow
2110 PRINT"                └─┘ └─┘ └─■"
2130 PRINT"                  ┌─┐
2140 PRINT"                  |G| green
2150 PRINT"                  └─■"
2170 PRINT"                  ┌─┐
2180 PRINT"                  |B| blue
2190 PRINT"                  └─■"
2210 PRINT"              ┌─────┐
2220 PRINT"              |SPACE|
2230 PRINT"              └─────┘
2240 FORDE=1TO3500:NEXT:RETURN
```

```
1 REM        **MZ-80K SIMON**
4 REM
5 REM
10 REM          written by
20 REM
30 REM       Tony Adams 1/JAN/81
35 REM
40 REM
45 REM
50 REM
55 DIMM$(4)
60 PRINT"■";"■■■";SPC(15);"▒▒▒▒▒▒▒▒▒▒▒▒":PRINTSPC(15);"▒";SPC(9);"▒"
80 PRINTSPC(15);"▒";"  SIMON  ";"▒":PRINTSPC(15);"▒";SPC(9);"▒":N$="SIMON"
90 PRINTSPC(15);"▒▒▒▒▒▒▒▒▒▒▒▒"
100 M1$="C2DE#FG7":M2$="■C4DECE7":M3$="■C5":M4$="■E5":M5$="■G5":M6$="C5"
110 TEMPO7:MUSICM1$:M2$:M$(1)="F2":M$(2)="E2":M$(3)="D2":M$(4)="C2":HS=10
120 M7$="■C":M8$="■C8":PRINT"■■■■■■■■■■■■■Do you want instructions (Y or N) ?"
130 GETA$:IFA$="Y"GOSUB2000
140 D=3:IFA$=""THEN130:REMD=No of goes
160 DIMRN(16),IN(16):REM       plus one
165 PRINT"■■■■■■■■■■■G E T  R E A D Y. . . .":FORDE=1TO1000:NEXT
170 FORA=0TOD:RN(A)=INT(4*RND(1)+1):NEXT:GOSUB1000:G=0:A=2:FORDE=1TO800:NEXT
180 ONRN(G)GOSUB700,720,740,760
185 FORDE=1TO60:NEXT
189 IF(EF=1)*(G=A-2)THENEF=0:GOTO800
190 G=G+1:IFG=ATHENG=0:A=A+1:GOTO200
195 GOTO180
200 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■NOW■■■■■COPY■■■■■■THAT":FORDE=1TO500:NEXT
210 GOSUB1000:I=0
220 GETI$:IF(I$="R")+(I$="Y")+(I$="G")+(I$="B")THEN230
223 T=T+1:IFT=100THENPRINT"■■■■■■■YOU TOOK TOO LONG!!.":FORDE=1TO500:NEXT
224 IFT=100THENT=0:GOTO260
225 GOTO220
230 T=0:IFI$="G"THENIN(I)=1
235 IFI$="R"THENIN(I)=2
240 IFI$="Y"THENIN(I)=3
245 IFI$="B"THENIN(I)=4
250 IFIN(I)=RN(I)THEN280
260 MUSICM7$:M7$:M7$:M7$:M8$:PRINT"■■■■■■THE CORRECT SEQUENCE FOLLOWS....."
270 FORDE=1TO800:NEXT:GOSUB1000:EF=1:GOTO180
280 ONIN(I)GOSUB700,720,740,760
290 IFSC=433THENPRINT"■YOU HAVE THE MAX.POSSIBLE SCORE!!!":FORDE=1TO500:NEXT
295 IFSC=433THEN800
```

```
300 SC=SC+1:IF(I=D)GOSUB500:I=0:GOTO170
320 IFI=A-2THENI=0:FORDE=1TO500:NEXT:GOTO180
340 I=I+1:GOTO220
500 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒O.K NOW ▒"
510 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒TRY THIS":D=D+2
520 FORI=1TO7:FORJ=255TO1STEP-20:POKE4514,I:POKE4513,J:USR(68):NEXTJ,I:USR(71)
530 FORDE=1TO600:NEXT:RETURN
699 END
700 P=53298:POKEP,119:POKEP+14,118:POKEP+118,52:POKEP+7,121:POKEP+136,52
710 MUSICM3$:POKEP,0:POKEP+14,0:POKEP+118,0:POKEP+7,0:POKEP+136,0:RETURN
720 POKE53495,119:POKE53417,121:POKE53694,52:POKE53935,118:POKE54017,121
730 MUSICM4$:POKE53495,0:POKE53417,0:POKE53694,0:POKE53935,0:POKE54017,0
735 RETURN
740 POKE53433,121:POKE53515,118:POKE53716,52:POKE53955,119:POKE54033,121
750 MUSICM5$:POKE53433,0:POKE53515,0:POKE53716,0:POKE53955,0:POKE54033,0
755 RETURN
760 POKE54099,118:POKE54017,52:POKE54145,121:POKE54033,52:POKE54111,119
770 MUSICM6$:POKE54099,0:POKE54017,0:POKE54145,0:POKE54033,0:POKE54111,0
775 RETURN
800 SP=53248:FORDE=1TO600:NEXT:PRINT"▒":IFSC<=HSTHENEF=1:GOTO835
810 PRINT"▒▒▒▒▒▒High Score, Enter Your Name▒▒":PRINTTAB(12);"(MAX 6 LETTERS)▒▒
820 PRINT"Then Press <CR>";:INPUTT$
830 HS=SC:IFLEN(T$)>6THEN810
832 IFHS>99THENSP=SP+1
833 IFHS>9THENSP=SP+1
835 PRINT"▒":IFEFTHENPRINT"YOU SCORED ";SC;"▒"
840 PRINT"▒▒▒▒▒▒▒HIGHEST SCORE:- ";HS;" by ";N$:N$=T$
841 IFEF=1THENEF=0:GOTO860
842 FORL=1TOLEN(T$):C=ASC(MID$(T$,L,1))-64:IFC<1THENC=0
844 FORJ=1TO21+L:POKESP+J,C:POKESP+J-1,0
845 POKESP+J+1,202:FORDE=1TO60:NEXTDE,J:FORI=1TO4:POKESP+J-1+(I*80),C
850 POKESP+J-1+(I*80)-80,0:MUSICM$(I):POKESP+J,0:NEXTI,L
855 FORI=1TO6:POKESP+341+LEN(T$)+I,0:NEXT
860 D=3:SC=0:PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒Another Game ?"
870 GETA$:IFA$="Y"THEN165
880 IFA$="N"THENPRINT"▒▒▒▒▒C H I C K E N !!":END
890 GOTO870
1000 PRINT"▒";SPC(12);CHR$(99);CHR$(99);" SIMON ";CHR$(99);CHR$(99):PRINT
1004 PRINT:PRINT"                 /‾‾‾‾‾‾‾‾‾‾‾‾‾‾\
1005 PRINT"                 |  GREEN   |
1010 PRINT"                 |__________|▒"
1020 PRINT"           ▒▒▒▒              ||||\
1030 PRINT"           ▒▒▒▒              |||||
1035 PRINT"           ▒▒▒▒              ||Y||
1040 PRINT"           ▒R▒               ||E||
1045 PRINT"           ▒E▒               ||L||
1050 PRINT"           ▒D▒               ||L||
1055 PRINT"           ▒▒▒▒              ||O||
1060 PRINT"           ▒▒▒▒              ||W||
1065 PRINT"           ▒▒▒▒              |||||
1070 PRINT"           ▒▒▒▒              ||||/▒"
1080 PRINT"                 ▒▒▒▒▒▒▒▒▒▒▒▒▒
1085 PRINT"                 ▒▒▒BLUE▒▒▒▒▒
1090 PRINT"                 ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
1095 RETURN
2000 PRINT"▒";SPC(15);"═══════════▒"
2010 PRINTSPC(16);"S I M O N▒"
2020 PRINTSPC(15);"═══════════▒"
```

```
2030 PRINT"  'SIMON' is a game which tests your■"
2040 PRINT"ability to memorise a sequence of■"
2045 PRINT"flashing lights & sounds.■"
2050 PRINT"■It will start off with a small■"
2052 PRINT"sequence. If you copy correctly,then■":PRINT"    it gets harder.■"
2055 PRINT"The further you go, the higher your■"
2057 PRINT"score.";SPC(16);"■■W A I T..."
2070 FORDE=1TO12000:NEXT
2090 PRINT"■";TAB(12);"KEYBOARD LAYOUT■■■■■■"
2095 PRINT"          ┌─┐ ┌─┐ ┌─┐
2100 PRINT"      red │R│ │ │ │Y│ yellow
2110 PRINT"          └─┘ └─┘ └─┘■"
2130 PRINT"              ┌─┐
2140 PRINT"              │G│ green
2150 PRINT"              └─┘■"
2170 PRINT"              ┌─┐
2180 PRINT"              │B│ blue
2190 PRINT"              └─┘■"
2210 PRINT"          ┌─────┐
2220 PRINT"          │SPACE│
2230 PRINT"          └─────┘
2240 FORDE=1TO3500:NEXT:RETURN
```

Many thanks to David Roberts of the MZ-80B-HK User Group in Hong Kong.

Print a Calendar for the year of your choice. Within the chain command, it is possible to add a picture before printing the calendar and the result, with form feed paper, is a two sheet picture and calendar combination.

```
1 REM Calendar by MZ-80B-HK User Group
2 REM GPO Box 7849 Hong Kong
10 CLR
11 M1$="+F2R+C+C+#C+C2R+ER"
12 M2$="+F"
13 M3$="2BR5"
14 M4$="D3+C5R0A5"
15 ME$="NIL"
16 YR$="1997"
17 YH$="PRESENT"
18 CONSOLE40:TEMPO6:GRAPHC,01,11
20 REM OPTION ENTER GOTO 800 HERE AND MISS LINES 21-90
21 L1$="   ### ### #   ### #  # ##  ### ###"
22 L2$="   #   # # #   #   ## # # # # # #"
23 L3$="   #   ### #   ### # ## # # ### ##"
24 L4$="   #   # # #   #   #  # # # # # #"
25 L5$="   ### # # ### ### #  # ##  # # #"
26 H1=25
30 PRINTCHR$(6)
31 H=H1-1
32 CURSOR 0,H
40 PRINTL1$
41 H1=H
42 H=H+1:IF H>24 THEN 65
45 PRINTL2$
46 H=H+1:IF H>24 THEN 65
50 PRINTL3$
51 H=H+1:IF H>24 THEN 65
55 PRINTL4$
56 H=H+1:IF H>24 THEN 65
60 PRINTL5$
61 H=H+1:IF H>24 THEN 65
65 FOR WA=1TO500:NEXT
66 IF H>14 THEN 30
67 LINE 20,60,300,60,300,120,20,120,20,60
68 FOR WA=1TO2000:NEXT
70 MUSIC M1$
71 PRINTCHR$(6)
72 CURSOR7,10:PRINT"You will require a printer":PRINT
73 PRINTTAB(8);"to produce this calendar":PRINT
75 MUSIC M2$
76 FOR WA=1TO3000:NEXTWA
80 GRAPH C
90 GOTO 800
95 PRINTCHR$(6)
99 CURSOR11,10:PRINT"P R I N T I N G"
100 REM CHARACTER PATTERNS FOR ENLARGED YEAR HEADING
120 DIM YEAR$(9,6)
130 FOR J=0TO6
140 FOR I=0TO9
150 READ YEAR$(I,J)
160 NEXT I,J
170 DATA " #### ","  #  ","### ","### ","#  # ","#### ","#### ","#### ","#### ","#### "
171 DATA "#    #","  #  ","  # ","  # ","#  # ","#    ","#    ","   # ","#  # ","#  # "
172 DATA "#    #","  #  ","  # ","  # ","#  # ","#    ","#    ","   # ","#  # ","#  # "
173 DATA "#    #","  #  ","  # ","  # ","#  # ","#### ","#### ","   # ","#  # ","#  # "
174 DATA "#    #","  #  ","  # ","  # ","#  # ","#    ","#    ","   # ","#  # ","#  # "
175 DATA "#    #","  #  ","  # ","  # ","#  # ","#    ","#    ","   # ","#  # ","#  # "
176 DATA " #### ","  #  ","### ","### ","#  # ","#### ","#### ","   # ","#### ","#### "
200 REM PROVIDES CHARACTER PATTERNS FOR SELECTED YEAR
220 PRINT/P CHR$(20);CHR$(17)
230 Y1=INT(YEAR/1000)
231 Y2=INT((YEAR-Y1*1000)/100)
232 Y3=INT((YEAR-Y1*1000-Y2*100)/10)
233 Y4=YEAR-Y1*1000-Y2*100-Y3*10
234 IF YH$="DELETED"THEN YH$="PRESENT":GOTO 300
240 FOR I=0TO6
250 PRINT/P TAB(45);YEAR$(Y1,I);"    ";YEAR$(Y2,I);"    ";YEAR$(Y3,I);"    ";YEAR$(Y4,I)
260 NEXT I
270 PRINT/P CHR$(16)
280 PRINT/P:PRINT/P:PRINT/P
300 REM CHECKS FOR LEAP YEAR AND START OF WEEKDAY
310 Y=YEAR-1900
311 A=Y+INT(Y/4)
312 W=A-INT(A/7)*7
313 T=YEAR/4:U=YEAR/100
320 L=1
325 IF (T=INT(T))*(U<>INT(U))THEN 350
330 W=W+1:L=0
```

```
335 IF U<>INT(U) GOTO350
340 W=W-1
350 FOR N=1TO4
355 IF D=7 THEN D=0
360 I=0:X=0:Y=0:Z=0
370 READ A$,B$,C$,F,G,H
375 IF (N=1)*(L=1) THEN G=29
400 REM ASSIGNS STARTING WEEKDAYS
420 REM ASSIGNS THE 1st WEEKDAY OF THE 1st MONTH
421 A=D+1
422 IF N=1 THEN A=W
423 IF A=7 THEN A=0
440 REM ASSIGNS THE 1st WEEKDAY OF THE 2nd MONTH
441 P=A+F
442 IF P>35 THEN B=7-(42-P)
443 IF P<=35 THEN B=7-(35-P)
444 IF B=7 THEN B=0
460 REM ASSIGNS THE 1st WEEKDAY OF THE 3rd MONTH
461 J=B+G
462 IF J<=35 THEN C=7-(35-J)
463 IF J>35 THEN C=7-(42-J)
464 IF C=7 THEN C=0
500 REM SETS UP THE HEADINGS FOR THE WEEKDAYS
510 PRINT/P TAB(17);A$;TAB(57);B$;TAB(97);C$
511 PRINT/P
520 PRINT/P TAB(11);"SUN  MON  TUE  WED  THU  FRI  SAT";
530 PRINT/P TAB(51);"SUN  MON  TUE  WED  THU  FRI  SAT";
540 PRINT/P TAB(91);"SUN  MON  TUE  WED  THU  FRI  SAT";
550 PRINT/P
560 X=X+1
570 GOTO 615
600 REM PRINTS DAYS ACROSS 3 MONTHS
610 A=0
615 IF (I>0)*(A=7) THEN 670
620 IF A=7 THEN 645
625 IF X<=F THEN PRINT/P TAB(A*5+12);RIGHT$(" "+STR$(X),2);:GOTO 635
630 GOTO 670
635 A=A+1
640 GOTO 560
645 Y=Y+1
650 GOTO 675
670 B=0
675 IF (I>0)*(B=7) THEN 750
680 IF B=7 THEN 710
685 IF Y<=G THEN PRINT/P TAB(B*5+52);RIGHT$(" "+STR$(Y),2);:GOTO 695
690 GOTO 750
695 B=B+1
700 GOTO 645
710 Z=Z+1
715 IF Z=H THEN D=C
720 IF (Z=H)*(C=7) THEN PRINT/P:GOTO 610
725 IF (X>F)*(Y>G)*(Z>H) THEN 785
730 GOTO 755
750 C=0
755 IF (X>F)*(Y>G)*(Z>H) THEN 785
760 IF C=7 THEN PRINT/P:I=I+1:GOTO610
765 IF Z<=H THEN PRINT/P TAB(C*5+92);RIGHT$(" "+STR$(Z),2);:GOTO775
770 PRINT/P:GOTO610
775 C=C+1
780 GOTO 710
785 PRINT/P:PRINT/P:PRINT/P
786 NEXT N
790 IF ME$="NIL"THEN ME$=""
791 PRINT/PTAB(124-LEN(ME$));ME$
792 IF ME$=""THEN ME$="NIL"
793 PRINTCHR$(6):CURSOR11,10:PRINT"C O M P L E T E D"
794 MUSICM1$,M2$:FOR WA=1TO2000:NEXT WA
795 GOTO 800
800 PRINTCHR$(6)
810 PRINTTAB(9);"CALENDAR MODIFICATIONS"
811 PRINTTAB(9);"_____________________"
820 CURSOR5,6:PRINT"YOU MAY CUSTOMISE YOUR CALENDAR":PRINT
821 PRINTTAB(4);"BY SELECTING ANY OF THE FOLLOWING":PRINT:PRINT:PRINT
830 MUSIC M4$
840 PRINT"       1. SELECT YEAR":PRINT
841 PRINT"       2. ENTER MESSAGE":PRINT
842 PRINT"       3. DELETE/ADD YEAR HEADING":PRINT
843 PRINT"       4. PROCEED WITH PRINTING":PRINT
844 PRINT"       5. EXIT PROGRAMME":PRINT:PRINT:PRINT
850 PRINTTAB(14);"(select 1-5)"
860 GET NO$:IF NO$=""THEN860
870 NO=VAL(NO$)
875 IF(NO<1)+(NO>5)THEN860
880 ON NO GOTO1000,1100,1200,1300,1400
900 REM DATA FOR MONTHLY HEADINGS AND NUMBER OF DAYS IN EACH MONTH
910 DATA #####  JANUARY  #####,##### FEBRUARY #####,#####    MARCH    #####,31,28,31
920 DATA #####    APRIL  #####,#####     MAY   #####,#####     JUNE    #####,30,31,30
930 DATA #####    JULY   #####,#####  AUGUST   #####,#####  SEPTEMBER #####,31,31,30
940 DATA #####  OCTOBER  #####,##### NOVEMBER #####,#####  DECEMBER  #####,31,30,31
1000 REM SELECT YEAR
1005 YS$=YR$
1010 PRINTCHR$(6)
1015 LINE 20,60,300,60,300,120,20,120,20,60
1020 CURSOR9,9:PRINT"Year required   ":PRINT
1021 PRINTTAB(13);"(enter 1800-2100)":PRINT
1022 PRINTTAB(9);"Present selection :";YR$
1030 CURSOR26,9:INPUT YR$
1040 IF(VAL(YR$)<1800)+(VAL(YR$)>2100)THEN YR$=YS$:GOTO 1010
1045 YEAR=VAL(YR$)
```

```
1050 PRINTCHR$(6)
1055 CURSOR 4,11:PRINT"Your selection is the year ";YR$
1060 CURSOR 11,22:PRINT"< agree Y or N >"
1061 MUSIC M3$
1065 GET YN$:IF YN$="" THEN 1065
1070 IF YN$="Y"THEN GRAPH C:GOTO 800
1075 IF YN$="N"THEN YR$=YS$:GOTO 1010
1080 GOTO 1050
1100 REM MESSAGE ROUTINE
1105 PRINTCHR$(6)
1110 CURSOR16,2:PRINT"MESSAGE"
1111 PRINTTAB(16);"-------":PRINT
1120 PRINT"   This option allows a display of a":PRINT
1121 PRINT"   message in reduced characters at":PRINT
1122 PRINT"   the bottom right corner after the":PRINT
1123 PRINT"   calendar has been printed (eg'By ":PRINT
1124 PRINT"   curtesy of the Peoples Republic":PRINT
1125 PRINT"   of China')"
1130 CURSOR1,18:PRINT"Present :";ME$
1140 CURSOR1,22:INPUT"New     :";ME$
1150 PRINTCHR$(6):CURSOR10,6
1160 PRINT"YOUR MESSAGE IS :":PRINT
1161 CURSOR0,12:PRINT ME$
1165 CURSOR 9,22:PRINT"< correct Y or N >"
1166 MUSIC M3$
1170 GET YN$:IF YN$=""THEN1170
1175 IFYN$="Y"THEN800
1180 IFYN$="N"THEN ME$="NIL":GOTO 800
1185 GOTO 1150
1200 REM DELETE/ADD YEAR HEADING
1205 PRINTCHR$(6)
1210 IF YH$="DELETED"THEN YH$="PRESENT":GOTO 1230
1220 YH$="DELETED"
1230 CURSOR10,10:PRINT"YEAR HEADING ";YH$
1240 MUSIC M3$
1250 FORWA=1TO2000:NEXT
1260 GOTO 800
1300 REM SUMMARY OF CUSTOMISING
1305 PRINTCHR$(6)
1310 CURSOR12,2:PRINT"C A L E N D A R"
1311 PRINTTAB(12);"---------"
1320 CURSOR0,6:PRINT"YEAR SELECTED : ";YR$:PRINT:PRINT
1330 PRINT"YEAR HEADING  : ";YH$:PRINT:PRINT
1340 PRINT"MESSAGE  : ";ME$
1350 CURSOR 11,22:PRINT"< agree Y or N >"
1360 MUSIC M3$
1365 GET YN$:IF YN$="" THEN 1365
1370 IF YN$="Y"THEN 95
1375 IF YN$="N"THEN 800
1380 GOTO 1365
1400 PRINTCHR$(6)
1410 END
```

```
0 REM*** CANNYON BY RAVI PANDEY***
0 REM***                     ***
0 REM***                     ***
1000 Z$="▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
1010 PRINT"▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣"
1020 PRINTZ$:IFC<>ATHEN1050
1030 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="◤   CANNYON  ◢":GOTO1050
1040 B=1:X$="◣  CANNYON  ◥"
1050 C=C+1:IFC=ATHENC=0:GOTO1030
1060 D=D+B:IF(D<0)+(D>26)THENB=B*-1:GOTO1060
1070 IFB=-1THENX$="◤   CANNYON  ◢"
1080 PRINTTAB(D):X$
1090 GETA$:IFA$=""THEN1020
1100 PRINT"               CANNYON"
1101 PRINTTAB(11);"────────▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣":PRINT"BY▣▣ RAVI PANDEY"
1102 PRINT"▣▣You are freely falling down a CANNYON.":PRINT
1103 PRINT"The objective of the game is to safely":PRINT:PRINT"steer your ma";
1104 PRINT"n through.▣":PRINT"WARNING:-This game is very addictive▣"
1105 PRINT"CAUTION:-The cannyon narrows at later▣":PRINT"stages of the game.▣"
1106 PRINT"Your score is the length of time you    survive.▣"
1107 PRINT"▣▣▣▣▣▣PRESS A KEY FOR CONTROLS"
1108 GETA$:IFA$=""THEN1108
1109 PRINT"▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣"
1110 PRINT"▣▣▣▣▣▣▣▣▣▣▣▣CONTROLS"
1111 PRINT"▣▣▣▣▣▣▣▣▣▣────────────────"
1112 PRINT"▣▣▣▣▣▣▣▣▣▣▣Q-LEFT▣▣▣":PRINT"▣▣▣▣▣▣▣▣▣▣E-RIGHT▣▣▣"
1113 PRINT"▣▣▣▣▣▣▣▣▣▣♦-MAN"
1114 PRINT"▣▣▣PRESS A KEY TO START"
1120 GETA$:IFA$=""THEN1120
1130 PRINT"▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣"
1140 TI$="000000"
1150 C=0:A=0:X=0:D=0
1160 Z$="▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
1170 PRINT"▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣▣"
1180 PRINTZ$:IFC<>ATHEN1210
1190 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="◤             ◢":GOTO1210
1200 B=1:X$="◣           ◥"
1210 C=C+1:IFC=ATHENC=0:GOTO1190
1220 D=D+B:IF(D<0)+(D>26)THENB=B*-1:GOTO1220
1230 IFB=-1THENX$="◤             ◢"
1240 PRINTTAB(D):X$
1250 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
1260 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
1270 IFPEEK(54008+X)<>0THEN1310
1280 POKE53968+X,68
1290 IFTI$="000010"THEN1390
1300 GOTO1180
1310 FORI=1TO100:POKE59555,0:FORZ=1TO3:NEXTZ:POKE59555,1:NEXTI
1320 FORI=1TO100:POKE59555,0:FORZ=1TO4:NEXTZ:POKE59555,1:NEXTI
1330 K=VAL(TI$)
1340 IFK>XXXTHENXXX=K:INPUT"HIGHEST SCORE:-NAME";N$
1350 PRINT"▣▣▣▣▣▣▣▣▣TIME-";K:PRINT:PRINT
1360 PRINT"HIGHEST SCORE ";XXX;" BY ";N$:PRINT:PRINT
1361 PRINT"PRESS A KEY TO START"
1362 FORI=1TO200:NEXT
1370 GETA$:IFA$=""THEN1370
1380 GOTO1140
1390 PRINTZ$:IFC<>ATHEN1420
1400 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="◤              ◢":GOTO1420
1410 B=1:X$="◣          ◥"
1420 C=C+1:IFC=ATHENC=0:GOTO1400
```

```
1430 D=D+B:IF(D<0)+(D)26)THENB=B*-1:GOTO1430
1440 IFB=-1THENX$="                    "
1450 PRINTTAB(D);X$
1460 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
1470 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
1480 IFPEEK(54008+X)<>0THEN1310
1490 POKE53968+X,68
1500 IFTI$="000020"THEN1520
1510 GOTO1390
1520 PRINTZ$:IFC<>ATHEN1550
1530 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="                    ":GOTO1550
1540 B=1:X$="                    "
1550 C=C+1:IFC=ATHENC=0:GOTO1530
1560 D=D+B:IF(D<0)+(D)27)THENB=B*-1:GOTO1560
1570 IFB=-1THENX$="                    "
1580 PRINTTAB(D);X$
1590 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
1600 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
1610 IFPEEK(54008+X)<>0THEN1310
1620 POKE53968+X,68
1630 IFTI$="000025"THEN1650
1640 GOTO1520
1650 PRINTZ$:IFC<>ATHEN1680
1660 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="                    ":GOTO1680
1670 B=1:X$="                    "
1680 C=C+1:IFC=ATHENC=0:GOTO1660
1690 D=D+B:IF(D<0)+(D)27)THENB=B*-1:GOTO1690
1700 IFB=-1THENX$="                    "
1710 PRINTTAB(D);X$
1720 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
1730 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
1740 IFPEEK(54008+X)<>0THEN1310
1750 POKE53968+X,68
1760 IFTI$="000030"THEN1780
1770 GOTO1650
1780 PRINTZ$:IFC<>ATHEN1810
1790 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="                    ":GOTO1810
1800 B=1:X$="                    "
1810 C=C+1:IFC=ATHENC=0:GOTO1790
1820 D=D+B:IF(D<0)+(D)28)THENB=B*-1:GOTO1820
1830 IFB=-1THENX$="                    "
1840 PRINTTAB(D);X$
1850 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
1860 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
1870 IFPEEK(54008+X)<>0THEN1310
1880 POKE53968+X,68
1890 IFTI$="000035"THEN1910
1900 GOTO1780
1910 PRINTZ$:IFC<>ATHEN1940
1920 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="                    ":GOTO1940
1930 B=1:X$="                    "
1940 C=C+1:IFC=ATHENC=0:GOTO1920
1950 D=D+B:IF(D<0)+(D)29)THENB=B*-1:GOTO1950
1960 IFB=-1THENX$="                    "
1970 PRINTTAB(D);X$
1980 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
1990 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
2000 IFPEEK(54008+X)<>0THEN1310
2010 POKE53968+X,68
2020 IFTI$="000040"THEN2040
2030 GOTO1910
2040 PRINTZ$:IFC<>ATHEN2070
2050 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="                    ":GOTO2070
2060 B=1:X$="                    "
2070 C=C+1:IFC=ATHENC=0:GOTO2050
2080 D=D+B:IF(D<0)+(D)30)THENB=B*-1:GOTO2080
```

```
2090 IFB=-1THENX$="                "
2100 PRINTTAB(D);X$
2110 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
2120 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
2130 IFPEEK(54008+X)<>0THEN1310
2140 POKE53968+X,68
2150 IFTI$="000048"THEN2170
2160 GOTO2040
2170 PRINTZ$:IFC<>ATHENC=C+1:GOTO2200
2180 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="                "
2190 B=1:X$="                "
2200 C=C+1:IFC=ATHENC=0:GOTO2180
2210 D=D+B:IF(D<0)+(D>31)THENB=B*-1:GOTO2210
2220 IFB=-1THENX$="                "
2230 PRINTTAB(D);X$
2240 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
2250 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
2260 IFPEEK(54008+X)<>0THEN1310
2270 POKE53968+X,68
2280 IFTI$="000055"THEN2300
2290 GOTO2170
2300 PRINTZ$:IFC<>ATHENC=C+1:GOTO2330
2310 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="                "
2320 B=1:X$="                "
2330 C=C+1:IFC=ATHENC=0:GOTO2310
2340 D=D+B:IF(D<0)+(D>32)THENB=B*-1:GOTO2340
2350 IFB=-1THENX$="                "
2360 PRINTTAB(D);X$
2370 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
2380 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
2390 IFPEEK(54008+X)<>0THEN1310
2400 POKE53968+X,68
2410 IFTI$="000100"THEN2430
2420 GOTO2300
2430 PRINTZ$:IFC<>ATHENC=C+1:GOTO2460
2440 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="                "
2450 B=1:X$="                "
2460 C=C+1:IFC=ATHENC=0:GOTO2440
2470 D=D+B:IF(D<0)+(D>33)THENB=B*-1:GOTO2470
2480 IFB=-1THENX$="                "
2490 PRINTTAB(D);X$
2500 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
2510 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
2520 IFPEEK(54008+X)<>0THEN1310
2530 POKE53968+X,68
2540 IFTI$="000105"THEN2560
2550 GOTO2430
2560 PRINTZ$:IFC<>ATHENC=C+1:GOTO2590
2570 A=INT(RND(1)*16)+1:IFA<8THENB=-1:X$="                "
2580 B=1:X$="                "
2590 C=C+1:IFC=ATHENC=0:GOTO2570
2600 D=D+B:IF(D<0)+(D>34)THENB=B*-1:GOTO2600
2610 IFB=-1THENX$="                "
2620 PRINTTAB(D);X$
2630 GETA$:IFPEEK(17828)=81THENX=X-1:POKE53968+X,68
2640 IFPEEK(17828)=69THENX=X+1:POKE53968+X,68
2650 IFPEEK(54008+X)<>0THEN1310
2660 POKE53968+X,68
2670 GOTO2560
4000 A=18440
4010 PRINTA;PEEK(A)
4020 FORX=18442TO18542
4030 IFPEEK(X)=13THENPRINTX+3;PEEK(X+3):X=X+4
4040 NEXTX
```

Dear Sirs,

Thank you for sending the Sharpsoft User Notes Issue No.7; I always enjoy the reading of these interesting pages. Please keep up the good work.

From time to time I notice that you can't go into every detail concerning members' queries. What do you think about the addition of two new columns to the Notes, namely "Readers Questions" and "Readers Answers"? This might possibly relieve you from providing unpaid soft and hardware consultancy, while it would satisfy the information seeking member at the same time.

In SUN 7 a couple of bugs crept up which are unfortunate insofar as they don't appear likely to enhance your sales. The fig-FORTH disc version is not only available for the MZ-80B. I run said FORTH under CP/M on the MZ-80K, and I recceived these very discs from you! And if FORTH runs under CP/M on the MZ-80B it should run on the MZ-80A too, as both machines use the same disc format.

The HISOFT Pascal 4 specification which appears in the supplement is only valid for the tape version. The disc version does very well support FILEs (of CHAR), and again, it is available for the MZ-80K too! (After all, it is the HP4D14 compiler which I use on the MZ-80K when I am not programming in Assembler (FORTH is only of academic interest to me.)).

I am sure, the above mentioned inaccuracies/omissions happened unintentionally, as it decidedly can't be your aim to frighten off potential customers who own an MZ-80K.

In the hopefully not too distant future I intend to bring up a Z80 version of the fig model on the MZ-80K (under CP/M), and if you are interested, I will gladly let you have a copy of it.

E. RAMM
FED. REP. OF GERMANY

Many thanks for pointing out our inaccuracies as, as you state, it is decidedly not our aim to frighten off potential customers. This Readers' Letters Section is intended as a Question and Answer service but unfortunately the marrying up of individual answers to their relevant questions, would be an extremely time consuming business. However, I hope that whilst combing through each Issue, members pick up useful, as well as useless, information along the way. Any letters we receive replying to queries in the User Notes, are included - so please keep writing.

EDITOR

****

Dear Sharpsoft User Notes,

I would like to thank other readers who have written in with hints, and would also like to offer a few items which I have noticed.

1. Firstly, a request to all readers offering BASIC program listings. With the increasing range of machines and extensions to SHARP BASIC it is becoming more and more likely that the reader wishing to use the program will have a different version of BASIC to that used by the program author. Whilst more of the differences can be got round relatively easily (e.g. PRINT@ can be replaced by copious use of the cursor keys), it would help if more program authors indicated which version of BASIC has been used (as some alredy do) to forewarn other users that changes may be needed.

One difference which I have noticed between SHARP EXTENDED BASIC (as used in several listings in S.U.N.) and SPEED BASIC (which I use) is the PRINT@ instruction.

```
EXTENDED BASIC uses PRINT@ line,column;
SPEED BASIC    uses PRINT@ column,line;
```

The changes required are easy and obvious - if you know which version of BASIC has been used - which is not always obvious from the program listing.

2. A further point about the PRINT@ instruction in SPEED BASIC. Like R.N. Humphries (S.U.N.5), I have had problems with random SYNTAX ERRORS traceable to this instruction. R.N.H. said that inserting a space before the @ would invariably cure this problem. Unfortunately, I have found that this is not quite the full story. There are still some (though fewer) problems but when I inserted another space after the @ I have no further problems. Thus instead of PRINT@X;Y use PRINT @ X,Y; Dare I claim to have finally laid this bug to rest? Or will it find some other way to rear its ugly head???

3. SHARP EXTENDED BASIC offers a SET(X,Y) function (to inspect a quarter of a character cell on the screen) which BASIC SP-5025 and SPEED BASIC do not. I have written a utility subroutine to carry out this function and a listing is enclosed. It includes notes on its use. Unfortunately it uses about 2.5K of memory but at least it could help will all but the longer programs.

4. I have entered the Word Processor progam WP2 (S.U.N.6) and found the problem at line 1180 prior to its mention in S.U.N.7. However, I also found that when I tried to use the option *F (Find) to correct a typing error, I obtained weird and wonderful results (it 'found' the old string in lines where it did not occur). I have re-written lines 2320-2400 inclusive to eliminate this problem. A warning, however, to anyone wishing to use this (or any other) word processor to replace, say, 'ARDVARK' by 'AARDVARK' - the string 'ARDVARK' will be found repeatedly, resulting in 'AAAAAAAARDVARK'. Instead, aim to replace, say, 'THE ARDVARK' by 'THE AARDVARK'.

I have also found that whilst the program will accept 'lines' of up to 255 characters, it is very likely that the program will fail due to a DATA ERROR when using the *A (Format) option if you use even nearly 255 characters per line. This is because the program joins spare characters from one line onto the front of the next line when formatting. It is therefore advisable to limit the length of lines entered to approximately the same length as lines will be after formatting. i.e. YOU provide course formatting but leave the find details to the program.

If you DO get such a DATA ERROR you will be automatically returned to BASIC. You can re-enter the program with minimal loss of text by using the DIRECT MODE instruction GOTO 2720. This returns you to the option menu so that you may continue. However, the text line immediately involved in the DATA ERROR will have been lost - so you should list the text to find what lines need re-inserting.

5. Finally - a request for information. Ken Gaston's program PARLIAMENT GAME (S.U.N.7) includes at line 230 the instruction CONSOLE C40. What does this mean please?

A. PALFREYMAN
SOUTH YORKSHIRE

I think you may find the articles **FURTHER IMPROVEMENTS TO WP2 by M.A. Hawes** and **WP1 HITS DISC by M. Belamy** of assistance with your Word Processor queries. The command CONSOLE C40 on the SHARP MZ-80B results in a 40 column screen display when it is already set to 80 columns.

EDITOR

```
59000 REM ** SET(ZX,ZY) FUNCTION SUBROUTINE IN TWO PARTS **
59010 REM *
59020 REM * A UTILITY SUBROUTINE WRITTEN USING SPEED-BASIC
59030 REM * BUT FULLY COMPATIBLE WITH SHARP BASIC SP-5025
59040 REM *
59050 REM * WRITTEN BY A. PALFREYMAN, BARNSLEY, SOUTH YORKSHIRE. (APRIL 1983)
59060 REM *
59999 REM ** SET(ZX,ZY) FUNCTION SUBROUTINE PART 1 **
60000 DIM XY(255)
60010 FOR Z=1 TO 255:XY(Z)=15:NEXT Z
60020 FOR Z=56 TO 62 STEP 2:XY(Z)=12:XY(Z+1)=10:NEXT Z
60030 XY(64)=0:XY(128)=0
60040 XY(110)=1
60050 XY(111)=2
60060 XY(42)=3:XY(48)=3:XY(52)=3:XY(54)=3:XY(98)=3:XY(103)=3:XY(112)=3
60070 XY(116)=3:XY(122)=3:XY(157)=3:XY(158)=3:XY(168)=3:XY(169)=3:XY(211)=3
60080 XY(219)=3
60090 XY(76)=4
60100 XY(49)=5:XY(53)=5:XY(55)=5:XY(113)=5:XY(117)=5:XY(123)=5:XY(176)=5
60110 XY(180)=5:XY(210)=5:XY(217)=5:XY(222)=5
60120 XY(45)=6:XY(108)=6:XY(118)=6
60130 XY(6)=7:XY(16)=7:XY(39)=7:XY(86)=7:XY(114)=7:XY(134)=7:XY(146)=7
60140 XY(177)=7:XY(187)=7:XY(213)=7:XY(230)=7
60150 XY(75)=8:XY(92)=8
60160 XY(89)=9:XY(91)=9:XY(119)=9:XY(227)=9:XY(238)=9
60170 XY(28)=10:XY(30)=10:XY(79)=10:XY(97)=10:XY(121)=10:XY(125)=10:XY(127)=10
60180 XY(179)=10:XY(183)=10:XY(209)=10:XY(218)=10:XY(221)=10:XY(225)=10
60190 XY(20)=11:XY(66)=11:XY(80)=11:XY(106)=11:XY(115)=11:XY(178)=11
60200 XY(184)=11:XY(228)=11
60210 XY(46)=12:XY(47)=12:XY(93)=12:XY(95)=12:XY(120)=12:XY(124)=12:XY(126)=12
60220 XY(160)=12:XY(167)=12:XY(185)=12:XY(186)=12:XY(212)=12:XY(214)=12
60230 XY(224)=12
60240 XY(12)=13:XY(50)=13:XY(69)=13:XY(77)=13:XY(159)=13:XY(181)=13:XY(215)=13
60250 XY(235)=13
60260 XY(10)=14:XY(44)=14:XY(51)=14:XY(78)=14:XY(88)=14:XY(90)=14:XY(94)=14
60270 XY(138)=14:XY(156)=14:XY(182)=14:XY(189)=14:XY(216)=14:XY(226)=14
60280 FOR Z=0 TO 15:XY(Z+240)=Z:NEXT Z
60290 RETURN
60999 REM ** SET(ZX,ZY) FUNCTION SUBROUTINE PART 2 **
61000 ZZ=1
61010 IF ZX/2<>INT(ZX/2) THEN ZZ=ZZ+1
61020 IF ZY/2<>INT(ZY/2) THEN ZZ=ZZ+2
61030 ZX=INT(ZX/2):ZY=INT(ZY/2)
61040 XY=53248+40*ZY+ZX:XY=PEEK(XY):XY=XY(XY)
61050 ON ZZ GOTO 61060,61080,61100,61120
61060 IF XY/2<>INT(XY/2) THEN 61140
61070 GOTO 61130
61080 IF (XY=2)+(XY=3)+(XY=6)+(XY=7)+(XY=10)+(XY=11)+(XY=14)+(XY=15)THEN 61140
61090 GOTO 61130
61100 IF ((XY)3)*(XY(8))+((XY)11)*(XY(16)) THEN 61140
61110 GOTO 61130
61120 IF (XY)7)*(XY(16) THEN 61140
61130 XY=0:RETURN
61140 XY=-1:RETURN
```

```
61150 REM
61160 REM
61170 REM ************************
61180 REM *                      *
61190 REM * INSTRUCTIONS FOR USE *
61191 REM *                      *
61192 REM ************************
61200 REM
61210 REM * PART 1 OF THIS UTILITY SUBROUTINE MUST BE CALLED BY 'GOSUB 60000'
61220 REM * AT OR NEAR THE BEGINNING OF THE PROGRAM TO DIMENSION & SET UP THE
61230 REM * 'LOOK-UP TABLE'
61240 REM *
61250 REM *
61260 REM * PART 2 IS USED AT ANY POINT IN THE PROGRAM WHERE THE SET(X,Y)
61270 REM * FUNCTION PROVIDED BY SHARP EXTENDED BASIC WOULD BE USED.
61280 REM *
61290 REM * TO USE PART 2 -
61300 REM * 1. PUT THE X & Y CO-ORDINATES OF THE POINT ON SCREEN TO BE CHECKED
61310 REM *    INTO VARIABLES ZX & ZY RESPECTIVELY
61320 REM * 2. CALL PART 2 BY 'GOSUB 61000'
61330 REM * 3. ON RETURN FROM THE SUBROUTINE, IF THE POINT CHECKED IS SET BY
61340 REM *    USE OF THE 'SET X,Y' INSTRUCTION, OR IF THAT POINT CONTAINS ANY
61350 REM *    PART OF ANY CHARACTER (INCLUDING GRAPHIC SYMBOLS), THEN THE
61360 REM *    VARIABLE XY HAS THE VALUE -1, OTHERWISE IT HAS THE VALUE 0
61370 REM *
61380 REM * eg IF THE POINT (15,35) IS TO BE CHECKED USING THE STATEMENT
61390 REM * 'IF SET(15,35) THEN ___', THE FOLLOWING SEQUENCE SHOULD BE USED -
61400 REM *
61410 REM * "ZX=15:ZY=35:GOSUB 61000:IF XY THEN ___"
61420 REM *
61430 REM * NOTE 1 - THE ARRAY XY(255) MUST NOT BE USED AT ANY OTHER PLACE IN
61440 REM * THE PROGRAM UNTIL AFTER THE LAST USE OF THIS SUBROUTINE
61450 REM * ALSO, THE VARIABLES ZX, ZY, Z & ZZ MUST NOT BE IN USE FOR OTHER
61460 REM * PURPOSES WHEN THIS SUBROUTINE IS CALLED.
61470 REM *
61480 REM * NOTE 2 - PART 1 COULD HAVE BEEN SHORTENED BY USE OF 'READ' &
61490 REM * 'DATA' STATEMENTS, BUT THIS WOULD CAUSE PROBLEMS IN ANY PROGRAM
61500 REM * WHICH USES 'READ' & 'DATA' IN ITS MAIN BODY
61510 REM *
61520 REM * NOTE 3 - THESE 'REM' STATEMENTS FROM LINE 61150 SHOULD NOT BE
61530 REM * TYPED IN AS PART OF THE SUBROUTINE - THEIR ONLY USEFUL PURPOSE
61540 REM * IS TO GUIDE A PROSPECTIVE USER.
```

```
2200 REM * AMMENDMENT TO WP2 (SHARPSOFT USER NOTES ISSUE 6 *
2202 REM *
2204 REM * BY A. PALFREYMAN, BARNSLEY, SOUTH YORKSHIRE
2206 REM *
2210 REM *
2220 REM * REPLACE EXISTING LINES 2320-2400 BY THOSE SHOWN BELOW
2230 REM *
2240 REM * DO NOT INCLUDE THESE 'REM' STATEMENTS
2250 REM
2260 REM
2320 FOR M=XTOV:PRINTM;:LS=LEN(A$(M)):IFLO>LSTHENNEXTM
2330 FOR N=1TOLS-LO+1:IFO$=MID$(A$(M),N,LO)GOSUB2350
2340 NEXTN,M:PRINT:GOTO170
2350 IFLS=LO+LN>240THENPRINT"REPLACEMENT WOULD MAKE LINE";M;" TOO LONG":RETURN
2360 IFN=1THENA$=N$+RIGHT$(A$(M),LS-LN):GOTO2390
2370 IFN=LS-LO+1THENA$=LEFT$(A$(M),N-1)+N$:GOTO2390
2380 A$=LEFT$(A$(M),N-1)+N$+RIGHT$(A$(M),LS-N+1-LO)
2390 IFRL=1THENA$(M)=A$
2400 PRINT:PRINT:PRINT M;" ";A$(M):PRINT:RETURN
```

Dear Sirs,

My Sharp MZ-80K is equipped with Disk Drive and BASIC 6021 (for the Xtal CP/M conversion) and in consequence a large number of the suggestions for PEEKs and POKEs do not work.

For example POKE 57346, 4 and POKE 57346, 6 have the same effect i.e. they both blank the screen momentarily when entered in the direct mode and need to be programmed for a longer lasting effect.

It may be of interest to note that in the programming mode POKE 5955, 0 +N*16 also blanks the screen and POKE 59555, 1 +N*16 restores the screen when N is any integer giving a POKE value of less than 255.

I find the utilities excellent but would like to suggest two areas of difficulty are overcome by:

a) In the Table.DP Utility making Line 60860 read LN$ = STR$ (PEEK(A+2) + PEEK (A+3)*256) + " " : A=A+3

This separates the individual line numbers and makes them easier to read.

b) The ability to APPEND the Utility can be achieved by the following procedure (assuming the programmed for RENUMBER-ing (say) is called OLD PROG)

i) SAVE "OLD PROG" to Disk.

ii) LOAD "RENUMBER" and LIST ensuring that the first line (60,000 ?) of RENUMBER is not less than three lines from the top of the screen.

iii) With the cursor of the bottom line of the screen LOAD "OLD PROG".

iv) When the READY prompt appears, HOME the cursor. This should place it at the beginning of the LISTED RENUMBER programme.

v) Press CR to enter each line displayed on the screen. This will gradually bring the cursor down the screen until it is just below LCAD "OLD PROG".

vi) Insert two spaces before LOAD "OLD PROG" and key DELETE in place of LOAD with CR.

vii) When the READY prompt appears, key SAVE "OLD PROG" with CR.

viii) When the READY prompt appears repeat Steps. ii) to vii) above but each time ensure that the last of the RENUMBER lines just previously entered is at the top of the screen.

The above procedure certainly will eliminate keying errors and (depending on the time OLD PROG takes to LOAD, DELETE and SAVE) is much quicker than manually entering the Utility lines.

All I need now is a Utility program that will DELETE all lines greater than 60,000. Any offers ?

A minor "bug" in the Utility Programme RENUMBER is that all GOTO's and GOSUB's in the original programme are replaced by 0's and consequently have to be re-entered. Is there a cure ?

Thanks for the Notes.

G.H. BENT
CORBY

****

Dear Sirs,

I am working with an MZ-80K with Disc, in normal Disc BASIC and Knight Disc Commander.

1. What "PEEKS" are different in Disc BASIC, apart from PEEK Protect? Is there a listing of DISC PEEKS, if considerably different. I have the regular list of Tape Based Peeks from S.U.N.

2. Where is there a list of "USR( )" routines? They are referred to in the BASIC MAnual, and Disc Manual, but no where does there appear to be a list of them.

3. When transferring to "FDOS", do "PEEKS" and "USR7s" translate correctly? I can't spot it in the FDOS Manual.

If there are any publications I could buy to cover these points, please let me know.

M. BELLAMY
SUSSEX

1. As far as we know there has not been a comprehensive list of PEEKs and POKEs produced from Disk BASIC, however, you may find the odd one or two listed e.g. (in Hexadecimal)

POKE $208B,0	Allows repeat key function on the GET statement
POKE $208B,$BE	Restores to normal
POKE $1F70,0	Allows quotes to be printed
POKE $1F70,$22	Restores to normal
POKE $1532,$04 and POKE $1533,$12	Allows equations after a GOTO statement e.g. GOTO 100+(A*10)
POKE $155D,$04 and POKE $155E,$12	Allows equations after a GOSUB statement
POKE $1E70,1	Removes PEEK protect
POKE $1E70,0	Restores to normal

Also see Mr. Wout Van Haaster's letter below. Can any other users help us with this problem as it is very often requested?

2. Try Issue No.1 of our User Notes pages 32-39. It is more related to machine code - but it might help.

EDITOR

****

Dear Sirs,

I've just "discovered" the BASIC "SP-6015 equivalents for some of the useful SP-5025 peeks and pokes:

POKE $2A60,0	allows the comma to be entered in an input string or when a file is read from the disc poke $2A60,44 to reset (this is necessary when reading data or when you require inputs like : INPUT A$,B$,C$
POKE $2A63,0	allows double quotes (") to be entered in input strings etc. poke $2A60,34 to reset.
POKE $1EE3,0	allows CHR$ <32 to be printed for instance print CHR$(22)=clear screen.
POKE $1EBF,$IE POKE $1EC2,$3B	allows AND to be used
POKE $1EC2,$3B POKE $1EC3,$2B	allows OR to be used
POKE $1F70,1	removes the peek protection
POKE 25902,0	makes first line of basic program 0

Addresses

24921-24926	ASCII of last TI$ used
24931-24932	Highest memory byte
24933-24934	Stores LIMIT address
25268-25269	Stores line number being executed

So far I've not seen any peeks and pokes regarding SP-6015 BASIC so I think this may be of interst for SUN-readers!

As soon as I've found out the rest of the peeks and pokes I'll let you all know.

WOUT VAN HAASTER
THE NETHERLANDS

****

Dear Sir,

Thank you for sending the SUN, which I received today. First I would like to answer Mr. Fernando that you can Append programs on the MZ-80B but you need a special BASIC called BASIC SB-6511 (RS 232).

I would like to make my own contribution to the quality of the SUN.

On a SHARP MZ-80B

1) to draw the symbols:
poke$08f2,$18 :and then you PRINTCHR$(i) i=1,..,6
back to normal:
poke$08f2,$20

2) to hear a beep every time you touch a key :
poke$0732,205
back to normal:
poke$0732,204

3) to use the indirect functions:LIST, DIR etc. in a program :
poke$1C10,$9 : poke$1C19,$C9
(you will be able to ask for a directory in a program)
Return to normal: poke$1C10,229 : poke$1C19,205

4) Could anyone tell me if there is a compiled version of double precision basic that will run on FDOS. Because the double precision basic SB-6610 contains a fatal error:

When you delete a program from a disk, it isn't really deleted. It's not anymore in the DIR(ectory) but it is still in the disk catalogue (you can only read this catalogue with a special program). In front of every deleted file there is a DEL. (this is the reason why:

1. You can recall deleted programmes if you know how to.
2. You can have error 53 even though the disk is not full.
3. When you save a file (or program) it will come on top of the file (or program) with the same name that you have deleted before. Never on top of any other deleted file (or program).

This is not the error, the error is in renaming a program or file:

If you rename a file "A", "B", then the computer will look if another file on the disk is called "B", if not he will rename "A", "B". However, the computer does not look among the programs or files with a DEL in front of them. The result is that if you delete and rename a lot of the same file, on the disk there will be 3 or 4 files with the same name with a DEL in front and 1 with the same name with a DEL in front and 1 with the same name but without a DEL in front (the only one you will see if you type DIR.

As the computer saves all new files on top of old files with the same name, the computer doesn't know where to save a new file with the same name as there are 3 or 4 already. At this stage the computer starts mixing them all up together, so you lose all the information on your new file.

The same error exists on the BASIC SB-6510, but I haven't found it on the compiled basic that works on FDOS (what a shame it isn't double precision).

E. BIRNBAUM
BELGIUM

****

Dear Sir,

MZ-80B

Ref. Issue 4 Pages 91-95; letter from Wymark, Birmingham.
Ref. Issue 6 Page 34; letter from Stanley, Notts.

To make a copy of any machine code programmed - including a copy of Sharp BASIC SB-5510, version 1.1.

1. Load BASIC SB-5510 version 1.1.

2. Transfer system control to Monitor
   Type MON then (CR)

3. Change loading address to 8000H (any place is OK between 7000H and D000H).

```
*M                    (CR)
M-ADR.$02B9           (CR)
02B9     2A     21    (CR)
02BA     DA     00    (CR)
02BB     10     80    (CR)
02BC     18           (BREAK)
*
```

4. Put in tape with the programme to be copied (SB-5510).

```
*L                    (CR)
FILE NAME:            (CR)
LOADING BASIC SB-5510
** MONITOR SB-1510 **
*
```

5. Check how big the programmed is.

```
*D                  (CR)
S-ADR.$8000         (CR)
E-ADR.$FF00         (CR)
```

The contents of the memory from 8000H to FF00H will be dumped onto the screen. Check when the programme has finished, i.e. continuos 00H. In the case of the BASIC SB-5510 version 1.1 programmed it is about C970H.

6. Put in blank tape.

```
*S                  (CR)
FILE NAME: BASCOPY  (CR)
S-ADR.$8000         (CR)
E-ADR.$C970         (CR)
J.ADR.$1280         (CR)
WRITING BASCOPY
```

The tapecounter goes up to about 077, don't worry. When the recording is finished the copy is ready to use. Whilst the programme is stored in the memory from 8000H any machine code changes may be made.

Regarding Peek and Pokes on the MZ-80B with Sharp Basic 5510 version 1.1 I think the Editor should inform us if there are any other versions of Basic and these should be mentioned in the User Notes to avoid confusion. The Basic Text, that is the first line number is at 20766 or 511EH.

The Utilities Programme which Wymark adapted for the MZ-80B should be changed (with the 1.1 version) to:-

```
line 60430  A=20572  changed to  A=20764
line 60930 L2=20571  changed to L2=20763
```

The "Variables Table" and the "String Search" will then work OK. I have not had time to check the "Renumber" utility but see no reason why it should not work if the above values are used.

Can anybody help with my problems? I have an early MZ-80B (Oct. 1981), Printer P5 and Basic SB-5510 version 1.1. Everything works fine. I bought several packages from KUMA with the following results:

APOLLO word processing package. It ignores the command to Print-out?

WDPRO word processing package. On the command to Print-out it simply feeds the paper continuously through the printer?

DISASSEMBLER (ZEN COMPATIBLE). On the command to Print-out it prints continuously on the same line, i.e. no paper feed.

Finally I purchased a Knights Disassembler - Hurrah a print-out at last. I used the Disassembler to look at the other machine code programs. I discovered that none of the programs used the recommended printer control subroutines given by Sharp in the MZ-80 P5 Printer Manual (pages 23 and 24).

It is obvious that these programs will run on some models of the MZ-80B or MZ-80 P5 without problems. Does anybody know where the problem is?

J. HUNT
SWITZERLAND

****

HELP

Re: The last page of the User Notes "Example Progs", I did type:

0 SUBMIT <CR> and got SYNTAX ERROR
and 12 PTAB <CR> and got SYNTAX ERROR!

WHY?

M.E. CREW

We think you are working in BASIC (hence the SYNTAX ERRORS) instead of Fig-FORTH!

EDITOR

****

Dear Sir,

Please see the enclosed copy of the display code table for the MZ-80K; can all the graphics characters be accessed directly from the keyboard? Those I have marked I have not yet discovered how to display. I have used the CHR$(A); type command to display same but can I "PRINT" them as part of a program design?

Can you also please explain why the :PRINT@12,22; etc.. in the "STORM" program by Peter Chase gives my poor old Micro "brainache", is it perhaps the "@" which we do not seem to be able to emulate successfully?

A. BINGHAM
HANTS

There are two immediate ways of displaying the hidden characters. The first you already know i.e. CHR$( ) method. The second is to poke the characters off the screen e.g. POKE 53248,202 displays a little man in the top left hand corner. The book PEEKING & POKEING the Sharp MZ-80K @ £4.30 (including postage) should help.

****

Gentlemen,

Thank you for the FORTH Tapes which arrived safely yesterday. I have checked them out and everything appears to be O.K. (I note that the BUG in Screen 9 has already been corrected).

There is one thing which I would like you to explain however, and that is why do one of the "Editor" screens at times LIST in "heiroglyphics" and what causes it to LIST correctly (or should I say intelligably) at other times and how does one set out to make it readable if one requires to have access to the information, or alternatively how does one de-cypher it?

J.V. VAN SCHOOR
REPUBLIC OF SOUTH AFRICA

****

Dear Sirs,

With regard to the Forth language tapes:-

You say it is necessary to write a bootstrap editor to correct the fault on Screen 9, this is not so - just load the editor as normal i.e. Decimal 7 Load, then Decimal 10 Load and you will then have all the editor screens in memory.

You then enter the editor via EDITOR and list screen 9 and make the necessary modifications i.e.

12 P -->, 15 E Then Flush - this will save the amended screen t6o tape.

Your counter settings for the screens on tape do not match up with mine i.e.:

```
SCR 7   45 - 50
SCR 8   50 - 57
SCR 9   57 - 62
SCR 10  62 - 69  and so on.
```

Is there a fault in your Forth language, because it will not completely load any screens which contain blank lines between definitions etc. i.e. SCREEN 9 EDITOR. If this is not a fault it was not made clear (not stated at all) that you could not separate lines written on the screen by blank lines i.e.

```
0    :    PRINT . ;
1
2    :    ADD   + ;
```

will not load/compile after line ONE!

In the two books on FORTH you sent me, they are allowed to use blank lines in the screen, this makes the screens more readable, I hope this information will be of some help, if not to you then to other readers.

How do you use definitions created by the assembler, on Screens? Do you have to have the assembler loaded every time you need these definitions?

R. WARD
TELFORD

We have found that the MZ-80K and MZ-80A tape counter settings are not the same for the FORTH EDITOR tape. The previously published settings were obtained from the MZ-80A. To use screens with FORTH words defined in assembler language the assembler must be loaded into the fig-FORTH system.

EDITOR

****

Dear Sharpsoft,

I would like to thank you for the rapid sending of the book "Introduction to FORTH" which arrived only two days after sending off for it. However, I have a few problems with the FORTH tapes which were supplied with Issue 4 of the User Notes. The following 'tutorials' do not function properly:-

5 (SCREEN-CLEAR1 and SCREEN-CLEAR3)
6 (the solution on p54 of Issue 5 does not work either)
7
8 (the second program)
9 (works partially - does not work with 170)
10 (crashes FORTH but solution on p41 of Issue 6 works)
11 (same result as printed on p48 of Issue 6)
(solution on p41 of Issue 6 appears to generate a continuous loop)

Also, the OK message does not appear on a new line after entering a command and I have seen none of the graphics error messages mentioned in yur User Notes. A couple of questions:-

1) How can the FORTH programs be broken into? and
2) Is there any sound available in FORTH?

If the screen is cleared (in FORTH) by pressing SHIFT and CLR/HOME anything entered on the top line gives an error.

On pages 46 and 47 of Issue 5 there are two similar programs for cursor addressing. However, in the program on p46 there is a piece of data missing and on p47 the method of execution is incorrect. The problem can easily be solved by entering the version on p47 and instead of using 'normal' brackets, the 'square' brackets used in the first version should be used. I have found the brackets can be changed to other characters in the following way:-

Using the second listing the number 91 in the data in line 1 and the number 93 in line 5 should be replaced with the ASCII code of the character to be used in place of opening and closing the brackets respectively. Thus, the 91 could be replaced with 64 for @ and 59 could be used instead of 93 for a semi-colon.

This works perrectly while keeping the MZ-80K switched on, but when reloading a recorded version of the updated BASIC, as soon as loading is complete, the BASIc crashes and cannot be accessed.

Regarding C. Choi's letter on p41 of issue 6, as stated in the summary of the FORTH in Issue 4, SHIFT and INSERT turns the printer on or off, depending upon its state (i.e. on/off) at the time.

On to the recurring topic of continuous input via the GET function.

i) PEEKing and POKEing at location 17828 has no effect on GET on my MZ-80K.

ii) The routine on p47 works well and, in my opinion, is the most convenient method.

iii) I have another routine which I have 'extracted' from a program I found in a magazine:

Enter this routine into the program:

```
FOR A=24553 TO 34560:EAD B:POKEA,B:NEXTA
DATA 205,27,0,50,240,95,201,0
```

When a keyboard input is required enter USR (24553) and then K=PEEK(24560) where K is the ASCII value of the key being pressed. N.B. If the program is quite long, the last couple of lines may have some characters replaced by graphics symbols, so it is advisable to have the initial subroutine at the end of the program, and it need not be used again in the running of the program.

Another useful routine which I have 'extracted' from a program was in that great little game "STORM" on p57 of Issue 6. It is a routine for downward scrolling. It reads:

```
LIMIT53236:FORI=53236 TO 53247:READP:POKEI,P:NEXT
DATA 1,32,3,17,071,211,33,031,211,237,184,201
```

Whenever the scroll is required enter USR(53236). The top 21 lines will scroll downwards, leaving space below for recording score etc. in games. One disadvantage is that it reprints the top line again at the top, so it is a good idea to keep this line blank. I would like to know how to alter the length of the scroll - does anyone know how to? Have you got a routine for later scroll?

Finally, I enclose a short program to convert a hexadecimal number into decimal.

Thanks for the User Notes and keep them up to the standard of Issue 6 which was excellent.

If you type into your program that you have a copyright on it, is it legal?

```
100  REM HEX TO DECIMAL CONVERTOR
105  PRINT "C";: REM CLEAR SCREEN
110  DIM DG$(15)
115  FOR I=0 TO 15 : READ DG$(I): NEXT I
120  INPUT "  NUMBER IN HEX:  ";HN$
125  IF RIGHT$(HN$,1)="H"THENHN$=LEFT$(HN$,LEN(HN$)-1)
130  IF LEN(HN$) > 4 GOTO 120
135  IF LEN(HN$) = 4 GOTO 145
140  FOR I = 1 TO 4-LEN(HN$):HN$="0"+HN$:NEXTI
145  NT=0:FORI=LEN(HN$) TO 1 STEP-1:x=0
150  FOR J=0 TO 15:IFDG$(J)=MID$(HN$,I,1)THENNT=NT+(J*16(4-I):x=1
155  NEXT J:IFX=0 GOTO 120
160  NEXT I
165  PRINT "NUMBER IN DECIMAL=";NT
170  GOTO 120
175  DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
```

D. HALE
LANCS

****

Dear Sir,

I recently purchased a copy of NEW FORTH and have found this version of FORTH most interesting and much more convenient re: editing and would like to offer the attached for the S.U.N. Newsletter.

To change ECHO to the more familiar EMIT, PATCH as follows:

```
4D 1825 C!  491826 C!  54 1827 C!
```

New word VLIST lists the complete dictionary in the same format as HELP:

```
: VLIST HEX 174C 1B44!  HELP 2CDF 1B44! ;
```

Note VLIST should be run to completion to enable the patch after HELP otherwise if the BREAK KEY is pressed during operation future uses of HELP will have the same function as VLIST.

New word DUMP will list addresses from specified address and contents.

```
: DUMP HEX CFFF OVER DO I . SPACE I
  C@ . SPACE I C@ EMIT ? WAIT CR
  LOOP ;
```

To use DUMP input <START ADDRESS. DUMP any key will hold listing Break will stop. CFFF is upper limit and can be set to suit the user.

The following is a fill screen word SCR, that uses a MC code routine within a new word definition.

```
CODE        AOPEN   FILL

            00 C    MVI
          D000 H    LXI
               O    LBL
            C  M    MOV
              D3    CPI
            O  V    JNZ
            L  A    MOV
              E8    CPI
            O  V    JNZ
                    RET

          ACLOSE

: SCR HEX 100 0 DO I 2D94 C! FILL LOOP ;
```

The adress 2D94 is dependent on where the MC code section is located in the dictionary and is the address (data byte) of the first instruction (00 C MVI). This location can be found after submitting the MC Section screen by using the HELP and DUMP commands.

The Bellring program in the notes should have a PSW PUSH before and PSW POP after the call to monitor or this routine does not save A and flags.

I.C. Butterworth
BORNEO

****

Dear Sharpsoft,

BACH'S BYTES - TWO PART RECORDING + MIX

"Bach's Bytes" is a two part/two track recording. The two parts were mixed after being made separately by the MZ-80K. Each track was made very simply by getting the computer to start the cassette recorder, have a pause, and then cue + pips, and then the music - all to ensure that the music is on the tape after a certain time. The latter track would have the same beginning of program.

The tape recorder was a stereo dual-cassette deck by SHARP. The input was connected to the right/left channel from the computer's speaker output (disconnect the speaker and use the two leads to go to line input).

The tape recorder was switched on by the computer after the tape has been rewound and the RECORD/PLAY button pressed, so that computer starts recording from the beginning of the tape. To switch on the recorder, an interface most trigger a relay suitable for A.C. currents.

The program here is a Prelude by Bach. type in the top part first, record onto tape; then run the bottom part, and record onto tape.

Then, once the two tapes have one part on each, the duel cassette deck was used to start them both at the same time. This time, the line out was fed into a Hi fi system (or another cassette recorder).

You should now have a finished product. It will be worth all the effort if it is good! If you have any problems with sync, then you must experiment a little.

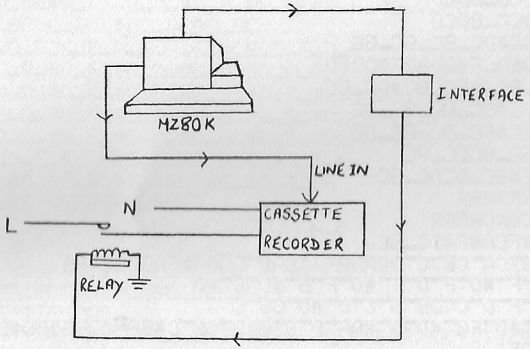
A professional recording technician I know,classed the task as "impossible because of drift"! But - I have done it.

If you haven't got the necessary equipment, then send a S.A.E. and a cheque for £3.00 to:

19 Madingley Road,
Cambridge, CB3 OEG

A tape will be sent to you containing "BACH'S BYTES".

P. HENDRIE
CAMBS.



```
100 OUT#1,255                          — TAPE RECORDER ON
110 TEMPO4
120 MUSIC"R9R%C1R5C1R5C1R9R            — DELAY — CUE PIPS — DELAY
130 TEMPO6
140 MUSIC""C1#DD#DC#DD#D""C#DD#DC#DD#D  — MUSIC START
150 MUSIC"#GFEFCFEF#GFEFCFEF
160 MUSIC"BF#DFDF#DFBF#DFDF#DF
170 MUSIC""CGFG#DGFG""CGFG#DGFG
180 MUSIC""#D#GG#G#D#GG#G""#D#GG#G#D#GG#G
190 MUSIC""D#FE#FD#FE#F""D#FE#FD#FE#F
200 MUSIC""DG#FGDG#FG""DG#FGDG#FG
210 MUSIC""CEDECEDE""CEDECEDE
220 MUSIC""CFEFCFEF""CFEFCFEF
230 MUSIC"#AF#DFDF#DF#AF#DFDF#DF
240 MUSIC"#AGFG#DGFG#AGFG#DGFG
250 MUSIC"#GGFG#DGFG#GGFG#DGFG
260 MUSIC"#GDCD_#ADCD#GDCD_#ADCD
270 MUSIC"G_#A_#G_#A#D_#A_#G_#AG_#A_#G_#A#D_#A_#G_#A
280 MUSIC"FC_#AC_AC_#ACFC_#AC_AC_#AC
290 MUSIC"FDCD_BDCDFDCD_BDCD
300 MUSIC"FDCD_BDCDFDCD_BDCD
310 MUSIC"#DC_BC_GC_BC#DC_BC_GC_BC
320 MUSIC"_F#DD#DF#DD#D_F#DD#DF#DD#D
330 MUSIC"_#FC_BC#DC_BC_#FC_BC#DC_BC
340 MUSIC"#DC_BC_GC_BC#DC_BC_GC_BC
350 MUSIC"#FC_BC_AC_BC#FC_BC_AC_BC
360 MUSIC"GC_BCDC_BCGC_BCDC_BC
370 MUSIC"#GC_BCDC_BC#GC_BCDC_BC
380 MUSIC"RRRRRRRRRRRRRRRR
390 MUSIC"RRRRRRRR#DCG#DRRRR
400 MUSIC"RRRC#DC_BC#FCA#F#DC_BC
410 MUSIC"R""D""C""#D""CB""CA""CB""C""DBABG","BAB""CAGA#FAGABG#FG
420 MUSIC"D""G""F""G""#G""F""#D""F""D""F""#D""F""G""#D""D""#D
430 MUSIC""C""#D""D""#D""F""D""C""DB""D""C""D""#D""CB""C
440 MUSIC"G""CB""C#G""F""#D""FG""#D""D""#DF""D""C""D#D""CB""C#GF#DFG#DD#DFDCD
450 MUSIC"E6R3":TEMPO5
460 MUSIC"E0CDEFG#G#A""C#A#GGF1G0EF6R3F0GFEFG#GGF#DD#DFD#DF":TEMPO6
470 MUSIC"_B4_D1_F_#G_G_F_B_FD_F_E_#G_G_F
480 MUSIC"_E#C_#A_GC_#G_F_#G_G_#A_G_#D_#G_F_D_F
490 MUSIC"_E_G_E_C_F_D_B_DR_G_C3_D1_E_G_#A_G_#GCFDF2#G""C3B""C4G6F7D8E9E9
```

TREBLE PART

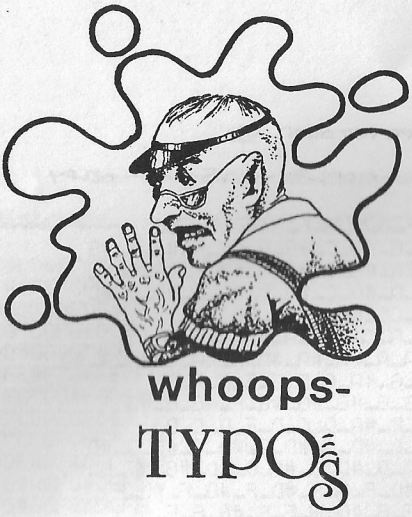
```
OUT#1,255
TEMP04
MUSIC"R8
TEMP06
MUSIC"_C1_G_F_G_#D_G_F_G_C_G_F_G_#D_G_F_G
MUSIC"_C_#C_G_#G_F_#G_G_#G_C_#G_G_#G_F_#G_G_#G
MUSIC"_C_#G_G_#G_F_#G_G_#G_C_#G_G_#G_F_#G_G_#G
MUSIC"_C_#D_D_#D_G_#D_D_#D_C_#D_D_#D_G_#D_D_#D
MUSIC"_CC_#AC_#GC_#AC_CC_#AC_#GC_#AC
MUSIC"_C_A_G_A_#F_A_G_A_C_A_G_A_#F_A_G_A
MUSIC"_#A_#A_A_#A_G_#A_A_#A_#A_#A_A_#A_G_#A_A_#A
MUSIC"_#A_G_F_G_E_G_F_G_#A_G_F_G_E_G_F_G_
MUSIC"_#G_#G_G_#G_F_#G_G_#G_#G_#G_F_#G_G_#G
MUSIC"_#A_D_C_D_F_D_C_D_#A_D_C_D_F_D_C_D
MUSIC"_G_#D_D_#D_G_#D_D_#D_G_#D_D_#D_G_#D_D_#D
MUSIC"_C_#D_D_#D_#G_#D_D_#D_C_#D_D_#D_#G_#D_D_#D
MUSIC"_D_F_#D_F_#G_F_#D_F_D_F_#D_F_#G_F_#D_F
MUSIC"_#D_G_F_G_#G_G_F_G_#D_G_F_G_#G_G_F_G
MUSIC"_#D_A_G_A_F_A_G_A_#D_A_G_A_F_A_G_A
MUSIC"_D_F_#D_F_#G_F_#D_F_D_F_#D_F_#G_F_#D_F_
MUSIC"_C_F_E_F_#G_F_E_F_C_F_E_F_#G_F_E_F
MUSIC"_C_#D_D_#D_F_#D_D_#D_C_#D_D_#D_F_#D_D_#D
MUSIC"_#GC_BCDC_BC_#GC_BCDC_BC
MUSIC"_A_E_D_E_C_E_D_E_A_E_D_E_C_E_D_E
MUSIC"_G_#D_D_#D_F_#D_D_#D_G_#D_D_#D_F_#D_D_#D
MUSIC"_G_#D_D_#D_C_#D_D_#D_G_#D_D_#D_C_#D_D_#D
MUSIC"_G_#D_D_#D_F_#D_D_#D_G_#D_D_#D_F_#D_D_#D
MUSIC"_G_#D_D_#D_F_#D_D_#D_G_#D_D_#D_F_#D_D_#D
MUSIC"_G_B_D_F_#G_F_E_F_B_FD_B_#G_F_E_F
MUSIC"_A_C_#D_GC_G_#F_GRRRRC_#G_G_#G
MUSIC"_G_A_#FRRRRRRRRRRRR
MUSIC"_G9","_A1DCD#EC_BC_AC_BCD_B_A_B
MUSIC"_G_B_A_BC_#G_G_#G_F_A_G_A_B_G_F_G
MUSIC"_#DGFG#GF#DFDF#DFG#DD#DC#DD#DFDCD#DC_BCD_B_A_B
MUSIC"C#DD#D_FDCD_#DC_BC_D_B_A_B
MUSIC"_C6R3":TEMP05
MUSIC"R0RRRRRRRRRRRRRR_C6R3R0RRRRRRRRRRRRRR":TEMP06
MUSIC"R1C_B3C8","_C9_C7","_C7_C5_C2_C2_C3_C3_C4_C6_C7_C8_C9_C9
```

TAPE RECORDER ON

DELAY — CUE PIPS — DELAY

music START

BASS PART



Issue 6

'Pascal Copy' Program by D. Willey

Approximately half way through the program the line which reads in the notes:

POKE (CHR(I),4357); should be -

POKE (CHR(Z),4355);

Issue 7

BASIC copy routine - Page 55

Line 50 must be

```
50 POKE$ 10D5,$00:USR($0251):USR($04CE):POKE$ 10D5,$ 80: USR
   ($0282):END
```

SHARP SOFT

Sharpsoft Ltd., 86 90 Paul Street, London EC2A 4NE
Printed by Oldham Press [T.U.], Chatham, Kent.