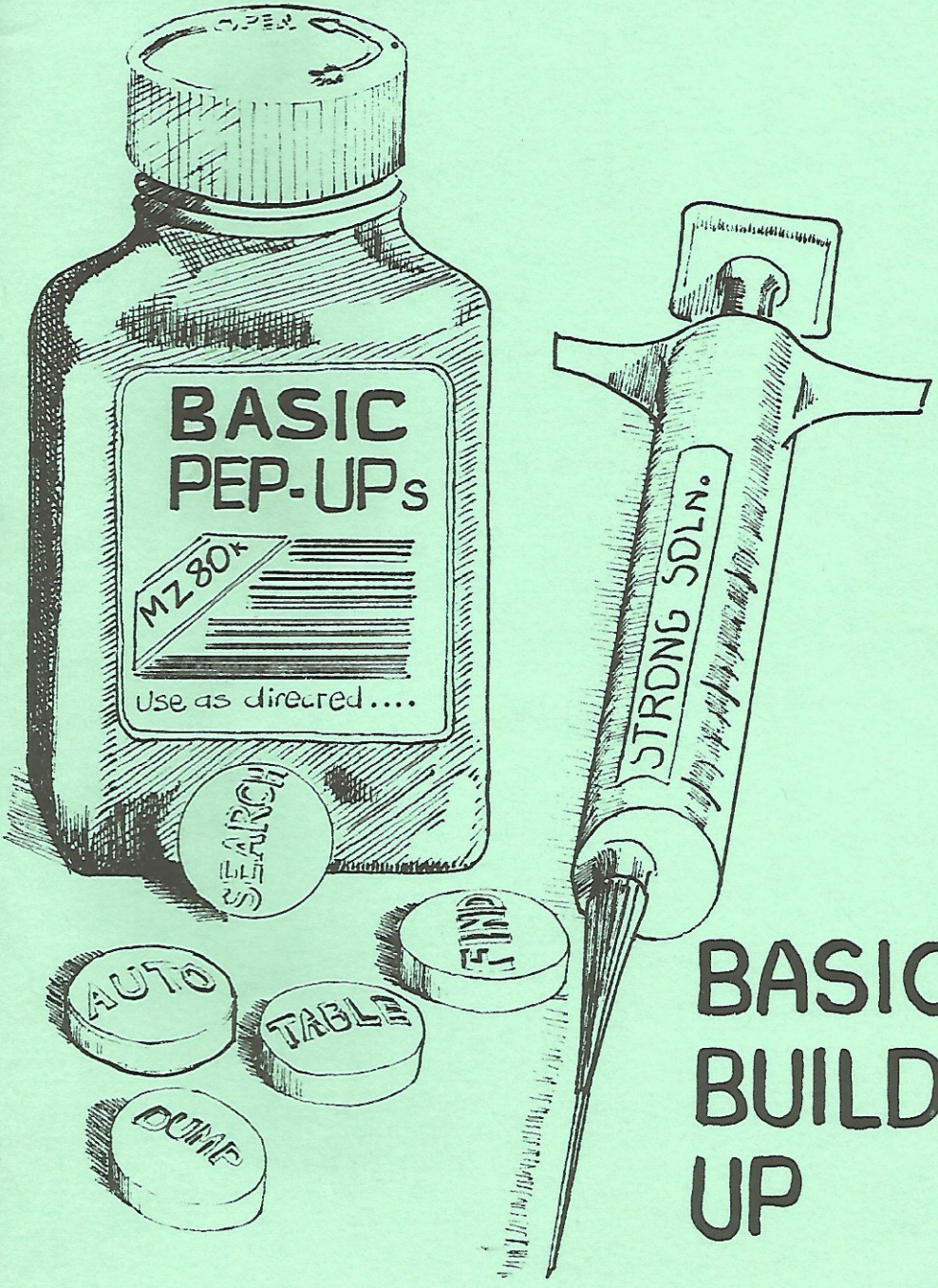
(3)

SHARPSOFT



BASIC BUILD UP

SHARPSOFT USER NOTES

Issue No. 3

This issue of the SHARPSOFT User Notes celebrates the end of our first year of publication. This summer has been an important period for MZ-80 users. Sharp have released in the UK their upgraded version of the MZ-80K, which is called the MZ-80B. In a future issue of the User Notes we will include information on this new machine. However, other news which is probably of more importance to our readers is the release of PASCAL and FORTH for cassette based MZ-80K computers. SHARP have also released a disc based double precision version of BASIC. This issue included introductory articles on each of these new packages.

By now you will have noticed our new front cover and the other illustrations. These have been drawn by one of our readers, John Trippick; adding a "touch of class" to the magazine!

Once again we include your letters, hints and comments. Our major article this issue is concerned with extensions to BASIC and BASIC "Toolkits". This article includes a number of diagnostic routines for use with BASIC SP-5025 and SP-6015.

From your feedback it appears that software, which you can run on the MZ-80K, is an aspect of these User Notes that readers find helpful and stimulating. Last issue we introduced "Tiny Pilot"; P.L. Birch wrote to us with his comments and suggested changes to this package. Mr Birch's comments are presented in the letter section of this issue.

Our thanks to all those readers who have donated their ideas and software. If you have hints and/or software you would like to share with other MZ-80K owners drop us a line at SHARPSOFT.

The first issue of the 1982 User Notes is scheduled for publication next January. Subscription renewal details are included with the article on FORTH. Next year we will be featuring topics on both PASCAL and FORTH. A series on the use of Z80 assemblers on the MZ-80K will, we hope, begin in the January issue. We will also be including all your comments and letters. ED.

Verbose
disk
FORTH
MZ80K
TINY C.

FIG-FORTH FOR CASSETTE BASED MZ-80K COMPUTERS

In the last year the language FORTH has become available for most medium size popular microcomputers. At SHARPSOFT we have developed a cassette based version of FORTH for MZ-80K computers with 48K of RAM. This is now ready for distribution. Our FORTH is based on the 8080 fig-FORTH 1.1 with an integrated cassette virtual memory system replacing the original CP/M disk storage routines. The SHARPSOFT package consists of two language tapes and instructions for loading and running FORTH on the MZ-80K. This package will be on sale in October at a cost of £15.

HOWEVER – Please note

For our subscribers to the SHARPSOFT USER NOTES we have decided to include a copy of FORTH with the first issue of the 1982 user notes. To cover the cost of the FORTH tapes and the printed user instructions the 1982 subscription for the SHARPSOFT USER NOTES will be increased from £3 to £7.50 (in the U.K.) and from £6 to £12 for overseas subscribers. We hope that our readers will agree with us that the increase is small considering our software policy. To help us estimate the demand for the 1982 User Notes – PLEASE renew your subscription as soon as possible.

Included in the FORTH package is an editor, for program preparation, and an 8080 assembler. The editor and assembler are on the second languge tape and can be loaded into the MZ-80K by the FORTH operating system, when required.

We must *stress* that this package is *NOT* a teach your-self guide to FORTH but a working language system. Readers who have no experience of FORTH but who wish to learn FORTH should consult the popular computing magazines or a FORTH text book.

FORTH is an unusual computer language, originally developed over ten years ago by an American, Charles Moore, as an alternative to existing high-level languages. Its early uses included operating systems for the control of Radio Telescopes. Today, FORTH has developed into a powerful and easily implemented high-level language with a rapidly growing group of hobbyist enthusiasts both in this country and America. Although FORTH was developed for control applications, its vocabulary can easily be expanded to include additional program structures, for example, PASCAL-like CASE statements, and data types. These extensions allow a wide range of different problems to be solved using FORTH. Moreover, FORTH allows programmers to develop their own special vocabularies and allows easy access to all the software and hardware components of the computer being programmed.

In the next issue of the USER NOTES we are putting together lots of SAMPLE PROGRAMS for you to put up and test your new version of FORTH. We also intend to organise lots of information on the language from

the American and UK FORTH Interest Groups, so this next issue should really be a bumper start to 1982.

PASCAL for the MZ-80K

Readers who regularly buy the popular computing magazines will have noted that SHARP have recently launched a tape based PASCAL for the MZ-80K.

We believe that this package has been on sale in Japan for some time and hence should be well tested and reasonably bug free. The original version would, of course, have been in Japanese and must have been rewritten for the U.K. market. SHARP call their PASCAL an interpreter which is unusual because PASCAL is normally implemented as a compiled or semi-compiled language. We suspect that SHARP really mean that the package is a "p" code interpreted form of PASCAL.

SHARP PASCAL is a healthy subset of standard PASCAL with a number of the more advanced or esoteric features of the language omitted. The PASCAL package is certainly a welcome addition to the MZ-80K range of language software. The subset includes the floating point data type and a similar range of predefined functions to the SHARP BASIC. Hence, this version of PASCAL is considerably more powerful, and useful, than the "Tiny" PASCAL's which are available on a number of the popular computers.

The SHARP PASCAL package consists of a PASCAL language tape, which is loaded from the monitor, a second tape with a number of example programs and a user manual. The user manual is similar in style to the MZ-80K BASIC manual but includes more background information.

The PASCAL language tape includes a powerful text editor. This is used for program preparation. The editor is a significant improvement to that provided with the BASIC interpreter. Running the example programs leaves one with the impression that the SHARP PASCAL is faster than their BASIC. However, a series of comparative bench marks need to be written and tested to confirm this point.

The major differences between SHARP's version of PASCAL and standard PASCAL are

1. No procedure or functions can be declared within another procedure or function declaration.
2. Structured data types are not supported.
3. Only value parameters can be used.

To run PASCAL your MZ-80K must have more than 32K bytes of RAM fitted.

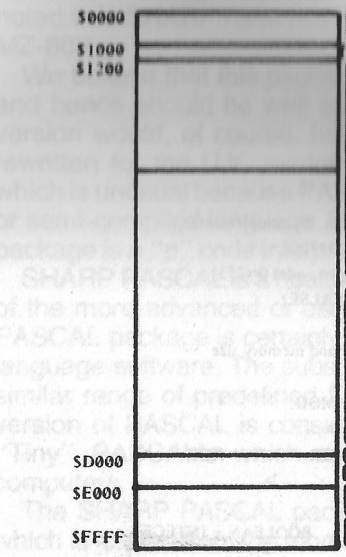
As a special offer to our user notes subscribers SHARPSOFT is selling the SHARP PASCAL package at the reduced price of £45. This figure includes V.A.T. and postage.

PASCAL SP-4015 specifications

System	Cassette tape base interpreter; starts immediately after loading
Size	Approx. 16K bytes
Required RAM capacity	More than 32K bytes
Cold start address	$1200
Warm start address	$1202
Error messages	45 messages
Data types	INTEGER
	REAL
	CHAR
	BOOLEAN
Data range	
INTEGER data	−32767 ~ +32767 (2 byte data, 2's complement)
REAL data	±0.27105055E−19 ~ ±0.92233720E+19
CHAR data	One character (corresponding to codes 0~255)
BOOLEAN data	TRUE and FALSE (TRUE>FALSE)
Number of array dimensions	Up to n dimensions
Range of array index	Varies according to data type and memory size
Identifier length	Up to 32 characters
Integer operators	*, DIV, MOD, +, −
Real operators	*, /, +, −
Logical operators	NOT, AND, OR, XOR
Relational operators	=, <>, <=, >=, <, >
Standard functions	
ODD	Checking whether odd or even. BOOLEAN ← INTEGER
CHR	CHAR ← INTEGER
ORD	INTEGER ← CHAR
PRED	Preceding character
SUCC	Following character
TRUNC	INTEGER ← REAL
FLOAT	REAL ← INTEGER
ABS	Absolute value of INTEGER or REAL data
SQRT	Square root
SIN	sinX
COS	cosX
TAN	tanX
ARCTAN	$\tan^{-1}X$
EXP	e^x
LN	$\log_e X$
LOG	$\log_{10} X$
RND	Random number
PEEK	Read-out from memory
CIN	Read in character at cursor position
INPUT	Read in from port
KEY	Read in from keyboard
RECTR	Receive one byte of data from the color control system

Memory map

The memory map is as shown below when the PASCAL interpreter is loaded in a system with 48K bytes of RAM



(Hexadecimal address)

Statements

Assignment statement	Variable: = < expression >
Compound statement	BEGIN < statement 1 >, < statement 2 >, . . . , < statement n > END;
IF statement	Conditional statement (including ELSE)
CASE statement	Selective execution
WHILE statement	Repetition
REPEAT statement	Repetition
FOR statement	Repetition (including either TO or DOWNTO)
WRITE statement	Data output
READ statement	Data input
CALL statement	User subroutine call
COUT statement	Outputs a character to the cursor position
POKE statement	Writes data into memory
OUTPUT statement	Outputs data to the specified port
MUSIC statement	Plays music (used with the TEMPO statement).
TEMPO statement	Specifies the tempo.
TRAN statement	Transfers a graphic command to the color terminal.
SYRET statement	Resets the color terminal and cold starts the system.
SYRET2 statement	Resets the color terminal and waits for a monitor command.

Others

(1) ..... A file identifier can include a maximum of 16 significant characters.
(2) ..... Statement numbers are automatically assigned by the system.
(3) ..... Recursive call capability.

Differences between the SP-4015 and standard PASCAL

1. No procedure or function can be declared within another procedure or function declaration.
2. Structured type data cannot be used.
3. Only value parameters can be used.

In future additions of these user notes we will be including a regular series on PASCAL. If you are using the SHARP PASCAL and have written any trial programs, or have found any bugs or have any comments to make about this new software write to us and we will publish a selection of your points.

The following books are suitable for those readers who wish to learn more about the fundamentals of the PASCAL language.

Wilson, I.R., Addyman, A.M., *A Practical Introduction to PASCAL*, The Macmillan Press Ltd., 1978.

Grogono, P., *Programming in PASCAL* (Revised edition), Addison-Wesley Publishing Company Inc, 1980.

Jensen, K., Winth, H., *PASCAL User Manual and Report*, Springen-Verlag, 1978

A specification summary of the SHARP PASCAL SP-4015 interpreter is given below.

Eight queens:– an example PASCAL program.

The following sample program arranges 8 queens on a chessboard so that no queen can take any other. There are 92 solutions. This program is an example of recursive programming. NOTE PROCEDURE ARYWRITE calls itself.

```
0.% EIGHT QUEENS %
1.VAR A,X:ARRAY[7]OF INTEGER;
2.    B,C:ARRAY[14]OF INTEGER;
3.    D,P:INTEGER;
4.PROCEDURE CLEAR;
5.  VAR N:INTEGER;
6.  BEGIN
7.    FOR N:=0 TO 7 DO A[N]:=1;
8.    FOR N:=0 TO 14 DO B[N]:=1;
9.    FOR N:=0 TO 14 DO C[N]:=1;
10.   P:=0
11. END;
12.PROCEDURE ARYWRITE;
13.  VAR Z:INTEGER;
14.  BEGIN
15.    Z:=0;
16.    REPEAT
17.      IF(A[Z]+B[P-Z+7]+C[P+Z])=3)THEN
18.        BEGIN
19.          X[P]:=Z;
20.          A[Z]:=0;
21.          B[P-Z+7]:=0;
22.          C[P+Z]:=0;
23.          P:=P+1;
24.          IF P=8 THEN DATAOUT
25.            ELSE ARYWRITE;%.....................RECURSIVE CALL %
26.          P:=P-1;
27.          A[Z]:=1;
28.          B[P-Z+7]:=1;
29.          C[P+Z]:=1
30.        END;
31.      Z:=Z+1
32.    UNTIL Z=8
33.  END;
34.PROCEDURE WAKU;
35.  VAR E,M,N:INTEGER;
36.  BEGIN
37.    D:=0;
38.    WRITELN("■■■      ** EIGHT QUEENS **■■■");
39.    WRITE("     ┌");
40.    FOR M:=0 TO 6 DO WRITE("─┬");
41.    WRITELN("─┐");
42.    FOR E:=0 TO 7 DO
43.      BEGIN
44.        WRITE("     |");
45.        FOR M:=0 TO 7 DO WRITE(" |");
46.        WRITELN();
47.        IF E<>7 THEN
48.          BEGIN
49.            WRITE("     ├");
50.            FOR M:=0 TO 6 DO WRITE("─┼");
51.            WRITE("─┤");
52.            WRITELN()
53.          END
54.        ELSE BEGIN WRITE("     └");
55.               FOR M:=0 TO 6 DO WRITE("─┴");
56.               WRITELN("─┘")
57.             END
58.    END
59.  END;
60.PROCEDURE DATAOUT;
61.  VAR F,M,N,Z:INTEGER;
62.  BEGIN
```

```
63.     D:=D+1;
64.     WRITE("■■■■■■■■");
65.     WRITE(D:2);
66.     WRITELN();WRITELN();WRITELN();%.............. 3 CARRIAGE RETURNS %
67.     FOR N:=0 TO 7 DO
68.       BEGIN
69.         WRITE("     ■");
70.         FOR M:=0 TO 7 DO
71.           BEGIN
72.             IF X[N]=M THEN
73.               BEGIN
74.                 F:=M+1;
75.                 WRITE("Q■");
76.                 CALL(62)Z,Z;%.......... BELL (Z IS A DUMMY VARIABLE.) %
77.               END
78.             ELSE WRITE(" ■")
79.           END;
80.         WRITELN("■",F:2,"■")
81.       END
82.   END;
83.BEGIN % ..................................................... EIGHT QUEEN MAIN %
84.   CLEAR;
85.   WAKU;
86.   ARYWRITE
87.END.
88.
```

Double precision BASIC for the MZ-80K

In some applications the accuracy of the SHARP BASIC SP-5025 or SP-6015 interpreters yield significant rounding errors during complex calculations. To overcome this problem SHARP have released a double precision BASIC interpreter for use with MZ-80K computers fitted with discs.

This new interpreter is an extended version of the existing BASIC which provides a number of new facilities in addition to giving much improved arithmetic accuracy.

Brief summary of SP-6115:–

RANDOM FILES	– 32 byte segments instead of 16 byes.
AUTO	– Auto line numbering option.
APPEND	– A BASIC program stored on disk or tape can be appended to the current program in memory.
DELETE	– Allows for the deletion of blocks of program lines as well as single lines.
DIM	– 1 – dimension arrays can have any number of elements, limited only by memory size. 2 – dimension arrays can have up to 255 elements in each dimension.
TAPE	– Data files are allowed on cassette.
PRINT USING	– PRINT using option allows for formatted output, both on the screen and the printer.
ARITHMETIC	– All arithmetic calculations are carried out in Binary Coded Decimal for increased accuracy. Numeric accuracy range:– +/–1E–48 to +/–9.999999999999999 × 78.

This package now available for £39.10 including V.A.T. Normal Price £46.00.

BASIC extensions and software development tools

In previous issues of the SHARPSOFT user notes we have tried to outline the background to the use of the SHARP BASIC SP-5025 and SP-6015 interpreters. These versions of BASIC are typical modern interpreters for the Z80 microprocessor. SHARP in their publicity literature make the important point that the MZ-80K computer is a "clean" computer with the memory mainly RAM, where the BASIC interpreter is loaded either from tape or disc. In some computers the BASIC interpreter is held permanently in ROM. This form of BASIC is faster to load but does mean that part of the Z80 address memory map is reserved for BASIC and *cannot* be changed. In the past when 8K floating point BASIC interpreters were the norm this was not a serious problem. However, as BASIC interpretes have grown in size a significant proportion of the Z80 address space becomes locked "off" for BASIC. With the MZ-80K, if the computer is to be used for programming in a different language, for example Z80 Assembly language or "Tiny-C", then we only have to re-load the new language from tape or disc. The full memory is also available for use by the new language. Do remember that in the normal Z80 system design the Z80 address map is limited to 64K and it is also not uncommon for BASIC interpreters to be as large as 24K. A second point worth noting is that it is often difficult to change, or add extra facilities to, a BASIC interpreter which is hard wired in ROM. The reverse is true for a RAM based BASIC.

Since the MZ-80K was first introduced by SHARP many programmers have analysed the Sharp versions of BASIC and highlighted their deficiencies.

For most hobbyist programmers, who may be using BASIC for the first time, SP-5025 or SP-6015 offer a comprehensive set of BASIC commands and functions. However, the more experienced BASIC programmer will realize that both these implementations have a number of limitations; for example the lack of program development aids like a RENUMBER or TRACE command. Also when writing longer programs some form of standardiation of the BASIC language is important. The American software house "Microsoft" have set the standard for 8080/Z80 and 6502/6800 microprocessor BASIC's. Their dielect of BASIC has tended in the last five years to be used as the de-factor industrial standard for microprocessor BASIC interpreters and compilers. SHARP BASIC does in general conform to the microsoft syntax but, however, there are a number of important differences. One of the most noticeable is the lack of a full implementation of the AND, OR and NOT logical operators.

To overcome these deficiences, and to extend the interpreters, a number of software houses and other programmers have either re-written sections of the SHARP BASIC or added to the interpreters. This is where the idea of BASIC extension programs or "Tool kits" comes from.

BASIC extensions and "Tool kits" are normally written in machine code and once loaded into the computer they change and/or increase the power of the BASIC interpreter. A second approach is to write useful utility programs in BASIC and to use these utilities to operate on the BASIC programme under development.

Both these approaches are discussed in the following notes.

Extensions to SHARP BASIC SP-5025

These extensions to SP-5025 were written by Dr B.R. Gladman and are available from the major MZ0-80K software distributors.

This package adds both language extensions and a number of utility commands to BASIC SP-5025. An interesting feature of the extensions is that the size of the BASIC interpreter is *NOT* increased leaving the same user program space in RAM after loading the extensions from tape. This is, of course, important for MZ-80K owners with the smaller 20K RAM computer.

The extension package adds the following language commands, functions and utilities to SP-5025:–

BREAK, TRACE AND SINGLE STEP OPERATION

The BREAK command

The command BREAK nn, where nn is any line number, sets a breakpoint in the Basic text immediately before the specified line. Subsequently if this line is encountered when a Basic program is running, execution stops and control returns to the user. The command OFF cancels the breakpoint.

The STEP command

The command STEP places Basic in single step mode after which a running Basic program will stop before the execution of each Basic statement and display its line number on the screen to the right of the current cursor position. Pressing any key except shift-break will cause a single statement to be executed; pressing shift-break will return control to the user after which the command OFF can be used to terminate single step operation. Note that the line number shown will not always change because some Basic lines contain more than one statement.

The TRACE command

The TRACE command is used with BREAK and STEP and causes Basic to keep a record of the line numbers of the last eight statements executed. These line numbers are displayed either when shift-break is pressed in single step mode or when a breakpoint set by BREAK is encountered.

TRACE is turned off by the command OFF – remember to do this for normal operation because TRACE slows the execution of Basic significantly.

The OFF command

The command OFF cancels all BREAK, STEP and TRACE options which are in effect.

BLOCK DELETE, RENUMBER AND LINE NUMBERING COMMANDS

The DELETE command

This command has the format: DELETE mm-nn, where mm and nn are the initial and final line numbers of a block of lines to be deleted. DELETE 100-300 for example will remove all lines with line numbers between 100 and 300 inclusive. As a safety feature both the initial and final line numbers must be explicit, the forms DELETE-100 and DELETE-300 as used in LIST not being allowed.

The RENUMBER command

In its full form this command has the format:

RENUMBER mm-nn/new, inc

where mm and nn are the initial and final line numbers of a block of lines to be renumbered, new is initial new line number, and inc is the step between new numbers. Thus the command RENUMBER 100-300/120,5 will re-number lines between 100 and 300 inclusive in the sequence 120,125,..... The range for renumbering can be in any of the following forms:

mm-nn	from mm to nn inclusive
mm-	from mm to top line number inclusive
-nn	from bottom line number to nn inclusive
mm	a single line
absent	all lines present

Also the new number and increment can appear in any of the forms:

/new, inc	start with 'new', step in units of 'inc'
/new	start with 'new', step in units of 10
/ , inc	start with 1000, step in units of 'inc'
absent	start with 1000, step in units of 10

Thus RENUMBER alone will renumber the whole Basic program in the

sequence 1000, 1010, 1020, ..., while RENUMBER/1,1 will renumber the whole file in the sequence 1,2,3,....

When using RENUMBER on part of a program (i.e. not the whole program) a range error will occur if the range and new numbers specified are such that a change in the order of Basic lines or duplicated line numbers would result. For example, using the program:

```
100 A = 1
100 B = 2
120 C = 3
130 D = 4
```

attempting RENUMBE 110-/100 will fail because the new number for line 110 will be 100 conflicting with a line already present. Similarly the command RENUMBER 100-120/121 will fail because line 120 would be renumbered 131 which would reverse the order of the last two program lines. An error will also occur if the values specified result in line numbers greater than 65535.

The RENUMBER command will detect references to lines which are not present if they lie in the range being renumbered. If such a line is detected an error message is displayed and the reference to the non-existant line is preceeded by a 'U' to indicate that it is undefined.

It should be remembered that the RENUMBER command may increase the length of a Basic program in memory if new line numbers contain more characters than old ones. If memory space is very limited it is therefore possible for a memory full error to occur during renumbering with the result that the program will be left in an undefined state. If there is any danger of this it is advisable to have a copy of the program being renumbered on tape.

The AUTO command

This command has the general format AUTO number, increment and is available to provide automatic line numbers during the entry of Basic programs using the keyboard. AUTO 100,10 for example will provide line numbers in the sequence 100,110,120,.... If 'number' is not present a value of 1000 is supplied and if 'increment' is not present its value is assumed to be 10. Thus AUTO on its own gives the sequence 1000, 1010, 1020, ... Automatic line numbering can be terminated at any time by pressing the shift-break keys.

LOGICAL OPERATIONS

String inequality

Logical expressions involving the inequality of strings are not allowed in

Sharp Basic but have been implemented in the extentions. Thus expressions such as Q$><"YES" are now legal.

AND, OR, and NOT

These logical operators are provided in extended Basic, NOT having the highest priority, followed by AND, and then OR. Thus the expression:

NOT A = 6 OR NOT B = 5 AND C = 3

is equivalent to:

(NOT A = 6) OR (NOT B = 5) AND C = 3)

In Sharp Basic true and false expressions are given values of -1 and 0 respectively. Because they are given numeric values logical expressions can be used in arithmetic statements provided they are enclosed in brackets. If an arithmetic value is used where a logical value is expected it will be treated as true if it is non-zero and false if zero.

NEW FUNCTIONS

SET(X,Y) and RESET(X,Y)

These functions are provided to allow a Basic program to determine whether a point on the screen in high resolution mode is either set or clear. The function SET(X,Y) will give a value of 0 if position X,Y on the screen is clear and a value of -1 if it is occupied by part of a normal character or is set in graphics mode (i.e. it has been set by a SET(X,Y) command). X and Y can be general arithmetic expressions. The function RESET(X,Y) is the opposite of SET(X,Y), giving a value of zero when SET would give -1, and -1 when SET would give zero. Thus the expressions RESET(X,Y) and NOT SET(X,Y) are identical.

USR(X) and USR(X,Y)

These functions are provided to pass values to and from machine code sub-routines. The first parameter X is evaluated and is used as the entry point to the machine code program. In the second form of the function USR(X,Y), the expression Y is evaluated and its value is placed in the BC register before the machine code program is entered (Y must be in the range 0 to 65535). Both USR(X) and USR(X,Y) are functions which are used in arithmetic expressions and have the value which is left in the BC register on exit from the machine code subroutine. For example, if the

machine code program at 41000 in memory leaves a value of 12 in BC on exit, the Basic statement:

```
A = 100*USR(41000) + 25
```

will set the variable to 1225. Similarly if the machine code program adds the BC register to itself the expresion:

```
A = USR(41000,300) + 2000
```

sets A to 2600.

CURSOR CONTROL IN PRINT STATEMENTS

Cursor control is implemented through a new PRINT statement which has the form:

```
PRINT OZ,Y:"YES"
```

which will print YES at position X,Y on the screen, X bring the line number (0-23) and Y being the position on the line (0-39). The semi-colon after OX,Y must be present. Cursor control can occur more thanonce in a PRINT statement as in:

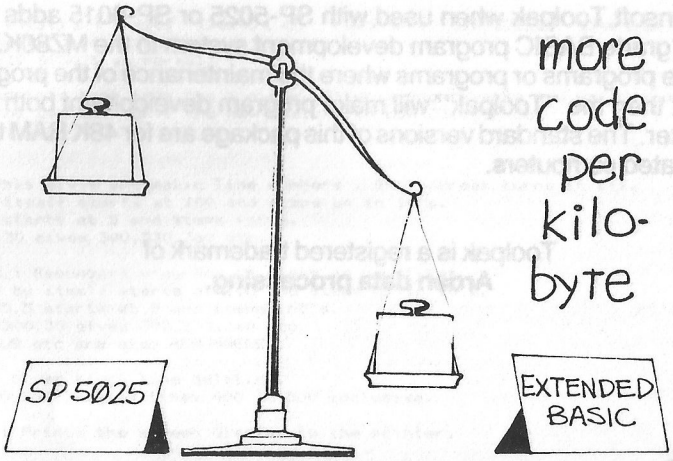
```
PRINT O10,10;X;O12,20;Y;O15,30;Z
```

which will print the values of the variables, X, Y, and Z at the specified screen positions. The two values which give the screen position (i.e. X and Y in OX,Y) can be general arithmetic expressions. If only one expression occurs after O it is used to set the position n the current line. Thus:

```
PRINT O12,I;"YES";OI+5;"NO";OI+10;"MAYBE"
```

with I equal to 20 will print YES, NO and MAYBE at positions 20, 25 and 30 on line 12. The use of O for cursor control is not compatible with the use of the comma in print statements for tabulating results.

We have used this package for program development and find that it adds important facilities for BASIC SP-5025. It will be especially useful to those MZ-80K owners with the smaller 20K machine. The RENUMBER and DELETE commands are very powerful making program development faster and easier. Those READERS who write machine code segments for use with BASIC will like the extended USR functions. The ability to pass parameters as arguments to the BASIC USR function considerably extends the power of the SP-5025 interpreter.



ARDENSOFT Toolpack

The software house Ardensoft distribute an extension package for the SHARP BASIC SP-5025 and SP-6015. This package provides programmers with a set of program development utilities rather than changes to the BASIC language.

Three major modifications for the BASIC operating environment are included in the package, these are:

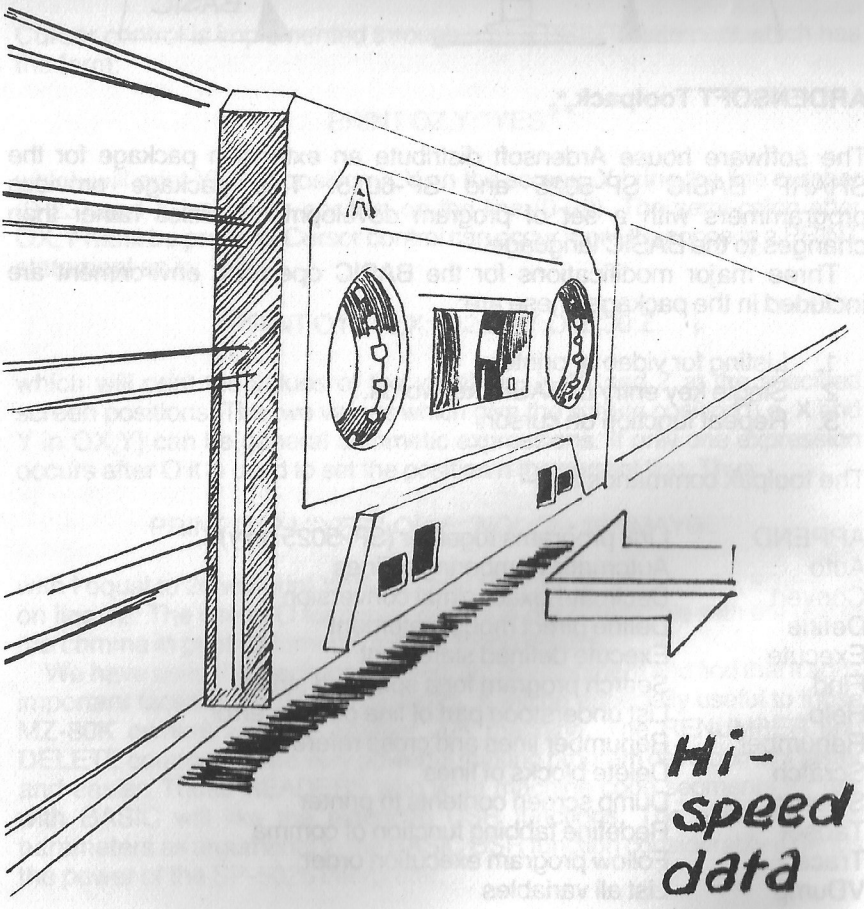
1. Listing for video or printer.
2. Single key entry of BASIC keywords.
3. Repeat function on cursor.

The toolpak commands are:–

APPEND	Link programs together (SP-5025 only)
Auto	Automatic numbering of lines
Convert	Decimal/hexadecimal conversion
Define	Define direct mode statement
Execute	Execute defined statement
Find	Search program for a specified string
Help	List understood part of line offer an error.
Renumber	Renumber lines and cross references
Scratch	Delete blocks of lines
SDump	Dump screen contents to printer
Tabset	Redefine tabbing function of comma
Trace	Follow program execution order
VDump	List all variables

The Ardensoft Toolpak when used with SP-5025 or SP-6015 adds a professional grade BASIC program development system to the MZ80K. If you write large programs or programs where the maintenance of the program is important then the "Toolpak" will make program development both easier and quicker. The standard versions of this package are for 48K RAM tape or disc operated computers.

Toolpak is a registered trademark of
Arden data processing.



Modified Basic (With SPEED Data)

This tape is used as follows-
Load BASIC SP5025 as normal
Enter BYE CR
Then load this tape from the monitor .After a short pause you will re-enter Basic. You can then SAVE the new basic with USR(33):USR(36)

AUTO : This gives automatic line numbers . Shift/Break turns it off.
AUTO by itself starts at 100 and steps up in 10's.
AUTO5,5 starts at 5 and steps in5's.
AUTO300,30 gives 300,330,360 etc.

RENUMBER : Renumbers programs in steps as you like.
RENUMBER by itself starts at 100 and steps up in 10's.
RENUMBER5,5 starts at 5 and steps in5's.
RENUMBER300,30 gives 300,330,360 etc.
GOTO GOSUB etc are also RENUMBERED.

DELETE : Gives block line deletion.
DELETE400,500 Deletes lines 400 TO 500 inclusive.

PRINT/S : Prints the screen display to the printer.

PRINT/A : Directs all PRINT and PRINT/P statements to both screen and printer.

PRINT/N : Directs all PRINT and PRINT/P statements to screen only.

PRINT/O : Restores to normal.

PRINT/B : Alters the comma auto spacing to the printer. Instead of the normally fixed 10 it can be set from 1 TO 255.
PRINT/B24 Sets this to 24. You can also use variables as the no.

APPEND Joins Basic programs together.
Load a program from tape and then APPEND a program that is higher numbered. Do not do anything between LOAD and APPEND!

SLOWn Is a trace function.
SLOW slows down program execution and displays the current line number in the top left hand corner. The value of n sets the delay and can be from 1 to 255. 1 is the fastest.

NOTE : Use of the bottom right graphic key will stop and restart Program execution,Listing and Dump

OFF Turns off SLOW.

DUMPn Lists the values of variables. n can be fom 1 to 7 :
DUMP1 Lists all user defined functions
DUMP2 Lists all simple string variables
DUMP3 Lists all single dimension string arrays
DUMP4 Lists all two dimensional string arrays
DUMP5 Lists all simple numeric variables
DUMP6 Lists all single dimension numeric arrays
DUMP7 Lists all two dimensional numeric arrays
DUMP0 Does all the above

```
When a two dimensional array is being dumped the first index changes most
rapidly and after each row the second index is printed in brackets.

PRINT@X,Y; Is a print at command.
X and Y are 0-39 and 0-24 respectively. eg.?@20,12;'K' prints K in the centre of
the screen.

Full string inequalities are now legal
ie IF Z$< '33445' THEN PRINT 'Its smaller !!!'

Logical AND & OR are also implemented .
ie IF X<5 AND Y<6 THEN GOTO 100

It may on occasion pay to bracket the functions for clarity.
AND & OR are also legal in arithmetic expressions provided they are bracketed

SPEED Data uses a new technique to save data to tape which is compatible with
normal basic. No commands have to be given  as it is implemented automaticaly.
Data recorded with this system loads in 1/3 of the normal time
```

SPEED BASIC

Modified Basic (With SPEED Data)

SPEED BASIC is a tool kit that also incorporates an improved data read system. The fast data read only takes a ⅓ (one third) of the time to load that standard SP5025 BASIC takes to load. It is totally compatable with DATA from 5025 BASIC so you can transfer data written in 5025 and run it on SPEED BASIC without any modification at all. The other commands so far as the tool kit go are as follows.

SPEED BASIC is an "overlay" of SHARP 5025© BASIC and retails for £11.50 including V.A.T.

SPEED BASIC along with the other tool kits or extensions to BASIC are available from most MZ-80K software stockists.

Utility functions written in BASIC

Although machine code utility programs are faster than the same functions coded in BASIC it is still viable to use BASIC as a means to diagnose a program under development.

The techniques for writing BASIC utility programs are easily understood if one remembers, and understands, the way in which BASIC "Tokenizes" and stores BASIC program text in RAM. This we outlined in the first issue of the SHARPSOFT user notes.

To use a BASIC utility program the following procedure is adopted. First a useful utility program is written in BASIC. Normally high line numbers are used, say starting with line 60000. Once this program is working it is loaded into the MZ-80K BEFORE the program under development is typed in. As the new piece of software is developed copies are saved on tape or disc *with* the utility program. To use the utility we use the direct mode command GOTO 60000. The utility then operates, displaying on the V.D.U. and or printer, diagnostic information on the program stored in memory with line numbers less than 60000.

It is a straightforward task to write BASIC RENUMBER, SEARCH, TABLE and HEX DUMP utilities. Different versions of these programs for disc and tape based computers with or without printers are given on the next few pages.

Each of these programs was written when we were testing SHARP BASIC. They are as far as we can tell "bug-free". However, if you find a bug then let us know and we will attempt to correct it.

A useful extension to these utilities would be on PACK/UNPACK program. Often when developing a program it is easier to enter one BASIC statement per line. This does however, waste memory space and slows down the operation of a program. Hence normally on completion of a program, programmers pack more than one BASIC statement per line and remove REM statements. The process of PACKing (or UNPACKing) a BASIC utility program can be done by a BASIC utility program. Perhaps some of our readers would like to try writing one? Send your solutions to us and we will publish the best three in a later issue!

MZ-80K – UTILITY PROGRAMS – DUMP UTILITY

(For use with SHARP DISK BASIC SP-6016 and SHARP CASSETTE BASIC SP-5025)

These utility programs are aids designed to help programmers write machine code segments for use with disk and cassette versions of SHARP BASIC. The dump utility programme can be used to display a page (265 bytes) of machine code in hexadecimal format. The dump utility program shuld be loaded into the MZ 80K memory before typing in your program. Also it is important to remember that the Dump utility uses BASIC line numbers 60000 to 60270, consequently the program being developed MUST NOT contain line numbers within this range. If you do not have a printer or are using the cassette version of the SHARP BASIC make changes shown in the program listing. To use the dump utility program enter the direct mode COMMAND RUN 60000, followed by a carriage return. The computer requests the user to enter the starting address (in hexadecimal) of the page to be displayed.

RUN 60000 ⟨CR⟩

⟨CR⟩ denote carriage return pressed

HEX DUMP ADDRESS

0000
Δ
typed in from keyboard

Dump utility

```
60000 REM Dump utility
60010 REM For SHARP cassette BASIC
60020 REM change line number 60050
60027 REM to POKE 10167,1
60030 PRINT"&";H$="0123456789ABCDEF":K1=4096:K2=256:K3=16
60040 DATA 4096,256,16,1
60045 DIMA(4):U=48:X=7:Y=9
60050 POKE 3048,1
60060 INPUT"      HEX DUMP ADDRESS    ";H$:IFLEN(A$)<4THEN60060
60070 A$=LEFT$(A$,3)+"0":GOSUB60230:PRINT:PRINT
60080 PRINT#P:PRINT#P:PRINT#P"■"
60090 PRINT"HEX   0 1 2 3 4 5 6 7 8 9 A B C E F":PRINT
60100 PRINT#P"HEX   0 1 2 3 4 5 6 7 8 9 A B C D E F":PRINT#P
60110 AH=A
60120 FORI=1TO16:A=AH:GOSUB60190:PRINTA$+" ";
60130 PRINT#PA$+" ";
60140 FORJ=0TO15:A=AH+J:A=PEEK(A):GOSUB60190:PRINTRIGHT$(A$,2);
60150 PRINT#PRIGHT$(A$,2);
60160 NEXTJ:AH=AH+J:PRINT
60170 PRINT#P:PRINT#P:PRINT#P
60180 NEXTI:END
60190 RESTORE:A$="":M=4
60200 READP:FORZ=1TO16:IFA-Z*P<0THEN60210
60205 NEXTZ
60210 A$=A$+MID$(H$,Z,1):A=A-(Z-1)*P:M=M-1:IFM>0THEN60200
60220 RETURN
60230 FORZ=1TO4:A(Z)=0:NEXTZ:FORZ=1TO4
60240 A(Z)=ASC(MID$(A$,Z,1))-U:IFA(Z)>YTHENA(Z)=A(Z)-X
60250 NEXTZ:A=K1*A(1)+K2*A(2)+K3*A(3)+A(4):RETURN
60260 REM ■■■■■■■■ IF NO PRINTER DELETE LINES
60270 REM 60080,60100,60130,60150 and 60170
```

MZ 80K – UTILITY PROGRAM – TABLE

A TABLE program for Sharp Cassette Basic SP 5025
A TABLE program for Sharp Cassette Basic SP 5025 with printer
A TABLE program for Sharp Disc Basic SP 6015
A TABLE program for Sharp Disc Basic SP 6015 with printer

The Basic utility programs TABLE are aids designed to help programmers to write SHARP CASSETTE or DISC BASIC programs. These programs can be used to search through a program to locate and list variable names and line numbers where the variables occur in the program.

NOTE – TABLE should be loaded into the MZ 80K memory before typing in your program. Also it is important to remember that these Table programs use BASIC line numbers 60000 to 61150, consequently the program being developed must NOT contain line numbers within this range.

To use TABLE enter the direct mode COMMAND RUN 60000, followed by carriage return. The computer then requests the user to select the line number range for the table search. The following example will help users understand how to operate these utilities:–

Example program

```
 10  FOR I=1TO10                          user "test: program
 20  B=COS (I)                              (remember load
 40  PRINTI;Y;Y;B                             TABLE first)
 50  NEXT I
 60  BS="TEST"
 65  DIM DU(10),CS(10)
 70  FOR A=1TO10
 80  DU(A)=A:CS(A)=BS:TS=TS+CS(A)
 90  PRINT DU(A);TS
100  NEXT A
200  END
```

RUN 60000 (CR)

Direct mode command (CR) to operate table program

(CR) denotes carriage return pressed

FROM ? ALL (CR) user requires all lines to be searched

**** Table ****

Line numbers processed

10 20 30 40 50 60 65 70 80 90 100

MZ 80K – UTILITIES PROGRAMS – TABLE

Variables-----Line numbers

```
I     10  20  30  50
Y     20
B     30
I;    40
BS    60  80
DU    65  80  90
CS    65  80
A     70  80  90  100
TS    80  90
```

RUN 60000 (CR)

FROM ? 30 (CR) Lines 30 to 60
TO ? 60 (CR) search

**** Table/TP ****

Line numbers processed
30 40 50

Variables---Line numbers

```
B     30
I     30  50
I;    40
```

RUN 60000 (CR)

FROM ? 40 (CR) lines 40 to "END"
TO ? END (CR) of program searched

**** Table/TP ****

Line numbers processed
40 50 60 65 70 80 90 100 200

MZ 80K – UTILITY PROGRAMS – TABLE

Variables---Line numbers

```
I;    40
I     50
BS    60  80
```

```
DU  65 80 90
CS  65 80
A   70 80 90 100
TS  80 90
```

LIMITATIONS

ONLY 256 different variable names may be searched for in each table search. If your program contains more than 256 variables search the complete program in sections.

Utility Table/T

```
60000 REM
60830 PRINT"♦♦♦♦♦♦   Table/T    ♦♦♦♦♦":PRINT
60840 POKE10167,1:DIMVS$(255):A=18438:C=0:LP=0:V=60000
60841 FORI=1TO100:VS$(I)="":NEXTI:GOSUB61100
60842 PRINT:PRINT:PRINT"Line numbers processed":PRINT:PRINT
60850 D=PEEK(A)+256*PEEK(A+1)
60860 LN$=STR$(PEEK(A+2)+PEEK(A+3)*256):A=A+3
60870 IFVAL(LN$)<>VGOTO60877
60871 PRINT:FORI=1TO39:PRINT"-";:NEXTI:PRINT
60872 PRINT:PRINT"Variables---Line numbers":PRINT:GOTO61010
60877 IFLP>10THENPRINT:LP=0:GOTO60880
60878 PRINT" ";LN$;
60879 LP=LP+1
60880 A$="":A=A+1:IFA>=DTHEN60850
60890 IFPEEK(A)=34THEN61040
60900 IF(PEEK(A)=129)+(PEEK(A)=128)THENA=D:GOTO60850
60910 IF(PEEK(A)>90)+(PEEK(A)<65)THEN60880
60920 A$=A$+CHR$(PEEK(A)):A=A+1:IFA=DTHEN60960
60930 P=PEEK(A):IF(P>47)*(P<91)*(P<>61)*(P<>58)THEN60920
60940 IFPEEK(A)=36THENA$=A$+"$"
60945 IFLEN(A$)<3THENA$=A$+" ":GOTO60945
60950 Z$=MID$(A$,3,1):Z=ASC(Z$):IFZ<>36THENA$=LEFT$(A$,2)
60960 IFLEN(A$)<3THENA$=A$+" ":GOTO60960
60970 A$=LEFT$(A$,3):IFA$="REM"THENA=D:GOTO60850
60990 FORX=1TOC:IFLEFT$(VS$(X),3)=A$THEN61060
61000 NEXTX:C=C+1:VS$(C)=A$+LN$:GOTO60880
61010 FORX=1TOC:PRINTVS$(X)
61020 NEXTX:END
61040 A=A+1:IF(PEEK(A)<>34)*(A<>D-1)THEN61040
61050 GOTO60910
61060 IFLN$=RIGHT$(VS$(X),LEN(LN$))THEN60880
61070 VS$(X)=VS$(X)+" "+LN$:GOTO60880
61100 INPUT"FROM ? ";LN$:IFASC(LN$)=65THENV=60000:RETURN
61110 X=VAL(LN$):INPUT"TO ? ";LN$:IFASC(LN$)=69THENV=60000:GOTO61130
61120 Y=VAL(LN$)
61130 LN$=STR$(PEEK(A+2)+PEEK(A+3)*256):IFVAL(LN$)=XTHENRETURN
61140 IFVAL(LN$)=YTHENPRINT"ERROR FROM = ";X;"TO = ";Y:END
61150 A=PEEK(A)+256*PEEK(A+1):GOTO61130
```

Utility Table/TP

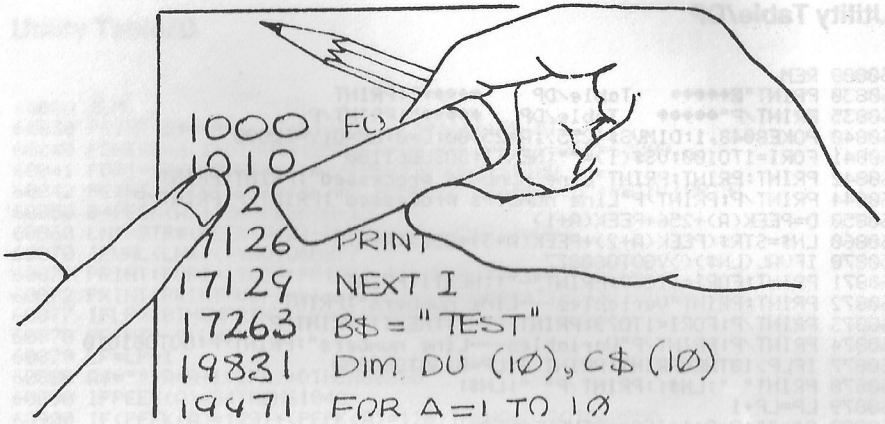
```
60000 REM
60830 PRINT"■♦♦♦♦♦   Table/TP    ♦♦♦♦♦":PRINT
60835 PRINT/P"♦♦♦♦♦   Table/TP    ♦♦♦♦♦":PRINT/P
60840 POKE10167,1:DIMVS$(255):A=18438:C=0:LP=0:Y=60000
60841 FORI=1TO100:VS$(I)="":NEXTI:GOSUB61100
60842 PRINT:PRINT:PRINT"Line numbers processed":PRINT:PRINT
60844 PRINT/P:PRINT/P"Line numbers processed":PRINT/P:PRINT/P
60850 D=PEEK(A)+256*PEEK(A+1)
60860 LN$=STR$(PEEK(A+2)+PEEK(A+3)*256):A=A+3
60870 IFVAL(LN$)<>YGOTO60877
60871 PRINT:FORI=1TO39:PRINT"-";:NEXTI:PRINT
60872 PRINT:PRINT"Variables---Line numbers":PRINT
60873 PRINT/P:FORI=1TO79:PRINT/P"-";:NEXTI:PRINT/P
60874 PRINT/P:PRINT/P"Variables---Line numbers":PRINT/P:GOTO61010
60877 IFLP>10THENPRINT:PRINT/P:LP=0:GOTO60880
60878 PRINT" ";LN$;:PRINT/P" ";LN$;
60879 LP=LP+1
60880 A$="":A=A+1:IFA>=DTHEN60850
60890 IFPEEK(A)=34THEN61040
60900 IF(PEEK(A)=129)+(PEEK(A)=128)THENA=D:GOTO60850
60910 IF(PEEK(A)>90)+(PEEK(A)<65)THEN60880
60920 A$=A$+CHR$(PEEK(A)):A=A+1:IFA=DTHEN60960
60930 P=PEEK(A):IF(P>47)*(P<91)*(P<>61)*(P<>58)THEN60920
60940 IFPEEK(A)=36THENA$=A$+"$"
60945 IFLEN(A$)<3THENA$=A$+" ":GOTO60945
60950 Z$=MID$(A$,3,1):Z=ASC(Z$):IFZ<>36THENA$=LEFT$(A$,2)
60960 IFLEN(A$)<3THENA$=A$+" ":GOTO60960
60970 A$=LEFT$(A$,3):IFA$="REM"THENA=D:GOTO60850
60990 FORX=1TOC:IFLEFT$(VS$(X),3)=A$THEN61060
61000 NEXTX:C=C+1:VS$(C)=A$+LN$:GOTO60880
61010 FORX=1TOC:PRINTVS$(X)
61020 PRINT/PVS$(X):NEXTX:END
61040 A=A+1:IF(PEEK(A)<>34)*(A<>D-1)THEN61040
61050 GOTO60910
61060 IFLN$=RIGHT$(VS$(X),LEN(LN$))THEN60880
61070 VS$(X)=VS$(X)+" "+LN$:GOTO60880
61100 INPUT"FROM ? ";LN$:IFASC(LN$)=65THENY=60000:RETURN
61110 X=VAL(LN$):INPUT"TO ? ";LN$:IFASC(LN$)=69THENY=60000:GOTO61130
61120 Y=VAL(LN$)
61130 LN$=STR$(PEEK(A+2)+PEEK(A+3)*256):IFVAL(LN$)=XTHENRETURN
61140 IFVAL(LN$)=YTHENPRINT"ERROR FROM = ";X;"TO = ";Y:END
61150 A=PEEK(A)+256*PEEK(A+1):GOTO61130
```

Utility Table/D

```
60000 REM
60830 PRINT"■♦♦♦♦♦    Table/D     ♦♦♦♦♦":PRINT
60840 POKE8048,1:DIMVS$(255):A=25900:C=0:LP=0:V=60000
60841 FORI=1TO100:VS$(I)="":NEXTI:GOSUB61100
60842 PRINT:PRINT:PRINT"Line numbers processed":PRINT:PRINT
60850 D=PEEK(A)+256*PEEK(A+1)
60860 LN$=STR$(PEEK(A+2)+PEEK(A+3)*256):A=A+3
60870 IFVAL(LN$)<>VGOTO60877
60871 PRINT:FORI=1TO39:PRINT"-";:NEXTI:PRINT
60872 PRINT:PRINT"Variables---Line numbers":PRINT:GOTO61010
60877 IFLP>10THENPRINT:LP=0:GOTO60880
60878 PRINT" ";LN$;
60879 LP=LP+1
60880 A$="":A=A+1:IFA>=DTHEN60850
60890 IFPEEK(A)=34THEN61040
60900 IF(PEEK(A)=129)+(PEEK(A)=128)THENA=D:GOTO60850
60910 IF(PEEK(A)>90)+(PEEK(A)<65)THEN60880
60920 A$=A$+CHR$(PEEK(A)):A=A+1:IFA=DTHEN60960
60930 P=PEEK(A):IF(P>47)*(P<91)*(P<>61)*(P<>58)THEN60920
60940 IFPEEK(A)=36THENA$=A$+"$"
60945 IFLEN(A$)<3THENA$=A$+" ":GOTO60945
60950 Z$=MID$(A$,3,1):Z=ASC(Z$):IFZ<>36THENA$=LEFT$(A$,2)
60960 IFLEN(A$)<3THENA$=A$+" ":GOTO60960
60970 A$=LEFT$(A$,3):IFA$="REM"THENA=D:GOTO60850
60990 FORX=1TOC:IFLEFT$(VS$(X),3)=A$THEN61060
61000 NEXTX:C=C+1:VS$(C)=A$+LN$:GOTO60880
61010 FORX=1TOC:PRINTVS$(X)
61020 NEXTX:END
61040 A=A+1:IF(PEEK(A)<>34)*(A<>D-1)THEN61040
61050 GOTO60910
61060 IFLN$=RIGHT$(VS$(X),LEN(LN$))THEN60880
61070 VS$(X)=VS$(X)+" "+LN$:GOTO60880
61100 INPUT"FROM ? ";LN$:IFASC(LN$)=65THENV=60000:RETURN
61110 X=VAL(LN$):INPUT"TO ? ";LN$:IFASC(LN$)=69THENV=60000:GOTO61130
61120 V=VAL(LN$)
61130 LN$=STR$(PEEK(A+2)+PEEK(A+3)*256):IFVAL(LN$)=XTHENRETURN
61140 IFVAL(LN$)=VTHENPRINT"ERROR FROM = ";X;"TO = ";V:END
61150 A=PEEK(A)+256*PEEK(A+1):GOTO61130
```

Utility Table/DP

```
60000 REM
60830 PRINT"♦♦♦♦♦    Table/DP    ♦♦♦♦♦":PRINT
60835 PRINT/P"♦♦♦♦♦    Table/DP    ♦♦♦♦♦":PRINT/P
60840 POKE8048,1:DIMVS$(255):A=25900:C=0:LP=0:V=60000
60841 FORI=1TO100:VS$(I)="":NEXTI:GOSUB61100
60842 PRINT:PRINT:PRINT"Line numbers processed":PRINT:PRINT
60844 PRINT/P:PRINT/P"Line numbers processed":PRINT/P:PRINT/P
60850 D=PEEK(A)+256*PEEK(A+1)
60860 LN$=STR$(PEEK(A+2)+PEEK(A+3)*256):A=A+3
60870 IFVAL(LN$)<>VGOTO60877
60871 PRINT:FORI=1TO39:PRINT"-";:NEXTI:PRINT
60872 PRINT:PRINT"Variables---Line numbers":PRINT
60873 PRINT/P:FORI=1TO79:PRINT/P"-";:NEXTI:PRINT/P
60874 PRINT/P:PRINT/P"Variables---Line numbers":PRINT/P:GOTO61010
60877 IFLP>10THENPRINT:PRINT/P:LP=0:GOTO60880
60878 PRINT" ";LN$;:PRINT/P" ";LN$;
60879 LP=LP+1
60880 A$="":A=A+1:IFA>=DTHEN60850
60890 IFPEEK(A)=34THEN61040
60900 IF(PEEK(A)=129)+(PEEK(A)=128)THENA=D:GOTO60850
60910 IF(PEEK(A)>90)+(PEEK(A)<65)THEN60880
60920 A$=A$+CHR$(PEEK(A)):A=A+1:IFA=DTHEN60960
60930 P=PEEK(A):IF(P>47)*(P<91)*(P<>61)*(P<>58)THEN60920
60940 IFPEEK(A)=36THENA$=A$+"$"
60945 IFLEN(A$)<3THENA$=A$+" ":GOTO60945
60950 Z$=MID$(A$,3,1):Z=ASC(Z$):IFZ<>36THENA$=LEFT$(A$,2)
60960 IFLEN(A$)<3THENA$=A$+" ":GOTO60960
60970 A$=LEFT$(A$,3):IFA$="REM"THENA=D:GOTO60850
60990 FORX=1TOC:IFLEFT$(VS$(X),3)=A$THEN61060
61000 NEXTX:C=C+1:VS$(C)=A$+LN$:GOTO60880
61010 FORX=1TOC:PRINTVS$(X)
61020 PRINT/PVS$(X):NEXTX:END
61040 A=A+1:IF(PEEK(A)<>34)*(A<>D-1)THEN61040
61050 GOTO60910
61060 IFLN$=RIGHT$(VS$(X),LEN(LN$))THEN60880
61070 VS$(X)=VS$(X)+" "+LN$:GOTO60880
61100 INPUT"FROM ? ";LN$:IFASC(LN$)=65THENV=60000:RETURN
61110 X=VAL(LN$):INPUT"TO ? ";LN$:IFASC(LN$)=69THENV=60000:GOTO61130
61120 V=VAL(LN$)
61130 LN$=STR$(PEEK(A+2)+PEEK(A+3)*256):IFVAL(LN$)=XTHENRETURN
61140 IFVAL(LN$)=YTHENPRINT"ERROR FROM = ";X;"TO = ";Y:END
61150 A=PEEK(A)+256*PEEK(A+1):GOTO61130
```



MZ 80K – UTILITY PROGRAM – RENUMBER

Renumber version 1.0

(For use with SHARP CASSETTE BASIC SP-5025)
(or SHARP DISC BASIC SP-6015)

The basic utility programs "Renumber/TAPE" and "Renumber/DISC" are aids designed to help programmers write SHARP MZ 80K Computer. These programs can be used to locate and renumber program line numbers. If you have a printer use utility "Renumber/P", if not use utility "Renumber". Copies of both programs are supplied on the cassette sent with these notes.

NOTE – either Renumber/P or Renumber should be loaded into the MZ 80K memory before typing in your program. Also it is important to remember that these Renumber programs use BASIC line numbers 60000 to 60460, consequently the program being developed MUST NOT contain line numbers within this range.

To use Renumber/P or Renumber enter the direct mode COMMAND RUN 60000, followed by carriage return. The computer then requests the user to enter the starting line number and the line encrement. The following example will help users understand how to operate these utilities:–

1. Example program

```
 60  FORI=1TO10
 70  Y=SIN(I)
 80  B=COS(I)
 90  PRINTI;Y;B
100  NEXTI
110  BS="TEST"
120  DIMDU(10),CS(10)
130  FORA=1TO10
140  DU(A)=A:CS(A)=BS:TS=TS+CS(A)
150  PRINTDU(A);TS
160  NEXTA
170  GOSUB210
180  REM
190  GOTO200
200  END
210  RETURN
```

User test program (Renumber to load Renumber/P or Renumber first)

```
Run 6000 CR
```

CR denotes carriage return

```
Renumber/TP

Starting line number is 60
Line number increment is 5
Number of lines processed is 16

ERRORS
```

Renumber/P output
No errors reported.

MZ 80K – UTILITY PROGRAM – RENUMBER'

2. Final form of example program

```
 60  FORI=1TO10
 65  Y=SIN(I)
 70  B=COS(I)
 75  PRINTI;Y;B
 80  NEXTI
 85  BS="TEST"
 90  DIMDU(10),CS(10)
100  FORA=1TO10
105  DU(A)=A:CS(A)=BS:TS=TS+CS(A)
110  PRINTDU(A);TS
```

```
115 NEXTA
120 GOSUB140
125 REM
130 GOTO135
135 END
140 RETURN
```

3. Commands and limitations

3.1 Enter commands from the keyboard as requested.

3.2 Only 256 lines at a time can be renumbered.

Renumber Utility

```
60000 REM
60010 PRINT"■◆◆◆◆◆   Renumber/D   ◆◆◆◆◆":PRINT:PRINT
60030 POKE8048,1:DIMTA(255):AL=44:AH=101:M=0:Q$=""
60040 INPUT"Starting line number ? ";SL
60050 INPUT"Line number increment ? ";IN
60080 H=INT(SL/256):L=SL-H*256
60090 D=256*AH+AL:A=PEEK(D+2):B=PEEK(D+3):LN=256*B+A
60100 IFLN=60000THEN60160
60110 TA(M)=LN:M=M+1
60120 POKED+2,L:POKED+3,H:L=L+IN:IFL>255THENL=L-256:H=H+1
60130 IFM>255THENPRINT"Line table full":GOTO60160
60140 AL=PEEK(D):AH=PEEK(D+1):GOTO60090
60150 REM second pass
60160 M=M-1:D=25899:PRINT"Number of lines processed is ";M+1
60180 PRINT:PRINT:FORI=1TO39:PRINT"-";:NEXTI:PRINT
60200 PRINT:PRINT:PRINT"ERRORS"
60210 D=D+4:L=PEEK(D-1)+256*PEEK(D):IFL>59999THENEND
60220 D=D+1:H=PEEK(D):IFH=13THEN60210
60230 IF(H<>137)*(H<>139)*(H<>173)THEN60220
60240 IF(H=173)*(PEEK(D+1)=137)THEND=D+1
60250 IF(H=173)*(PEEK(D+1)=139)THEND=D+1
60260 A=D
60270 D=D+1:H=PEEK(D):IFH=32THEN60270
60280 IF(H>47)*(H<58)THENQ=H-48:Q$=Q$+RIGHT$(STR$(Q),1):GOTO60270
60290 IFQ$=""THEN60220
60300 IFH=44THENGOSUB60330:GOTO60260
60310 GOSUB60330:IFH=13THEN60210
60320 GOTO60220
60330 Q=VAL(Q$):FORJ=0TOM:IFTA(J)=QTHEN60370
60340 NEXTJ
60350 PRINT"Old line number ";Q;" in line ";L:PRINT"          does not exist"
60360 GOTO60430
60370 QL=IN*J+SL:QL$=STR$(QL):QL$=RIGHT$(QL$,LEN(QL$))
60380 IFLEN(QL$)>(D-A-1)GOSUB60440:GOTO60430
60390 IFLEN(QL$)<(D-A-1)THENQL$=QL$+" ":GOTO60390
60400 FORZ=A+1TOD-1:Q$=MID$(QL$,Z-A,1):Q=VAL(Q$)+48:POKEZ,Q
60410 IFQ$=" "THENPOKEZ,32
60420 NEXTZ
60430 Q$="":RETURN
60440 PRINT"Change old line number ";Q;" to ";QL;" in new line ";L
60460 RETURN
```

Renumber Utility

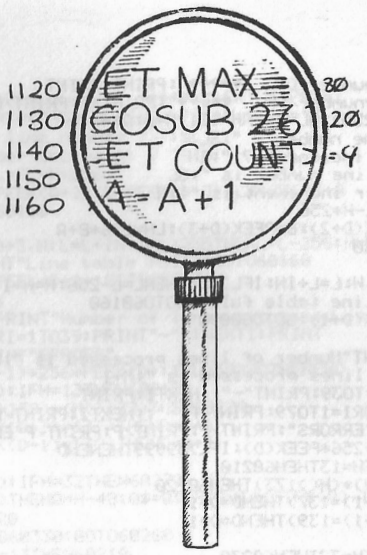
```
60000 REM
60010 PRINT"@♦♦♦♦♦    Renumber/TP    ♦♦♦♦♦":PRINT:PRINT
60020 PRINT/P"♦♦♦♦♦    Renumber/TP    ♦♦♦♦♦":PRINT/P:PRINT/P
60030 POKE10167,1:DIMTA(255):AL=6:AH=721:M=0:Q$=""
60040 INPUT"Starting line number ? ";SL
60050 INPUT"Line number increment ? ";IN
60060 PRINT/P"Starting line number is ";SL
60070 PRINT/P"Line number increment is ";IN
60080 H=INT(SL/256):L=SL-H*256
60090 D=256*AH+AL:A=PEEK(D+2):B=PEEK(D+3):LN=256*B+A
60100 IFLN=60000THEN60160
60110 TA(M)=LN:M=M+1
60120 POKED+2,L:POKED+3,H:L=L+IN:IFL>255THENL=L-256:H=H+1
60130 IFM>255THENPRINT"Line table full":GOTO60160
60140 AL=PEEK(D):AH=PEEK(D+1):GOTO60090
60150 REM second pass
60160 M=M-1:D=18437:PRINT"Number of lines processed is ";M+1
60170 PRINT/P"Number of lines processed is ";M+1
60180 PRINT:PRINT:FORI=1TO39:PRINT"-";:NEXTI:PRINT
60190 PRINT/P:PRINT/P:FORI=1TO79:PRINT/P"-";:NEXTI:PRINT/P
60200 PRINT:PRINT:PRINT"ERRORS":PRINT/P:PRINT/P:PRINT/P"ERRORS":PRINT/P
60210 D=D+4:L=PEEK(D-1)+256*PEEK(D):IFL>59999THENEND
60220 D=D+1:H=PEEK(D):IFH=13THEN60210
60230 IF(H<>137)*(H<>139)*(H<>173)THEN60220
60240 IF(H=173)*(PEEK(D+1)=137)THEND=D+1
60250 IF(H=173)*(PEEK(D+1)=139)THEND=D+1
60260 A=D
60270 D=D+1:H=PEEK(D):IFH=32THEN60270
60280 IF(H>47)*(H<58)THENQ=H-48:Q$=Q$+RIGHT$(STR$(Q),1):GOTO60270
60290 IFQ$=""THEN60220
60300 IFH=44THENGOSUB60330:GOTO60260
60310 GOSUB60330:IFH=13THEN60210
60320 GOTO60220
60330 Q=VAL(Q$):FORJ=0TOM:IFTA(J)=QTHEN60370
60340 NEXTJ
60350 PRINT"Old line number ";Q;" in line ";L:PRINT"                    does not exist"
60360 PRINT/P"Old line number ";Q;" in line ";L;" does not exist":GOTO60430
60370 QL=IN*J+SL:QL$=STR$(QL):QL$=RIGHT$(QL$,LEN(QL$))
60380 IFLEN(QL$)>(D-A-1)GOSUB60440:GOTO60430
60390 IFLEN(QL$)<(D-A-1)THENQL$=QL$+" ":GOTO60390
60400 FORZ=A+1TOD-1:Q$=MID$(QL$,Z-A,1):Q=VAL(Q$)+48:POKEZ,Q
60410 IFQ$=" "THENPOKEZ,32
60420 NEXTZ
60430 Q$="":RETURN
60440 PRINT"Change old line number ";Q;" to ";QL;" in new line ";L
60450 PRINT/P"Change old line number ";Q;" to ";QL;" in new line number ";L
60460 RETURN
```

Renumber Utility

```
60000 REM
60010 PRINT"B♦♦♦♦♦    Renumber/T    ♦♦♦♦♦":PRINT:PRINT
60030 POKE10167,1:DIMTA(255):AL=6:AH=721:M=0:Q$=""
60040 INPUT"Starting line number ?.";SL
60050 INPUT"Line number increment ? ";IN
60080 H=INT(SL/256):L=SL-H*256
60090 D=256*AH+AL:A=PEEK(D+2):B=PEEK(D+3):LN=256*B+A
60100 IFLN=60000THEN60160
60110 TA(M)=LN:M=M+1
60120 POKED+2,L:POKED+3,H:L=L+IN:IFL>255THENL=L-256:H=H+1
60130 IFM>255THENPRINT"Line table full":GOTO60160
60140 AL=PEEK(D):AH=PEEK(D+1):GOTO60090
60150 REM second pass
60160 M=M-1:D=18437:PRINT"Number of lines processed is ";M+1
60180 PRINT:PRINT:FORI=1TO39:PRINT"-";:NEXTI:PRINT
60200 PRINT:PRINT:PRINT"ERRORS"
60210 D=D+4:L=PEEK(D-1)+256*PEEK(D):IFL>59999THENEND
60220 D=D+1:H=PEEK(D):IFH=13THEN60210
60230 IF(H<>137)*(H<>139)*(H<>173)THEN60220
60240 IF(H=173)*(PEEK(D+1)=137)THEND=D+1
60250 IF(H=173)*(PEEK(D+1)=139)THEND=D+1
60260 A=D
60270 D=D+1:H=PEEK(D):IFH=32THEN60270
60280 IF(H>47)*(H<58)THENQ=H-48:Q$=Q$+RIGHT$(STR$(Q),1):GOTO60270
60290 IFQ$=""THEN60220
60300 IFH=44THENGOSUB60330:GOTO60260
60310 GOSUB60330:IFH=13THEN60210
60320 GOTO60220
60330 Q=VAL(Q$):FORJ=0TOM:IFTA(J)=QTHEN60370
60340 NEXTJ
60350 PRINT"Old line number ";Q;" in line ";L:PRINT"          does not exist"
60360 GOTO60430
60370 QL=IN*J+SL:QL$=STR$(QL):QL$=RIGHT$(QL$,LEN(QL$))
60380 IFLEN(QL$)>(D-A-1)GOSUB60440:GOTO60430
60390 IFLEN(QL$)<(D-A-1)THENQL$=QL$+" ":GOTO60390
60400 FORZ=A+1TOD-1:Q$=MID$(QL$,Z-A,1):Q=VAL(Q$)+48:POKEZ,Q
60410 IFQ$=" "THENPOKEZ,32
60420 NEXTZ
60430 Q$="":RETURN
60440 PRINT"Change old line number ";Q;" to ";QL;" in new line ";L
60460 RETURN
```

Renumber Utility

```
60000 REM
60010 PRINT"@♦♦♦♦♦    Renumber/DP    ♦♦♦♦♦":PRINT:PRINT
60020 PRINT/P"♦♦♦♦♦    Renumber/DP    ♦♦♦♦♦":PRINT/P:PRINT/P
60030 POKE8048,1:DIMTA(255):AL=44:AH=101:M=0:Q$=""
60040 INPUT"Starting line number ? ";SL
60050 INPUT"Line number increment ? ";IN
60060 PRINT/P"Starting line number is ";SL
60070 PRINT/P"Line number increment is ";IN
60080 H=INT(SL/256):L=SL-H*256
60090 D=256*AH+AL:A=PEEK(D+2):B=PEEK(D+3):LN=256*B+A
60100 IFLN=60000THEN60160
60110 TA(M)=LN:M=M+1
60120 POKED+2,L:POKED+3,H:L=L+IN:IFL>255THENL=L-256:H=H+1
60130 IFM>255THENPRINT"Line table full":GOTO60160
60140 AL=PEEK(D):AH=PEEK(D+1):GOTO60090
60150 REM second pass
60160 M=M-1:D=25899:PRINT"Number of lines processed is ";M+1
60170 PRINT/P"Number of lines processed is ";M+1
60180 PRINT:PRINT:FORI=1TO39:PRINT"-";:NEXTI:PRINT
60190 PRINT/P:PRINT/P:FORI=1TO79:PRINT/P"-";:NEXTI:PRINT/P
60200 PRINT:PRINT:PRINT"ERRORS":PRINT/P:PRINT/P:PRINT/P"ERRORS":PRINT/P
60210 D=D+4:L=PEEK(D-1)+256*PEEK(D):IFL>59999THENEND
60220 D=D+1:H=PEEK(D):IFH=13THEN60210
60230 IF(H<>137)*(H<>139)*(H<>173)THEN60220
60240 IF(H=173)*(PEEK(D+1)=137)THEND=D+1
60250 IF(H=173)*(PEEK(D+1)=139)THEND=D+1
60260 A=D
60270 D=D+1:H=PEEK(D):IFH=32THEN60270
60280 IF(H)47)*(H<58)THENQ=H-48:Q$=Q$+RIGHT$(STR$(Q),1):GOTO60270
60290 IFQ$=""THEN60220
60300 IFH=44THENGOSUB60330:GOTO60260
60310 GOSUB60330:IFH=13THEN60210
60320 GOTO60220
60330 Q=VAL(Q$):FORJ=0TOM:IFTA(J)=QTHEN60370
60340 NEXTJ
60350 PRINT"Old line number ";Q;" in line ";L:PRINT"                does not exist"
60360 PRINT/P"Old line number ";Q;" in line ";L;" does not exist":GOTO60430
60370 QL=IN*J+SL:QL$=STR$(QL):QL$=RIGHT$(QL$,LEN(QL$))
60380 IFLEN(QL$))(D-A-1)GOSUB60440:GOTO60430
60390 IFLEN(QL$)<(D-A-1)THENQL$=QL$+" ":GOTO60390
60400 FORZ=A+1TOD-1:Q$=MID$(QL$,Z-A,1):Q=VAL(Q$)+48:POKEZ,Q
60410 IFQ$=" "THENPOKEZ,32
60420 NEXTZ
60430 Q$="":RETURN
60440 PRINT"Change old line number ";Q;" to ";QL;" in new line ";L
60450 PRINT/P"Change old line number ";Q;" to ";QL;" in new line number ";L
60460 RETURN
```



MZ 80K – UTILITY PROGRAM – SEARCH

1. A SEARCH for SHARP CASSETTE BASIC SP 5025
2. A SEARCH for SHARP CASSETTE BASIC SP 5025 with printer
3. A SEARCH for SHARP DISC BASIC SP 6015
4. A SEARCH for SHARP DISC BASIC SP 6015 with printer

The BASIC utility programs are aids designed to help programers write SHARP BASIC programs. These programs can be used to search through a program to locate and list a specified variable and the line numbers where the variable occurs in the program.

Search may also be used to change the name of a variable or a string of ASCII characters. Search/TP and Search/D outputs a hard copy record of the variable name and line numbers. If you have a printer use the utility Search/TP or Search/DP. Copies of both programs are supplied on the cassette sent with these notes.

NOTE: either Search/P or Search should be loaded into the MZ 80K memory before typing in your program. Also it is important to remember that these Search progams use BASIC line numbers 60000 to 61340, consequently the program being developed MUST NOT contain line numbers within this range.

To use Search/P or Search enter the direct mode COMMAND RUN 6000, followed by a carriage return. The computer then requests the user to select either FIND, REPLACE, BLOCK or END command. The following example will help users to understand how to operate these utilities:–

1. Example Program

```
 10  FORI=1TO10
 20  Y=SIN(I)
 30  B=COS(I)                          user "test" program
 40  PRINTI;Y;B;                       (remember to load SEARCH)
 50  NEXTI
 60  BS="TEST"
 65  DIMDU(10),CS(10)
 70  FORA=1TO10
 80  DU(A)=A*CS(A)=BS:TS=TS+CS(A)
 90  PRINTDU(A);TS
100  NEXTA
200  END

RUN 60000 (CR)                       (CR) denotes carriage return
```

Direct mode command to operate Search program

```
**** Search ****                     Search output

Command  =  FIND                     FIND command used to search
String   =  I                        "test" for ASCII String I.

10  20  30  40  50                   Line numbers 10 – 50 contain variable I
```

MZ 80K – UTILITY PROGRAM – SEARCH

```
Command  =  REPLACE                  REPLACE command used to change
String-1 = I  String-2 = M           I to M in "test" program

Command  =  BLOCK                    Search "Block" changed from all lines
                                     to 20–90

Command  =  F                        Short form for FIND command
```

String = BS

60 80

Command = R Short form of REPLACE Command

String-1 = BS String-2 = MS

Command END Exit back to BASIC

2. FINAL FORM OF EXAMPLE PROGRAM

```
 10  FORM=1TO10
 20  Y=SIN(M)
 30  B=COS(M)
 40  PRINTM;Y;B
 50  NEXTM
 60  MS="TEST"
 65  DIMDU(10),CS(10)
 70  FORA=1TO10
 80  DU(A)=A^5/8CS(A)=MS:TS=TS+CS(A)
 90  PRINTDU(A);TS
100  NEXTA
200  END
```

```
 80  DU(A)=A:CS(A)?MS:TS?TS+CS(A)
```

LIMITATIONS

1. Only 256 lines at a time can be searched – use block command if the program is longer.
2. In the replace command "string-1" and "string-2" must be the same length.
3. Only ASCII characters can be searched for a replaced – remember BASIC reserved words are held as "tokens".
4. If entering from the keyboard a string with leading blanks then enter as " TEST ; i.e. quotes before the leading blanks.

Utility Search/T

```
60000 REM
60910 PRINT"B♦♦♦♦♦    Search/T     ♦♦♦♦♦":PRINT
60930 POKE10167,1:OM=0:L2=18437:V=60000:LP=0
60940 DIM L1(255):L=L2
60950 INPUT"Command = ";NL$:PRINT
60970 IFASC(NL$)=70THENGOSUB61005
60980 IFASC(NL$)=82THENGOSUB61130
60985 IFASC(NL$)=66THENGOSUB61290
60990 IFASC(NL$)=69THENEND
61000 L=L2:OM=0:GOTO60950
61005 FORI=0TO255:L1(I)=0:NEXTI
61010 INPUT"String ? ";P1$:IFP1$=""THEN61010
61030 L=L+4:LN=256*PEEK(L)+PEEK(L-1):IFLN>=VTHEN61100
61040 GOSUB61210:IFCH=13THEN61030
61050 L1(OM)=LN:OM=OM+1:IFOM>255THEN61080
61060 L=L+1:IFPEEK(L)<>13THEN61060
61070 GOTO61030
61080 PRINT"Too many - first 256 output"
61100 FORI=0TO255:IFL1(I)<>0THENPRINTL1(I);:PRINT#PL1(I);:LP=LP+1
61110 IFLP>10THENPRINT:LP=0
61120 NEXTI:PRINT:RETURN
61130 INPUT"String-1 ";P1$:INPUT"String-2 ";P2$
61150 IFLEN(P1$)=LEN(P2$)THEN61170
61160 PRINT"Unequal lengths ":GOTO61130
61170 L=L+4:LN=256*PEEK(L)+PEEK(L-1):IFLN>=VTHENRETURN
61180 GOSUB61210:IFCH=13THEN61170
61190 FORI=1TOLEN(P2$):POKE(L+I-1),ASC(MID$(P2$,I,1)):NEXTI
61200 L=L+LEN(P2$)-1:GOTO61180
61210 J=-1:L=L+1:CH=PEEK(L):IFCH=13THENRETURN
61220 IFCHR$(CH)=LEFT$(P1$,1)THEN61240
61230 GOTO61210
61240 FORI=1TOLEN(P1$)
61250 IFCHR$(PEEK(L+I-1))=MID$(P1$,I,1)THEN61270
61260 J=1
61270 NEXTI:IFJ=1THEN61210
61280 RETURN
61290 INPUT"FROM ? ";NL$:IFASC(NL$)=65THENL2=18437:V=60000:RETURN
61295 X=VAL(NL$)
61300 INPUT"TO ? ";NL$:IFASC(NL$)=69THENV=60000:GOTO61320
61310 V=VAL(NL$)
61320 A=18438
61325 NL$=STR$(PEEK(A+2)+PEEK(A+3)*256):IFVAL(NL$)=XTHENL2=A-1:RETURN
61330 IFVAL(NL$)=VTHENPRINT"ERR FROM = ";X;" TO = ";V:L2=18437:V=60000:RETURN
61340 A=PEEK(A)+256*PEEK(A+1):GOTO61325
```

Utility Search/TP

```
60000 REM
60910 PRINT"■♦♦♦♦♦   Search/TP     ♦♦♦♦♦":PRINT
60920 PRINT/P"♦♦♦♦♦   Search/TP     ♦♦♦♦♦":PRINT/P
60930 POKE10167,1:OM=0:L2=18437:V=60000:LP=0
60940 DIM L1(255):L=L2
60950 INPUT"Command = ";NL$:PRINT
60960 PRINT/P"Command = ";NL$
60970 IFASC(NL$)=70THENGOSUB61005
60980 IFASC(NL$)=82THENGOSUB61130
60985 IFASC(NL$)=66THENGOSUB61290
60990 IFASC(NL$)=69THENEND
61000 L=L2:OM=0:GOTO60950
61005 FORI=0TO255:L1(I)=0:NEXTI
61010 INPUT"String ? ";P1$:IFP1$=""THEN61010
61020 PRINT/P"String = ";P1$
61030 L=L+4:LN=256*PEEK(L)+PEEK(L-1):IFLN>=VTHEN61100
61040 GOSUB61210:IFCH=13THEN61030
61050 L1(OM)=LN:OM=OM+1:IFOM>255THEN61080
61060 L=L+1:IFPEEK(L)<>13THEN61060
61070 GOTO61030
61080 PRINT"Too many - first 256 output"
61090 PRINT/P"Too many - first 256 output":PRINT/P
61100 FORI=0TO255:IFL1(I)<>0THENPRINTL1(I);:PRINT/PL1(I);:LP=LP+1
61110 IFLP>10THENPRINT:LP=0
61120 NEXTI:PRINT:PRINT/P:RETURN
61130 INPUT"String-1 ";P1$:INPUT"String-2 ";P2$
61140 PRINT/P"String - 1 = ";P1$;" String - 2 = ";P2$
61150 IFLEN(P1$)=LEN(P2$)THEN61170
61160 PRINT"Unequal lengths ":PRINT/P"Unequal lengths":GOTO61130
61170 L=L+4:LN=256*PEEK(L)+PEEK(L-1):IFLN>=VTHENRETURN
61180 GOSUB61210:IFCH=13THEN61170
61190 FORI=1TOLEN(P2$):POKE(L+I-1),ASC(MID$(P2$,I,1)):NEXTI
61200 L=L+LEN(P2$)-1:GOTO61180
61210 J=-1:L=L+1:CH=PEEK(L):IFCH=13THENRETURN
61220 IFCHR$(CH)=LEFT$(P1$,1)THEN61240
61230 GOTO61210
61240 FORI=1TOLEN(P1$)
61250 IFCHR$(PEEK(L+I-1))=MID$(P1$,I,1)THEN61270
61260 J=1
61270 NEXTI:IFJ=1THEN61210
61280 RETURN
61290 INPUT"FROM ? ";NL$:IFASC(NL$)=65THENL2=18437:V=60000:RETURN
61295 X=VAL(NL$)
61300 INPUT"TO ? ";NL$:IFASC(NL$)=69THENV=60000:GOTO61320
61310 V=VAL(NL$)
61320 A=18438
61325 NL$=STR$(PEEK(A+2)+PEEK(A+3)*256):IFVAL(NL$)=XTHENL2=A-1:RETURN
61330 IFVAL(NL$)=VTHENPRINT"ERR' FROM = ";X;" TO = ";V:L2=18437:V=60000:RETURN
61340 A=PEEK(A)+256*PEEK(A+1):GOTO61325
```

Utility Search/D

```
60000 REM
60910 PRINT"B♦♦♦♦♦   Search/D     ♦♦♦♦♦":PRINT
60930 POKE8048,1:OM=0:L2=25899:Y=60000:LP=0
60940 DIM L1(255):L=L2
60950 INPUT"Command = ";NL$:PRINT
60970 IFASC(NL$)=70THENGOSUB61005
60980 IFASC(NL$)=82THENGOSUB61130
60985 IFASC(NL$)=66THENGOSUB61290
60990 IFASC(NL$)=69THENEND
61000 L=L2:OM=0:GOTO60950
61005 FORI=0TO255:L1(I)=0:NEXTI
61010 INPUT"String ? ";P1$:IFP1$=""THEN61010
61030 L=L+4:LN=256*PEEK(L)+PEEK(L-1):IFLN>=YTHEN61100
61040 GOSUB61210:IFCH=13THEN61030
61050 L1(OM)=LN:OM=OM+1:IFOM>255THEN61080
61060 L=L+1:IFPEEK(L)<>13THEN61060
61070 GOTO61030
61080 PRINT"Too many - first 256 output"
61100 FORI=0TO255:IFL1(I)<>0THENPRINTL1(I);:PRINT/PL1(I);:LP=LP+1
61110 IFLP>10THENPRINT:LP=0
61120 NEXTI:PRINT:RETURN
61130 INPUT"String-1 ";P1$:INPUT"String-2 ";P2$
61150 IFLEN(P1$)=LEN(P2$)THEN61170
61160 PRINT"Unequal lengths ":GOTO61130
61170 L=L+4:LN=256*PEEK(L)+PEEK(L-1):IFLN>=YTHENRETURN
61180 GOSUB61210:IFCH=13THEN61170
61190 FORI=1TOLEN(P2$):POKE(L+I-1),ASC(MID$(P2$,I,1)):NEXTI
61200 L=L+LEN(P2$)-1:GOTO61180
61210 J=-1:L=L+1:CH=PEEK(L):IFCH=13THENRETURN
61220 IFCHR$(CH)=LEFT$(P1$,1)THEN61240
61230 GOTO61210
61240 FORI=1TOLEN(P1$)
61250 IFCHR$(PEEK(L+I-1))=MID$(P1$,I,1)THEN61270
61260 J=1
61270 NEXTI:IFJ=1THEN61210
61280 RETURN
61290 INPUT"FROM ? ";NL$:IFASC(NL$)=65THENL2=25899:Y=60000:RETURN
61295 X=VAL(NL$)
61300 INPUT"TO ? ";NL$:IFASC(NL$)=69THENY=60000:GOTO61320
61310 Y=VAL(NL$)
61320 A=25900
61325 NL$=STR$(PEEK(A+2)+PEEK(A+3)*256):IFVAL(NL$)=XTHENL2=A-1:RETURN
61330 IFVAL(NL$)=YTHENPRINT"ERR FROM = ";X;" TO = ";Y:L2=25899:Y=60000:RETURN
61340 A=PEEK(A)+256*PEEK(A+1):GOTO61325
```

Utility Search/DP

```
60000 REM
60910 PRINT"B♦♦♦♦♦   Search/DP     ♦♦♦♦♦":PRINT
60920 PRINT/P"♦♦♦♦♦   Search/DP     ♦♦♦♦♦":PRINT/P
60930 POKE8048,1:OM=0:L2=25899:Y=60000:LP=0
60940 DIM L1(255):L=L2
60950 INPUT"Command = ";NL$:PRINT
60960 PRINT/P"Command = ";NL$
60970 IFASC(NL$)=70THENGOSUB61005
60980 IFASC(NL$)=82THENGOSUB61130
60985 IFASC(NL$)=66THENGOSUB61290
60990 IFASC(NL$)=69THENEND
61000 L=L2:OM=0:GOTO60950
61005 FORI=0TO255:L1(I)=0:NEXTI
61010 INPUT"String ? ";P1$:IFP1$=""THEN61010
61020 PRINT/P"String = ";P1$
61030 L=L+4:LN=256*PEEK(L)+PEEK(L-1):IFLN>=YTHEN61100
61040 GOSUB61210:IFCH=13THEN61030
61050 L1(OM)=LN:OM=OM+1:IFOM>255THEN61080
61060 L=L+1:IFPEEK(L)<>13THEN61060
61070 GOTO61030
61080 PRINT"Too many - first 256 output"
61090 PRINT/P"Too many - first 256 output":PRINT/P
61100 FORI=0TO255:IFL1(I)<>0THENPRINTL1(I);:PRINT/PL1(I);:LP=LP+1
61110 IFLP>10THENPRINT:LP=0
61120 NEXTI:PRINT:PRINT/P:RETURN
61130 INPUT"String-1 ";P1$:INPUT"String-2 ";P2$
61140 PRINT/P"String - 1 = ";P1$;" String - 2 = ";P2$
61150 IFLEN(P1$)=LEN(P2$)THEN61170
61160 PRINT"Unequal lengths ":PRINT/P"Unequal lengths":GOTO61130
61170 L=L+4:LN=256*PEEK(L)+PEEK(L-1):IFLN>=YTHENRETURN
61180 GOSUB61210:IFCH=13THEN61170
61190 FORI=1TOLEN(P2$):POKE(L+I-1),ASC(MID$(P2$,I,1)):NEXTI
61200 L=L+LEN(P2$)-1:GOTO61180
61210 J=-1:L=L+1:CH=PEEK(L):IFCH=13THENRETURN
61220 IFCHR$(CH)=LEFT$(P1$,1)THEN61240
61230 GOTO61210
61240 FORI=1TOLEN(P1$)
61250 IFCHR$(PEEK(L+I-1))=MID$(P1$,I,1)THEN61270
61260 J=1
61270 NEXTI:IFJ=1THEN61210
61280 RETURN
61290 INPUT"FROM ? ";NL$:IFASC(NL$)=65THENL2=25899:Y=60000:RETURN
61295 X=VAL(NL$)
61300 INPUT"TO ? ";NL$:IFASC(NL$)=69THENY=60000:GOTO61320
61310 Y=VAL(NL$)
61320 A=25900
61325 NL$=STR$(PEEK(A+2)+PEEK(A+3)*256):IFVAL(NL$)=XTHENL2=A-1:RETURN
61330 IFVAL(NL$)=YTHENPRINT"ERR FROM = ";X;" TO = ";Y:L2=25899:Y=60000:RETURN
61340 A=PEEK(A)+256*PEEK(A+1):GOTO61325
```

MZ 80K – UTILITY PROGRAM – SEARCH

Notes

Short forms for the command words are allowed – F for FIND, R for REPLACE, B for BLOCK and E for END.

2. At entry to SEARCH the program assumes that all the user programk will be searched.

3. The selection of a block of line numbers is allowed using the following forms:–

   Responding to "FROM ? " with ALL (short form A) will switch the program back to search all lines.

   Entering two distinct line numbers will define a search and replace block – for example:–

   10

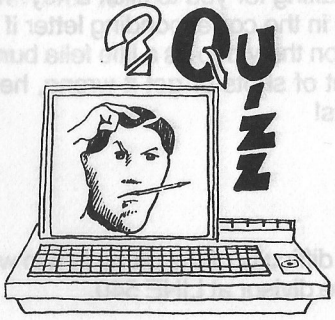
   100

   search FROM ? 100 — FROM ? ALL – sets search over
   block TO ? 200 — entire program

   200

   1000

   Responding with End to "TO ?" causes a search from line x to the end of the program. Note the short form END is E.



"QUIZZ" program for the SHARP MZ-80K

PROGRAM SPECIFICATION

QUIZZ is a general purpose question-and-answer type game. Although I've done it here as a Pop quiz, it's just as suitable for most subjects; eg music from the twenties (for PCW editorial staff!) or boring things like history, geography, etc. . .

The object of the game is to figure out the answer (line 3 of the display) by successively building up the clues (lines 1 and 2 of the display). Each of these lines consists of up to 36 dots, one for each capital letter: when you clik a key that key's letter pops into place on each line IF it occurs, rather like in Hangman.

The catch is that you've only got so many goes to work out line 3 (see the program narrative for how to set this). If you don't get the answer inside your allowance; of if you get it wrong, the machine will insult you, a little guy'll fall off the top of the screen, and your score for that game will be zero.

Get it right however, you'll be congratulated and your score'll accumulate. Either way the computer will fill in the dots so you can see what each line said.

Finally you'll be asked if you want another game. If you don't then your accumulated score will be presented as a percentage.

PROGRAM NARATIVE

In the program each individual game consists of 3 entries, printed on the video as three lines, one above the other. These are *1. The CLUE; 2* An additional HINT or whatever; *3. The ANSWER which you've to guess. Let's call the groups made up of these 3 lines ITEMS. Each item fits comfortably into one DATA statement.

The computer will choose one of the many ITEMs at random and print out its 3 entries as 3 lines of stars, a star for every letter. Beneath it'll print full instructions on what to do.

The program's now waiting for you to click a key: When you do it'll scan along each line and pop in the corresponding letter if it occurs. Your shots are counted and shown on the video as a litle fella bumping one to the right each time. If you run out of shots or get it wrong, he falls off the top with appropriate sound effects!

Features

*1. You can adjust the difficulty of the game in two ways:
   a) By changing the divisor at LINE 540.

```
100 REMARK♥ GENERAL QUIZ PROGRAM
105 REMARK♥  by JOHN TRIPPICK 1981
110 REMARK♥ Set variables
120 PRINT"@"
130 TRIES=0
140 ALLOWANCE=0
150 ITEMS=20:REMARK♥ ITEMS is the                    number of GAME items
160 DIM LINE$(3),L(3),T(3)
170 FOR K=1 TO 3
180 LINE$(K)=""
190 NEXT K
200 REMARK♥ Print headings
210 READ HEADING$
220 PRINT HEADING$
230 READ KEY$
240 PRINT KEY$
250 RESTORE
260 REMARK♥ Print board
270 LET A=53448:B=53485:C=73
280 FOR K=1 TO 38
290 POKE A,C
300 A=A+1
310 NEXT
320 A=53488
330 FOR K=1 TO 8
340 POKE A,C:POKE B,C
350 LET A=A+40:B=B+40
360 NEXT K
370 FOR K=1 TO 38
380 POKE A,C
390 A=A+1
400 NEXT K
410 REMARK♥ Choose item (all three                   lines)
420 CHOICE=INT(ITEMS*RND(1))+3
430 IF CHOICE = PREVIOUS THEN 420
440 LET PREVIOUS = CHOICE
450 FOR K=1 TO CHOICE
460 READ LINE$(1),LINE$(2),LINE$(3)
470 NEXT K
480 RESTORE
490 REMARK♥ Set up line controls
500 FOR K=1 TO 3
510 LET L(K)= LEN(LINE$(K))
520 LET T(K)= (40-L(K))/2
530 NEXT
540 LET ALLOWANCE = L(3)/1.75
550 REMARK♥ Set up little man
560 LET A=53406:B=71
570 FOR K=1 TO ALLOWANCE
580 POKE A,B
590 A=A-1
600 NEXT
610 LET B=202
620 POKE A,B
630 REMARK♥ Print dots for each letter
640 PRINT"■■■■■"
650 FOR K=1 TO 3
660 PRINT"■":PRINT TAB(T(K));
```

```
670 FOR I= 1 TO L(K)
680 M$=MID$(LINE$(K),I,1)
690 LET M = ASC(M$)
700 IF (M < 65)+(M > 90) THEN PRINT M$;      :GOTO 720
710 PRINT"*";
720 NEXT I,K
730 REMARK♥ Print the rules
740 PRINT"■■■"
750 PRINT"       Clik any letters until you       "
760 PRINT"     think you know what's on line 3     "
770 PRINT"     then type the [=] sign followed     "
780 PRINT"       by the name and the CR key";
790 REMARK♥ Accept player's guess
800 MUSIC "A0G0A0"
810 GET GUES$:IF GUES$=""THEN 810
820 IF GUES$="=" THEN 1010
830 TRIES=TRIES + 1
840 IF TRIES > ALLOWANCE THEN 1060
850 REMARK♥ Fill in guessed letter                    wherever it occurs
860 PRINT"■■■■■■■■"
870 FOR K=1 TO 3
880 PRINT TAB(T(K));
890 FOR I=1 TO L(K)
900 M$=MID$(LINE$(K),I,1)
910 IF GUES$=M$ THEN PRINT M$;:                MUSIC"A0":GOTO930
920 PRINT"■";
930 NEXT I:PRINT"■"
940 NEXT K
950 REMARK♥ Move the little man
960 POKE A,0
970 A=A+1
980 POKE A,B
990 GOTO 800
1000 REMARK♥ Accept guessed name
1010 PRINTTAB(T(3)-2);
1020 MUSIC "▂A1▔G1▂A1"
1030 INPUT GUES$
1040 IF GUES$=LINE$(3) GOTO 1300
1050 REMARK♥ Else guess proved wrong.                 Do duffers' endins.
1060 PRINT"■■■■■■■■■■■■■■■■■";
1070 FOR K=1 TO 5
1080 PRINT"Duffer! ";:NEXT
1090 FOR J=7 TO 4 STEP -1
1100 FOR K=1 TO 3
1110 TEMPO K:MUSIC "B0"
1120 TEMPO J:MUSIC "G0"
1130 NEXT K,J
1140 TEMPO 4
1150 REMARK♥ Push little man off the                  top.
1160 FOR K=1 TO ALLOWANCE-TRIES
1170 POKE A,0:POKE C,202
1180 A=A+1:C=C+1:POKE A,B
1190 MUSIC "B0"
1200 NEXT K
1210 FOR K=1 TO 5:B=201
```

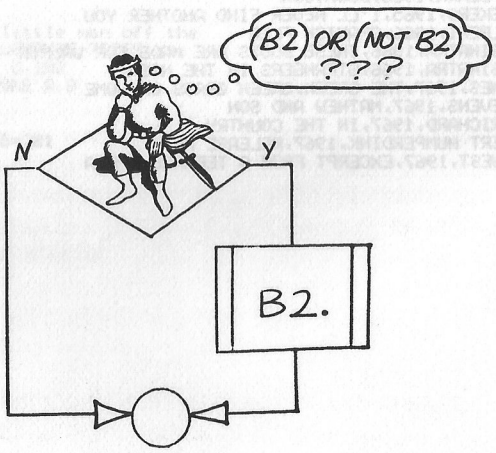
```
1220 FOR J=1 TO 4
1230 POKE A,0
1240 LET A=A+40:B=B+1
1250 POKE A,B
1260 MUSIC "B0"
1270 NEXT J,K
1280 GOTO 1340
1290 REMARK♥ Winners' ending
1300 PRINT"■■■■■■■■■■■■■■■■■  CONGRATULATIONS!   CONGRATULATIONS!
1310 MUSIC "G0A0G0A0"
1320 POINTS= POINTS+1
1330 REMARK♥ Fill in the answer
1340 PRINT"■■■■■■■■"
1350 FOR K=1 TO 3
1360 PRINT TAB(T(K));
1370 FOR I=1 TO L(K)
1380 M$=MID$(LINE$(K),I,1)
1390 PRINT M$;
1400 MUSIC "A0"
1410 NEXT I:PRINT"■"
1420 NEXT K
1430 REMARK♥ Tot up the score so far.                Ask if another game.
1440 GAMES =GAMES+1
1450 FOR K=1 TO 3000:NEXT
1460 PRINT"■■■■■■■■■■Want another game? (Y or N)
1470 GET KEY$:IF KEY$=""THEN 1470
1480 IF KEY$="Y" THEN 120
1490 POINTS= INT(POINTS*100/GAMES)
1500 MUSIC "T1A1R1A1R1_A1"
1510 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■Your score is ";POINTS;"%"
1520 STOP:END .
1530 REMARK: Data section
1540 DATA POP QUIZ
1550 DATA Lines 1 to 3: Singer ,Year, Title
1560 DATA DUMMY
1570 DATA ELVIS PRESLEY,1961,ARE YOU LONESOME TONIGHT
1580 DATA DUANE EDDY,1961,RING OF FIRE
1590 DATA PETULA CLARK,1961,ROMEO
1600 DATA JIMMY DEAN,1961,BIG BAD JOHN
1610 DATA ACKER BILK,1962,STRANGER ON THE SHORE
1620 DATA CLIFF RICHARD,1962,THE YOUNG ONES
1630 DATA GERRY AND THE PACEMAKERS,1963,HOW DO YOU DO IT
1640 DATA THE BEATLES,1963,FROM ME TO YOU
1650 DATA THE CRYSTALS,1963,AND THEN HE KISSED ME
1660 DATA PETER,PAUL AND MARY,1963,BLOWIN' IN THE WIND
1670 DATA HERMAN'S HERMITS,1964,I'M INTO SOMETHING GOOD
1680 DATA PETULA CLARK,1965,DOWNTOWN
1690 DATA THE SEEKERS,1965,I'LL NEVER FIND ANOTHER YOU
1700 DATA HERP ALPERT,1966,SPANISH FLEA
1710 DATA NANCY SINATRA,1966,THESE BOOTS ARE MADE FOR WALKIN'
1720 DATA FRANK SINATRA,1966,STRANGERS IN THE NIGHT
1730 DATA TOM JONES,1967,THE GREEN,GREEN GRASS OF HOME
1740 DATA CAT STEVENS,1967,MATHEW AND SON
1750 DATA CLIFF RICHARD,1967,IN THE COUNTRY
1760 DATA ENGELBERT HUMPERDINK,1967,RELEASE ME
1770 DATA KEITH WEST,1967,EXCERPT FROM A TEENAGE OPERA
```

ALLOWANCE is the variable to determine how many shots you'll be allowed. It's calculated by dividing the ANSWER line of the chosen item by 1.75. Increase to 2 or more and make it harder!

b) The program only stars out CAPITALS, so you can show figures, special symbols and small letters.
This is controlled at LINE 700.

*2. It's also very easy to add or change items. . . Just remember that each entry shouldn't be longer than 36 characters. Also change variable ITEMS at LINE 150: This is the number of game questions and shouldn't include your subject title and line key.

John Trippick
ML1 4AA



BOOLEAN ALGEBRA ON THE MZ-80K

by
P.L. BIRCH

The term 'Boolean algebra' suggests a rather esoteric mathematical discipline, but in fact it is a technique which you use all the time in your programming, perhaps without realising it. A fuller understanding of the ways in which Boolean methods are used by the computer can lead to more economical code, and a number of useful programming 'tricks'.

The 'IF . . . THEN' Statement

The computer uses Boolean algebra to make decisions on whether statements, or groups of statements, are true or false. True statements are assigned the value '-1', while false statements are assigned the value '0'. Though in fact the computer will recognise any non-zero number as representing a true statement. Let's see how this applies to the 'IF . . . THEN' statement which is ts most common use.

Look at the following simple program:

```
10  X=1
20  IF X=1 THEN PRINT "TRUE": END
30  PRINT "FALSE"
```

Running this will of course result in the word 'TRUE' being printed. The computer has examined the statement 'X=1', found it to be true and assigned it the Boolean value '-1'. This 'truth' value then activates the 'THEN' part of the statement and the word 'TRUE' is printed. If line 10 is changed to 'X=23', the statement 'X=1' in line 20 is now false, and will therefore be given the value '0'. The 'THEN' statement will not now be invoked and the program will continue to line 30. To convince yourself what this is indeed what has happened, try the following experiment. Simply change line 20 to:

```
20  IF 0 THEN PRINT "TRUE": END
```

The computer prints the word 'FALSE'. Instead of giving the computer a statement to evaluate here, we have given it a Boolean 'false' value already worked out. It recognises this and acts accordingly. Similarly, changing line 20 to 'IF -1 THEN. . .' will cause the computer to print 'TRUE' since it recognises the Boolean 'truth' value. -1 may be replaced with any non-zero number with the same effect.

AND/OR

The true value of this mode of working comes when we wish the computer to evaluate groups of statements. To enable this to be done Sharp BASIC provides what are known as 'logical operators'. Two of thse are described in the SP-5025 manual: 'AND', represented by the '★' symbol, and 'OR', represented by '+'. To demonstrate the use of these operators we can change line 20 of our program as follows:

```
20  IF (X=1)*(Y=1) THEN PRINT "TRUE": END
```

By assigning various different values to X and Y in line 10, we can test this out. The program will print 'TRUE' only if BOTH X AND Y are set to 1, in other words only if BOTH statements are true. Now that we know the Boolean basis on which the computer makes decisions we can see why this should be so. The two statements, 'X=1' and 'Y=1' are each assigned their appropriate Boolean value according to their truth or falsity, and the '★' operates on them exactly as it does in its more familiar role as a multiplication sign. If both statements are true then -1★-1=1 – a non-zero value representing truth – and the 'THEN' statement is invoked. If either statement is false (value 0), however, the whole expression will evaluate to zero, because anything multiplied by zero gives zero. The 'OR' operator may be similarly tested by substituting '+' for '★' in line 20. The computer will now print 'TRUE' if EITHER statement is correct, or if they are BOTH correct. Again the '+' sign is acting in its usual arithmetic sense:

-1+-1=-2	both statements true – non-zero result indicates truth.
0+-1=-1	one true, one false – still a non-zero result.
0+0=0	both false, therefore combined result is false.

Statements connected by the two operators described may be joined in chains of any desired length, subject of course to the maximum length of program line permitted by the BASIC interpreter. Thus:

```
20  IF (A=1)+(B=1)+(C=1)+(D=1) THEN. . .
```

will invoke the 'THEN' statement if any one or more of the statements is true. Change the '+' symbols to '★' signs and all four must be true before the 'THEN' comes into play The logical operators have their usual mathematical priorities so

```
20  IF (A=1)+(B=1)+(C=1)★(D=1) THEN. . .
```

will work if A=1 OR if B=1 OR if BOTH C AND D=1. These priorities may be modified with brackets in the usual way, for instance:

```
20  IF ((A=1)+(B=1)+(C=1))★(D=1) THEN. . .
```

would give a true result only if D AND one of the other thre variables were equal to one. If this seems confusing on paper, a few minutes spent playing around with various combinations on the computer should make it clearer.

IF . . . THEN Party Tricks.

Most of the above techniques may be understood and used without an appreciation of the Boolean algebra underlying the 'IF . . . THEN' statement. We can now, however, start to probe a little deeper and uncover one or two wrinkles which are not in the official Sharp BASIC manual.

The '+' sign used as a logical operator in an 'IF' statement represents what is known in logic as an 'inclusive or'. In other words it returns a 'true' signal if either or BOTH of the statements related by it are true. Another operator which is sometimes useful is the 'exclusive or'. This will return 'true' only if ONE of the two statements is true, NOT if they are both false or both true. The Sharp manual makes no mention of the existence of an exclusive 'or' operator for use in 'IF . . . THEN' statements, but it does exist. It is the minus sign '−'. You can try this out by substituting '−' for '+' or '★' in line 20 of our usual program. You will find that the computer prints 'TRUE' if X is equal to one and Y is not, or vice versa, and 'FALSE' if they are both equal to one, or both not equal to one. The other arithmetic operators ('/' and '↑') may also be used in 'IF' statements, but '/' tends to cause error messages as it leads to division by zero, and '↑' does nothing particularly useful than I can see. However, you might have some fun playing about them them!

Since all the above tests may be carried out on numbers using '<', '>', or '<>' instead of '=', it is not generally useful to be able to NEGATE a test of truth by Boolean manipulation. To test the UNTRUTH of X=1, we can simply test the TRUTH of X1. Negation can be useful however when we are dealing with strings rather than numerals. Some BASICs allow statements of the form: IF A$<>"X" . . . but Sharp SP-5025 BASIC does not. It allows us only to test the equality of strings. This often leads to awkward constructions such as:

```
100  IF A$="X" THEN 120
110  GOTO 1000
120  . . . . .
```

when a branch is required conditional on the strings being unequal. Our knowledge of Boolean algebra enables us to overcome this by NEGATING the test of equality. Change the program to:

```
100  IF (A$="X")+1 THEN 1000
120  .....
```

If the statement 'A$="X" is true it will be assigned the value '−1'. Our new program will add one to this, giving 0. The computer will interpret this as untrue and will not branch. If the statement is untrue it will be assigned the value '0'. 1 added to this gives 1, a non-zero value, and the computer will branch to line 1000, interpreting the statement as true. One word of warning if you are using this method to negate a series of statements connected by the '+' operator. If more than one statement is true this will lead to a total 'truth' value of perhaps −2 or −3 being calculated. This will not be negated by +1 as in the above example. The safest way out is to use the 'SGN' function, which returns a value of −1 for any negative number, as follows:

```
100  IF SGN((A$="X")+(B$="Y"))+1 THEN 1000
```

Once again it's well worth while spending some time playing around with this technique so that you can become familiar with it.

Avoiding IF . . . THEN.

The Boolean functions described above were provided in the BASIC interpreter principally for use within 'IF . . . THEN' statements. Since they are closely related to the normal arithmetic routines of the interpreter, however, they can also be used in other situations, though once again the Sharp manual does not tell us this. When the interpreter encounters an expression of equality, or an expression of inequality (using '' or '') totally enclosed in parentheses, it will evaluate the expression for truth or falsity and assign to it the appropriate Boolean value. To confirm this enter the following to the computer:

```
X=4:PRINT (X=0),(X0)
```

The computer prints '0' and '−' respectively, since the first statement is false and the second true. Omission of the parentheses will result in an eror message, by the way. We have the mean, then, to assess the truth of falsity of an expression without recourse to the 'IF . . . THEN' statement, but what use is this to us in practical programming? Well, in some circumstances it can save a good deal of program space. Let's look at a simple example. Suppose you are programming a 'Battleships' type game. There are three types of ship: battleships, cruisers and destroyers. You wish to compute a score which depends on the type of ship sunk. 6 points are to be awarded for a battleship, 4 for a cruiser and 2 for a destroyer. The obvious way to code this is something like the following:

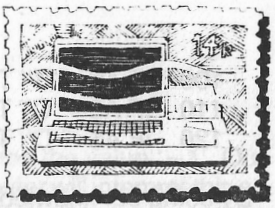
```
100  IF A$="B" THEN SC=6
110  IF A$="C" THEN SC=4
120  IF A$="D" THEN SC=2
```

By using our knowledge of Boolean algebra, however, we can condense this all into one line of code, as follows:

```
100  SC=-6*(A$="B")-4*(A$="C")-2*(A$="D")
```

The score relating to the true statement is the only one which will be accumulated into 'SC' since the other two will be multiplied by the zero Boolean constants. The minus signs take care of the fact that the Boolean 'truth' indicator is −1, and thus ensure that 'SC' ends up as a positive value. These Boolean numbers can be subjected to any of the mathematical operations or functions of which the computer is capable, but be careful of those which would create an error condition if applied to zero, or to a negative number.

I hope this article will encourage you to examine the decision making capabilities of your MZ-80K more closely. The classical use of the 'IF . . . THEN' statement is often the best course of action, but sometimes the techniques discussed here will make your life easier, and your programs more economical in memory.



Dear Sirs:

With reference to Issue Two of "Sharpsoft User Notes", in particular the Hints section, I would like to add my discoveries to your list. These are a few of the most useful:

1. POKE 4464,1 changes to lower case letters and graphics but does NOT switch the LED to red. Using 0 as the argument returns the computer to upper case as usual.
2. USR (33) (0021H) records the program header, and USR (36) (0024H) the program. Note: this can be used if the command NEW has been executed, thus saving a potentially lost program.
3. A RESET switch can be fitted to the socket on the main PCB as follows: connect either the two holes nearest the RAM chips or the opposite two holes via a push-to-make switch. (See diagram). The middle two holes do nothing. If a conventional switch is fitted, two presses will of course be necessary.
4. Address 17828 decimal contains the ASCII value of one key being pressed, and can be accessed thus: first POKE 10167,1 (0 will restore the PEEK protect) and then use a GET without the usual IF . . . THEN tacked on the end. On the next line put whatever variable you like equalling PEEK(17828) and this will become the ASCII code of the key currently depressed. After this the user can adapt the routine how he likes; an effective use is in games, because this function eliminates the need for constant hitting of a specific key. Once a key is depressed, the computer needs no more presses to achieve continuous action. This is certainly very difficult to describe and is best demonstrated on an actual machine.
5. A line with the number 0 can be achieved thus: make line 1 the line you want to be line 0, and then POKE 18440,0 and POKE 18441,0. The line 1 will now be line 0, and cannot be deleted from the program (unless it is renumbered using a BASIC toolkit routine).

Other points I would like to make are: what is a global variable? Disc as in drive is spelt disk as is programme spelt program in terms of computers.

I hope the above will be of use to you in the future. I cannot emphasise the superiority of the PEEK(17828) routine over a normal GET, it really needs trying out on an MZ-80K.

Thanks for an enlightening and very readable publication.

Yours faithfully,

P. Clark

SS1 3QP

Note Mr Hollingdale, the author of the enclosed music programs, has given permission for their use in any situation.

```
8 TEMP06
10 A$="_F5D8F5#C8_F5DGF_#AD8"
11 B$="#A5G#DAFD_B66F3#D7D_G8_#F5_FD7F5#C_A7_G5#DG_AD_#A8"
20 C$="A3F1DAFD"D#AG#AGEGE#CGE#CAFDAFD"A"F"D"F"DA"D#AG"D#AG"G"E"C"E"CG"C"
21 D$="A1F"CAF"F"D#A"D#AG"G"E"C"E"CA"A"F"D"F"D#A"#A"
22 E$=""G1"E"G"E"C"#B"A"F"A"F"C"A"F"D"F"DA"F"D#A"DA#F"D"
23 F$="#A1G#AGD#AGC#AGE#AAFAFCFD_#AFC_#AE"
24 G$="C1_AFC_AC_A_FC_A_FC_A_#FC_A_#FD#_A_#FD_A_#FC"
25 H$="_#A1_GD_#A_GD_#A_GE_#A_G#F_#A_G1G1D_#AG#D_AGD_A#F"
26 I$="D1_#AGD_#AD_#A_GD_#A_GD_B_#GD_B_#GF_B_#GE_B_#GD"
27 J$="C1_AEC_AEC_A#FC_A#GC_AAECAF_BAE_B#G"
28 K$="E1CAEC"CFD#AGE#AAFAC_AAD_BGE#CG"
29 L$="F1D"DAF"F#AG"#D"CA"#D"D#A"DFD"DGE"CA#FC"
30 M$="#A1G"D#AG"G"#DG"G"DG"G"#CA"G"E"C"#A"E"C"A"E"C"G"
31 N$=""F1"DAFDAFDBFD"#CFD"DFDAFD#AE#CA"
32 O$="E1#CGD_AFD_#AEC_F#D_#A_FD_A_FD_#A_ED_A_E#C"
33 P$="_A1_#F_AC_AC#DC#D#F3_#A1_G_#AD_#ADGDG#A3"
34 Q$="#A1G"E#AG"G#AG"E#AG"#C#AG"E#AG"G#AG"E#AG"#C"
35 R$="A1#F"DA#F"#DA#F"DA#F"CA#F"DA#F"#DA#F"DA#F"C"
36 S$="G1DG#AG#A"D#A"D"G"D"G"#A"A"G"F"E"D"#CA"#C"E"#C"E"
37 T$=""G1"E"D"#CBAGEG#AG#A"#CG"E"#C#A"E"#CA"E"#CG"E"
38 U$=""C1A"D#AG"#DA#F"DA#F"CGD#AGF#AGD##AGD#A"
39 V$="G1#C#AE#C#AE#CAE#CGD_BFD#CFD_BFD_AF"
40 W$="#C1_GE#C_#AE#C_AE#C_GE_B_#GD#C_#GED_#GF_G3"
41 X$="_G1_E_G#C_G#CE#CEG3F1DFAFA"DA"DF3"
42 Y$="F1B"D"F"D"F"#G"F"#G"B"#G"F"#A"G"E"A"#F"#D"#G"F"D"
44 Z$=""G1"E"#C"#F"#D"C"F"DE"E"#C#A"#D"CA"D7"
100 MUSICA$,B$,A$,B$
150 FORX=1TO2000:NEXTX
199 TEMP05
200 MUSICC$,D$,E$,F$,G$
210 MUSICH$,I$,J$,K$,L$
220 MUSIC M$,N$,O$,P$,Q$
230 MUSIC R$,S$,T$,U$,V$
240 MUSIC W$,X$,Y$,Z$
250 GOTO150
```

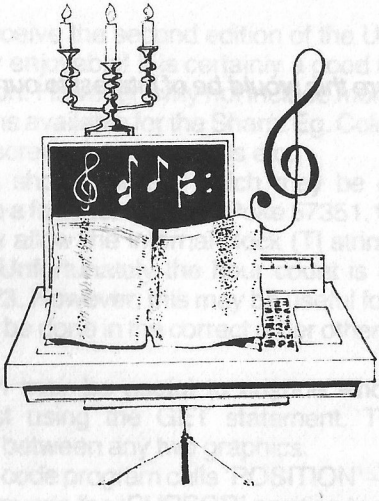
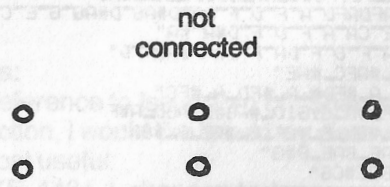


DIAGRAM FOR RESET SWITCH

returns to monitor,
does NOT clear memory



P.S. Sharp says $1200H for BASIC cold start, and $1260 for warm start. The latter is incorrect, as 1260H is in the middle of a routine. 124BH will be found to be correct.

Dear Sharpsoft,

I have discovered that if you enter:

POKE 10680,1

in the start of a program, it cannot be listed. This can be cancelled by entering:

POKE 10680,0

If you would like a routine that stops the program in use being "broken into", I have a short listing that can be entered at the very start of a program.

P.HENDRIE
CB3 0EG

Yes please _ I am sure this would be of interest to our readers.

R.J. WALSBY
Surrey

Mr. Walsby,

Work is being carried out in this area and as soon as something aspires we will let you know. Ed.

To SHARPSOFT Ltd.

Simple BASIC SP-5025 line renumbering

The following short program renumbers lines of a BASIC program starting from N and increasing in steps of S. It does *not* adjust the line numbers of GOTO and GOSUB commands but for a simple program these can subsequently be adjusted using a manual edit.

```
5000 J=18438:M=256:N=100:S=10
5010 JN=PEEK(J)+M*PEEK(J+1)
5020 LN=PEEK(J+2)+M*PEEK(J+3)
5030 IF LN=5000 THEN STOP
5040 HB=INT(N/M): LB=N-HB*M
5050 POKE J+2,LB:POKE J+3,HB
5060 J=JN:N=N+S:GOTO 5010
```

Enter this program after the program to be renumbered and then enter RUN 5000.

C.J.McD WOOD
BN6 8NE

Dear Sirs,

I was pleased to receive the second edition of the User Notes. The Star Wars program is very enjoyable! It is certainly a good idea to print a major program in each edition. However, why not include more information on the various hardware items available for the Sharp. Eg. Colour, High Resolution graphics, 80 column screen, interface units etc.

I have enclosed a short program which may be of interest to some readers. Also here are a few more Pokes: Poke 57351,116: Poke 57359,09: Poke 57349,02. They allow the internal clock (TI string) to count down to 1/60th of a second. Unfortunately the hour count is lost and the minute count's maximum is 23. However, this may be useful for some games etc. (NB. The Pokes must be done in the correct order otherwise the conversion will not work.)

This short program may be useful to anyone who wishes to have a flashing cursor whilst using the GET statement. The cursor is user-definable and 'blinks' between any two graphics.

The short machine-code program calls 'POSITION' – a monitor machine-code routine which converts the 'CURSOR' position to an absolute address

Dear Sir,

I have a suggestion for the new User Notes, that the article printed on page 109 of the November 1980 Practical Computing, concerning Cursor Addressing be reproduced in the User Notes. I have found that the resultant ease of cursor control superb and it is likely that many MZ-80K users did not see the original article.

P.E. PARKER
GU14 7DT

Due to copyright we cannot reprint the above article – suggest readers obtain a copy from Practical Computing.

Dear Sir,

1. If B.P. Windibank wanted to generate a pause in a BASIC program he could have omitted the PRINT " "; and saved himself a few bytes (and coding effort).
2. POKE 4464,1 gives lower case letters
   POKE 4464,0 gives upper case letters
3. POKE 10608,1 makes a program unlistable
   POKE 10680,0 restores.

The following shows how continuous movement can be made only when a key is depressed:–

```
10  POKE 10167,1
20  GET A$:IF PEEK (17828)=0 THEN 20
30  PRINT "  ^ Z";:GOTO 20
```

To find the numeric value of a key, run the following:–

```
10  GET A$:PRINT PEEK (17828);:GOTO 10
```

If a lot of printing is to be done in a program spaced equally across the screen, put the text into a string and then call the print subroutine.

e.g.

```
10   A$="THIS IS":GOSUB 500
20   A$="WHATEVER YOU WANT TO PRINT": GOSUB 500
30   END
500  PRINT TAB((40–LEN(A$))/2);A$:RETURN
```

B. LOCKEY
Woolwich

Dear Sir,

I have noticed in certain arcade games that 8 keys are used to control your piece on the screen, horizontally, vertically and diagonally. While this is all right for slow games it is impossible for fast ones.

So is it possible to fix joysticks to the MZ-80K, because this is what these 8 keys are really doing. I feel with SHARP's excellent graphic keys this would be a natural extension.

(D000-D3FF) and returns with the value in the HL register. This value is then loaded into 'VIDPOS'.

USR(2559) calls another monitor routine which 'blinks' the cursor on the screen pointed to by 'VIDPOS'.

N= the number of character entries required.
CO= any graphic code corresponding to the Display Code Table in the BASIC manual i.e. 0–255.
C1= similar to CO.

```
  5  POKE 4498,CO: POKE 4494,C1
 10  GOSUB 100
 15  FOR T= 1 TO N
 20  USR(50000)
 25  USR(2559): GET AS: IF AS= " " THEN 25
 30  B£=BS+AS
 35  PRINTAS;
 40  NEXT
 45  END
100  LIMIT 50000
105  FOR U= 50000 TO 50006
110  READ F: POKE U,F
115  NEXT
120  RETURN
125  DATA 205,177,15,34,143,17,201
130  REM CALL OFB1H
135  REM LD (118F),HL
140  REM RET
```

P. SHARMAN
Lincolnshire

Dear Sir,

I have a couple of hints that may be handy for other MZ-80K users.

1. POKE 6637,80
   disenables the SHIFT and BREAK keys.
   POKE 6637,30 reenables.
2. POKE 10407,0
   allows 'GET' to continue normally, for example

```
10  GET A$:IF A$=" " THEN 10
20  PRINT A$
30  GOTO 10
RUN
```

Dear Sir,

For anyone who has purchased the ZEN assembler, and who has typed in the source listing given in the manual the following modifications will be found useful as it allows binary numbers to be used by post-scripting the number with a 'B' (eg. 11011B)

1. ABOVE LINE 976 (page 17) INSERT

```
LD C,2
CP 'B'
JR 2,CVO
```

ie. between JR 2,CVO
and LD C,10

2. CHANGE LINE 1002 (page 18)
From ADD HL,HL to CV4: ADD HL,HL
3. CHANGE LINE 994 (page 18)
From JR NZ,CV3 tok JR NZ,CV5
4. BELOW LINE 1009 (page 18) [RET{ INSERT

```
CV5: BIT 3,C
     JR NZ,CV3
     JR CV4
```

For anyone who is planning to enter the same code, two errors have appeared in the listing due to limited printer width.

i. The last byte of line 2044 (page 35) should read 36H (not 3).
ii. LINE 2045 (page 35) should read 36H (not 3) as the last byte.

Note also that the program origin should be of the form XX00 (hex) ie the least significant digit must be 0.

B. TANNER
WR14 1HW

Dear Sir,

For Sharpsoft User Notes:–

1. One very useful routine for use in BASIC programs is:–

```
100  PRINT "CONTINUE (Y/N)?"
110  GET A$: IF A$=" " GOTO 110
120  IF ASC(A$) = 89 THEN 1000
130  IF ASC(A$) = 78 THEN END
140  GOTO 1000
1000 Rest of program etc. . . .
```

This saves the annoying need to press CR after pressing Y/N etc.

2. Is it possible to make the SHARP S100 compatible?
3. Are SHARP likely to bring out a Disk controller for 8" drives, or can the existing one be modified reasonably easily?

A.L. VALE
BN2 3HY

Mr. Vale,

To our knowledge it is not at present possible to make an MZ-80K S100 compatible. However if any one has any ideas on this one we would like to know.

Crystal Electronics have been working on the possibility of putting up an 8" drive under the CP†M© operating systems. But I assure you that as soon as an 8" drive is commercially available we will be in touch with a newsletter. Ed.

Dear Sir,

Herewith a program with explanatory notes which you might like to publish in the SHARPSOFT User Notes.

I would very much like to know how to append BASIC programs on the MZ-80K using SP-5025.

BASIC with machine code routines

PROGRAM Z1

When BASIC SP-5025 is loaded, it automatically checks all free memory locations from 17408 (4400H) to 53248 (D000H) by loading each with FFH and then with 00. It thus verifies that all bits are operative and also determines the top of available memory, but in so doing, it destroys the contents of these locations. To prevent this destruction of contents it is possible to limit the extent of the test sweep. This is essential if machine code programs have previously been loaded into memory and are to be called from BASIC programs by using the USR command.

To prevent overwriting stored programs select a suitable upper limit of the test sweep (say 40K=40960=A000H). The upper byte (160=A0H) must be 'poked' into location 4619 and a new copy of BASIC made. The following small program achieves this:

```
10  POKE 4619,160
20  USR(33):USR(36)
```

Record the copy of BASIC on a new tape (wind past the leader first and set the counter to 0); stop the tape manually at about 35 as there is no automatic termination. When this copy of basic is loaded memory locations above 40960 will NOT be tested or changed.

When incorporating machine code routines in BASIC programs, it proves convenient to have a utility program which enables one to examine and modify the contents of memory. The following program offers these facilities; the subroutines at 1000 and 1100 illustrate ways of converting between hexadecimal and decimal numbers.

C.F. McD. WOOD
BN2 4GJ

```
0090 PRINT"■":LIST 100-299
0100 REM ****************************
0110 REM *                          *
0120 REM * Program to modify or     *
0130 REM * list contents of stores  *
0140 REM * Start address can be     *
0150 REM * entered in either        *
0160 REM * DEC or HEX (eg 0C5AH)    *
0170 REM * Press any key for 4 more *
0180 REM * lines of listing         *
0190 REM * Enter "=" for no change  *
0200 REM * Enter "^" for backspace  *
0210 REM * Enter any non-HEX code   *
0220 REM * to quit modification     *
0225 REM * Press BREAK to end       *
0230 REM * Author                   *
0240 REM *   C.F.McD. Wood          *
0250 REM *   Brighton Polytechnic   *
0260 REM *   July 1981              *
0270 REM *                          *
0280 REM ****************************
0290 REM
0292 REM To run, enter  10 GOTO 300
0294 REM     then enter  RUN
0300 POKE 16167,1:REM unprotects memory
0310 CH$="0123456789ABCDEF":REM HEX chs
0320 PRINT "■"
0330 PRINT"Is start address of listing"
0340 INPUT"given in HEX or DEC: H/D? ";R$
0350 IF LEFT$(R$,1)="H" THEN 400
0360 IF LEFT$(R$,1)="D" THEN 450
0370 PRINT "■Response not clear■"
0380 GOTO 330
0400 INPUT"■Start address in HEX = ";H$
0410 GOSUB 1000
0420 AD=D:GOTO 500
0450 INPUT"■Start address in DEC = ";D
0460 AD=D:M=4096:GOSUB 1100
0500 INPUT "■Tabulate or modify stores:T/M? ";R$
0510 IF LEFT$(R$,1)="T" THEN 600
0520 IF LEFT$(R$,1)="M" THEN 800
0530 PRINT "■Response not clear■":GOTO 500
0600 REM Tabulation of stores
0610 PRINT"■■    Address"
0620 PRINT" DEC      HEX            Data(H)■"
0630 PS=8*INT(AD/8):REM modulo 8
```

```
0640 PN=PS:REM stores start address
0650 FOR K=0 TO 3:REM sets of 4 lines
0660 PL=PN+8*K:REM 8 items per line
0670 D=PL:M=4096:GOSUB 1100:A$=H$
0680 LP$=" "
0690 FOR J=0 TO 7
0700 D=PEEK(PL+J):REM set data in store
0710 M=16:GOSUB 1100
0720 LP$=LP$+H$:NEXT J
0730 PRINT PL;TAB(7);A$;LP$
0740 NEXT K: PRINT
0750 GET A$:IF A$="" THEN 750
0760 PN=PN+8*K:GOTO 650
0800 REM Modification of stores
0810 PRINT"    Address          Data (H)"
0820 PRINT" DEC       HEX     old  new "
0830 PN=D:REM Initialises position
0840 D=PN:M=4096:GOSUB 1100:A$=H$
0850 D=PEEK(PN):M=16:GOSUB 1100
0860 LP$="   "+H$
0865 IF L>2 THEN L=2
0870 PRINT PN;TAB(8);A$;LP$;
0880 INPUT R$:L=LEN(R$)
0890 IF L>2 THEN L=2
0900 Z=0:FOR J=1 TO L
0910 IF R$="=" THEN 990:REM no change
0920 REM Enter  or P to step back one
0930 IF (R$="")+(R$="P") THEN PN=PN-1:GOTO 840
0940 X=ASC(MID$(R$,J,1))
0950 IF (X<48)+(X>57)*(X<65)+(X>70) THEN STOP
0960 IF (X>57) THEN X=X-7
0970 Z=16*Z+(X-48):NEXT J
0980 POKE PN,Z
0990 PN=PN+1:GOTO 840
1000 REM HEX(H$) to DEC(D) conversion
1010 ER=0:D=0:M=16:L=LEN(H$):J=0
1020 J=J+1:IF J>L THEN RETURN
1030 X=ASC(MID$(H$,J,1))
1035 IF (X=32) THEN 1020:REM skip space
1040 IF (X<48)+(X>57)*(X<65) THEN ER=1:GOTO 1020
1050 IF (X>70) THEN RETURN
1060 IF (X>57) THEN X=X-7
1070 D=D*M+(X-48):GOTO 1020
1100 REM DEC(D) to HEX(H$) conversion
1110 REM enter with M=4096 for 16 bit
```

```
1120 REM    or with M=16   for  8 bit
1130 M1=M:H$="":X=D
1140 Z=INT(X/M1):H$=H$+MID$(CH$,Z+1,1)
1150 X=X-Z*M1:M1=M1/16
1160 IF M1>0.5 THEN 1140
1170 IF M>16 THEN 1190
1180 H$=H$+" ":RETURN
1190 H$=H$+"H ":RETURN
```

Mr. McD. Wood,
The SPEED BASIC has an Append command. So has the Ardensoft tool pack.
Ed.

Dear Sir,

Thank you for the latest issue of 'User Notes'. It was very refreshing to find software being given away more or less free in such large chunks! I hope your business manages to survive it.

I was particularly interested in the 'Tiny Pilot' listing. I have had an even tinier version of the language, culled from a magazine, for some time, but it was really too small to be useful. As you suggested in the preamble to the listing, there were one or two minor bugs. I have noted below those which I have found, together with one or two other modifications which readers may find useful.

1. Adding ':LS=LEN(P$(M))' to the end of line 7160 corrects the truncation of the line which can occur when the string to be replaced is found more than once in the same line.
2. If you try to replace (for instance) 'P' with 'PP' using the 'Find' function, you tend to end up with a long string of P's! Change 'LC=LC+1' to 'LC=LC+LN−LO+1' in line 7170 to correct this.
3. In line 13110 '(U$(I)' should be replaced by 'LEFT$(U$(1)(,2)'. This prevents a variable being used twice in the same program. This can cause problems in some program situations however. For instance, if you use a counting loop to allow the user, say, three attempts at a question before the program takes action, this would be seen as an attempt to duplicate the use of a variable. I have changed line 13110 to:

   13110 IFD$=LEFT$(U$(1),2)THENU$(I)=D$+E$:RETURN

   This relplaces the existing value of the variable with the new value, just as in a 'BASIC' program.
4. Line 20110 lacks a 'NEXT' statement.
5. A slightly inconvenient feature of the interpreter was that no easy way existed of extending a partly written program. The command 'N' wipes out the existing program, while 'I' only allows one line at a time to be inserted. I have therefore added a new command, 'W' (=write) which

permits the addition of new lines to the end of the existing program. The new command requires only two additional lines of code as follows:

```
360  IFC$="W"THEN1700
1700 P=P1+-1*(P1=0):P$(P)="":GOTO1050
```

The '+-1*(P1=0) in this line is important as without it use of the 'W' command to write a program from the beginning will cause the first line to be numbered '0' instead of '1'. The new command should also be added to menu in some convenient position.

6. Finally, I encountered a few problems with the subroutine at 11000–11320 which handles the 'T:' statement. I have in face rewritten this subroutine and the code is given in the accompanying listing. As well as removing the bugs (and probably introducing some others which I haven't found yet!) the new code is shorter and runs rather faster.

I hope these comments have been of interest. If you feel that they will be useful to other readers' please feel free to edit them as you think necessary.

P.L. BIRCH
PA2 6AW

```
11000 IF LEFT$(D$,2)="!H"THENPRINT"@":D$=RIGHT$(D$,LEN(D$)-2)
11010 I=1
11020 IFMID$(D$,I,1)="$"THENGOSUB11100
11030 IFI<LEN(D$)THENI=I+1:GOTO11020
11040 PRINTD$
11050 RETURN
11100 FORJ=1TOV1
11110 IFLEFT$(V$(J),1)=MID$(D$,I-1,1)THEN11130
11120 NEXTJ:PRINT"Undefined variable, LINE";P:RETURN
11130 D$=LEFT$(D$,I-2)+RIGHT$(V$(J),LEN(V$(J))-2)+RIGHT$(D$,LEN(D$)-I)
11140 I=I+LEN(V$(J))-4
11150 RETURN
```

Dear Sir,

As a Research Chemist, I am a relative newcomer to Computer Programming, I first became involved with computers, in a personal sense, when in December 1980 it was decided to computerise our aquired experimental data. As might be expected this involved several feasibility studies to decide on the format of the data file required, and the means of inputing data into it. The computer on site is a VAX-11/VMS with multiple terminal input/output facilities, and is capable of using Comand Language, Basic, Fortran, Pascal, and Cobal, as well as functioning as a Word Processor.

During December and January, I obtained several books on "Basic", and after a short period of study of these, and the VAX/VMS Basic Manual, I devised in Basic a program to input our accumulated data into a specially formated data file, which, with the VAX library "Search" routine would enable selected data to be readily retrieved.

This exercise fired my enthusiasm for computer programming, and I wrote several more short programs for the VAX-11 to aid us in our day to day work in the laboratory.

Then in ate April/81, I acquired a MZ-80K (36k) computer for my own use. After a short period of familiarisation with the computer I decided to try my hand at writing a short game program in order to learn more about the special features and capabilities of the MZ-80K.

I decided first to devise a computerised version of the American dice game "CRAPS", with a changing dice display. I felt this would not be too difficult and would help me to get used to the graphics routines.

I first drew up a flow diagram, as shown in Diagram 1, and then from this constructed the program shown.

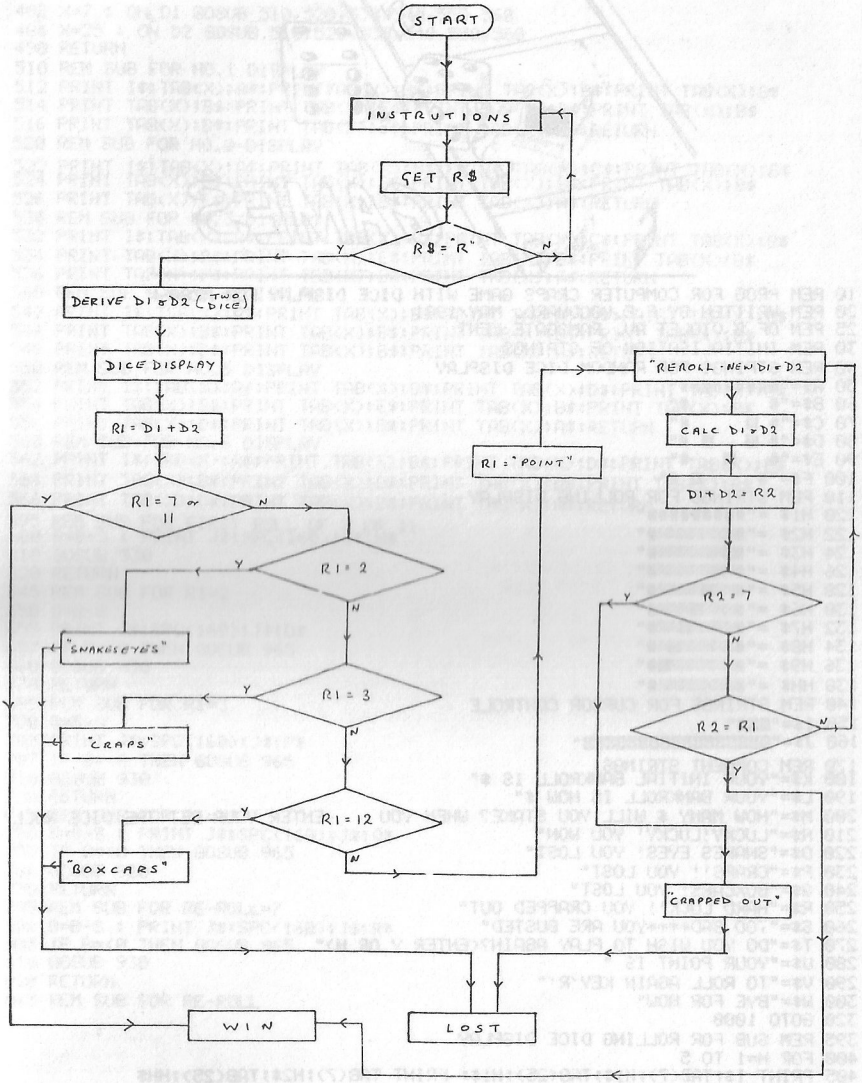
I then felt I would like to experiment with graphics using "POKE" commands, and for this purpose decided to write a short program for a "Roulette Wheel", which would have a "bali" starting from the random number, revolving round the wheel a few times, and finishing on another random number. This resulted in the second program shown.

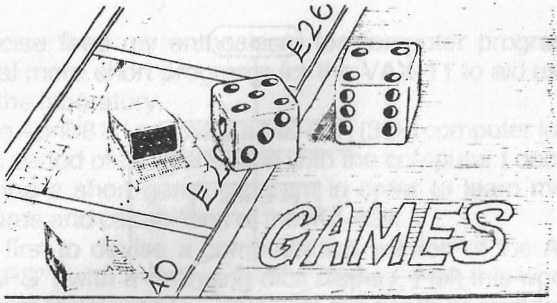
By this time (June/81) I felt ready to try something a little more ambitious, and decided to write a program for a game of "PONTOON", with a changing card display, and the computer programmed to play it's cards according to the rules applying to dealers at a casino.

After some effort this resulted in the flow charts shown in Diagrams 2 and 3. Then from these the program shown was written.

Obviously I make no claim to these programs being in any way perfect, or even approaching the best way of carrying out the ideas aimed at. However, I hope the above notes, and programs will be of interest to "SHARPSOFT" readers, and will give some indication of what can be achieved after only a relatively short period of experience in computer programming.

F.E.. WOODWARD
8 Violet Avenue,
Ramsgate

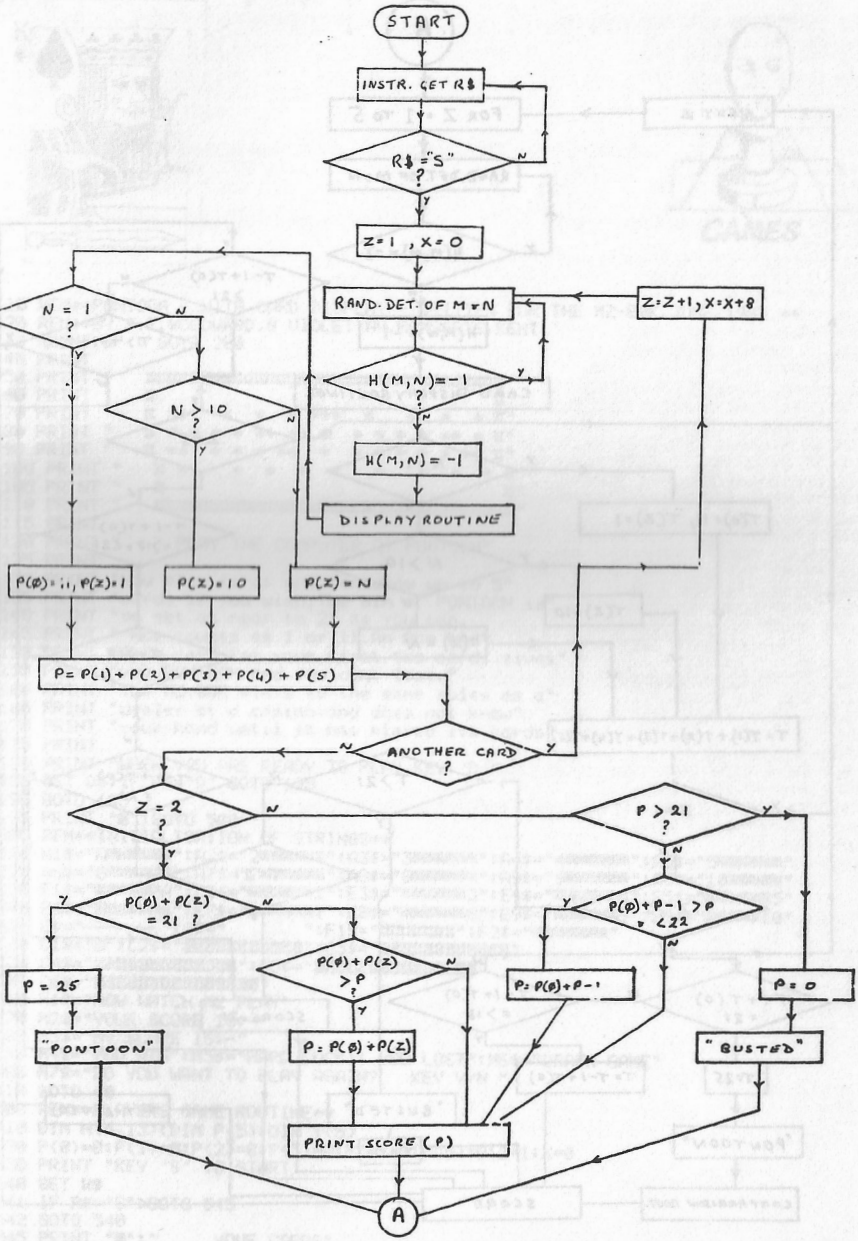

START
INSTRUCTIONS
GET R$
R$ = "R"
Y
N
DERIVE D1 & D2 (TWO DICE)
DICE DISPLAY
R1 = D1 + D2
R1 = 7 or 11
Y
N
R1 = 2
Y
N
"SNAKEEYES"
R1 = 3
Y
N
"CRAPS"
R1 = 12
Y
N
"BOXCARS"
WIN
R1 = "POINT"
"REROLL - NEW D1 + D2
CALC D1 + D2
D1 + D2 = R2
R2 = 7
Y
N
R2 = R1
N
Y
"CRAPPED OUT"
LOST

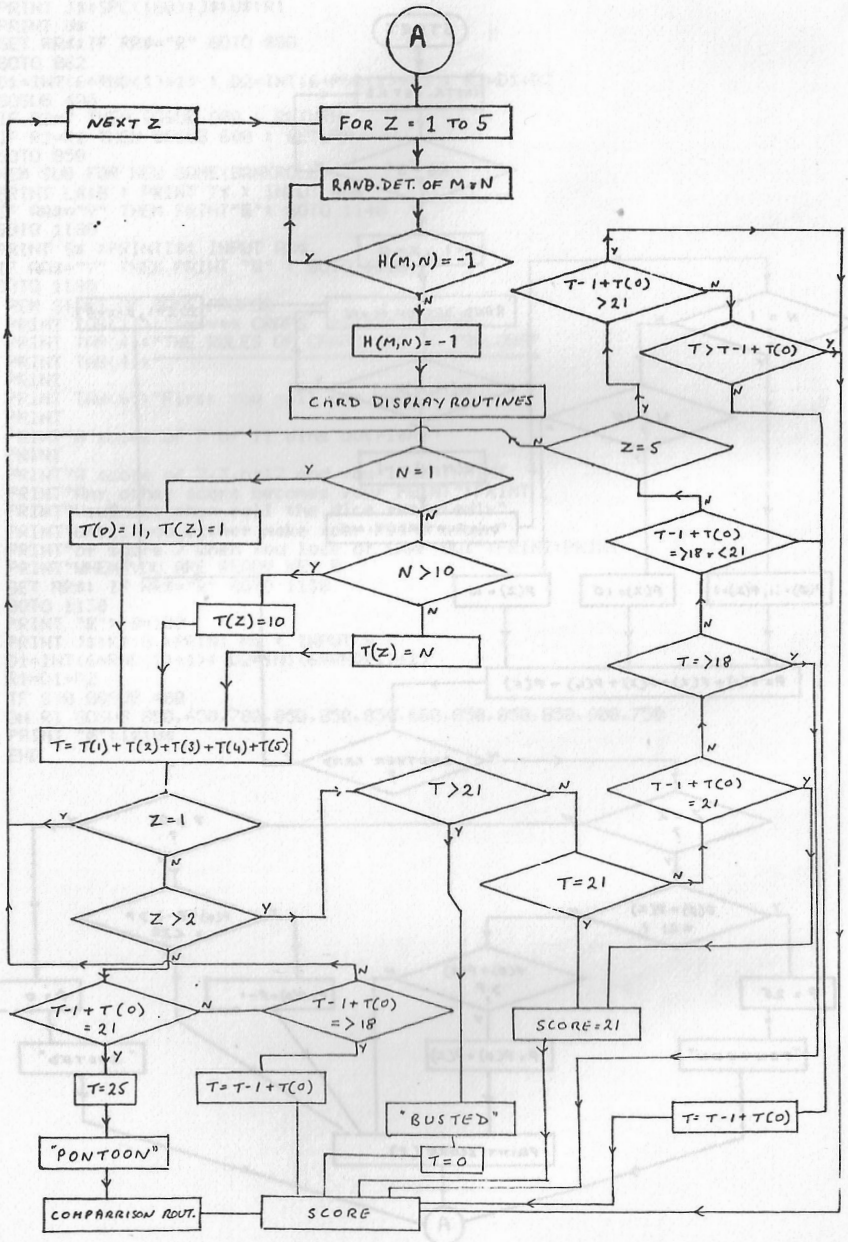


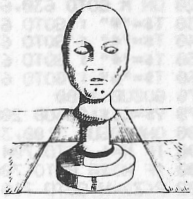

```
10 REM PROG FOR COMPUTER CRAPS GAME WITH DICE DISPLAY FOR MZ80-K
20 REM WPITTEN BY F.E.WOODWARD. MAY/1981
25 REM OF 8 VIOLET AV. RAMSGATE KENT
30 REM INITIALISATION OF STRINGS
40 REM STPINGS FOR RANDOM DICE DISPLAY
50 A$="##########"
60 B$="#        #"
70 C$="# ■      #"
80 D$="# ■   ■ #"
90 E$="#   ■    #"
100 F$="#     ■ #"
110 REM STRINGS FOR ROLLING DISPLAY
120 H1$ ="##########"
122 H2$ ="#▒▒▒▒▒▒▒▒#"
124 H3$ ="#■▒▒▒▒▒▒▒#"
126 H4$ ="#▒■▒▒▒▒▒▒#"
128 H5$ ="#▒▒■▒▒▒▒▒#"
130 H6$ ="#▒▒■▒■▒▒▒#"
132 H7$ ="#▒▒▒▒■▒▒▒#"
134 H8$ ="#▒▒▒▒■▒■▒#"
136 H9$ ="#▒▒▒▒■▒■■#"
138 HH$ ="#▒▒▒▒▒▒▒▒#"
140 REM STRINGS FOR CURSOR CONTROLE
150 I$="■■■"
160 J$="■■■■■■■■■■■■■■■■■■■"
170 REM COMMENT STRINGS
180 K$="YOUR INITIAL BANKROLL IS $"
190 L$="YOUR BANKROLL IS NOW $"
200 M$="HOW MANY $ WILL YOU STAKE? WHEN YOU     ENTER YOUR BET THE DICE ROLL"
210 N$="LUCKY!LUCKY! YOU WON"
220 O$="SNAKES EYES! YOU LOST"
230 P$="CRAPS!! YOU LOST"
240 Q$="BOXCARS! YOU LOST"
250 R$="HARD LUCK!! YOU CRAPPED OUT"
260 S$="TOO BAD****YOU ARE BUSTED"
270 T$="DO YOU WISH TO PLAY AGAIN?(ENTER Y OR N)"
280 U$="YOUR POINT IS "
290 V$="TO ROLL AGAIN KEY'R'"
300 W$="BYE FOR NOW"
320 GOTO 1000
395 REM SUB FOR ROLLING DICE DISPLAY
400 FOR N=1 TO 5
405 PRINT I$;TAB(7);H1$;TAB(25);H1$: PRINT TAB(7);H2$;TAB(25);HH$
415 PRINT TAB(7);H3$;TAB(25);H9$ : PRINT TAB(7);H4$;TAB(25);H8$
420 PRINT TAB(7);H5$;TAB(25);H7$ : PRINT TAB(7);H6$;TAB(25);H6$
425 PRINT TAB(7);H7$;TAB(25);H5$ : PRINT TAB(7);H8$;TAB(25);H4$
430 PRINT TAB(7);H9$;TAB(25);H3$ : PRINT TAB(7);HH$;TAB(25);H2$
435 PRINT TAB(7);H1$;TAB(25);H1$
440 PRINT I$;TAB(7);H1$;TAB(25);H1$: PRINT TAB(7);HH$;TAB(25);H2$
445 PRINT TAB(7);H9$;TAB(25);H3$ : PRINT TAB(7);H8$;TAB(25);H4$
450 PRINT TAB(7);H7$;TAB(25);H5$ : PRINT TAB(7);H6$;TAB(25);H6$
455 PRINT TAB(7);H5$;TAB(25);H7$ : PRINT TAB(7);H4$;TAB(25);H8$
460 PRINT TAB(7);H3$;TAB(25);H9$ : PRINT TAB(7);H2$;TAB(25);HH$
465 PRINT TAB(7);H1$;TAB(25);H1$
480 NEXT N
```

```
482 X=7 : ON D1 GOSUB 510,520,530,540,550,560
484 X=25 : ON D2 GOSUB 510,520,530,540,550,560
490 RETURN
510 REM SUB FOR NO.1 DISPLAY
512 PRINT I$;TAB(X);A$:PRINTTAB(X);B$:PRINT TAB(X);B$:PRINT TAB(X);B$
514 PRINT TAB(X);B$:PRINT TAB(X);E$:PRINT TAB(X);B$:PRINT TAB(X);B$
516 PRINT TAB(X);B$:PRINT TAB(X);B$:PRINT TAB(X);A$:RETURN
520 REM SUB FOR NO.2 DISPLAY
522 PRINT I$;TAB(X);A$:PRINT TAB(X);B$:PRINT TAB(X);C$:PRINT TAB(X);B$
524 PRINT TAB(X);B$:PRINT TAB(X);B$:PRINT TAB(X);B$:PRINT TAB(X);B$
526 PRINT TAB(X);F$:PRINT TAB(X);B$:PRINT TAB(X);A$:RETURN
530 REM SUB FOR NO.3 DISPLAY
532 PRINT I$;TAB(X);A$:PRINT TAB(X);B$:PRINT TAB(X);C$:PRINT TAB(X);B$
534 PRINT TAB(X);B$:PRINT TAB(X);E$:PRINT TAB(X);B$:PRINT TAB(X);B$
536 PRINT TAB(X);F$:PRINT TAB(X);B$:PRINT TAB(X);A$:RETURN
540 REM SUB FOR NO.4 DISPLAY
542 PRINT I$;TAB(X);A$:PRINT TAB(X);B$:PRINT TAB(X);D$:PRINT TAB(X);B$
544 PRINT TAB(X);B$:PRINT TAB(X);B$:PRINT TAB(X);B$:PRINT TAB(X);B$
546 PRINT TAB(X);D$:PRINT TAB(X);B$:PRINT TAB(X);A$:RETURN
550 REM SUB FOR NO.5 DISPLAY
552 PRINT I$;TAB(X);A$:PRINT TAB(X);B$:PRINT TAB(X);D$:PRINT TAB(X);B$
554 PRINT TAB(X);B$:PRINT TAB(X);E$:PRINT TAB(X);B$:PRINT TAB(X);B$
556 PRINT TAB(X);D$:PRINT TAB(X);B$:PRINT TAB(X);A$:RETURN
560 REM SUB FOR NO.6 DISPLAY
562 PRINT I$;TAB(X);A$:PRINT TAB(X);B$:PRINT TAB(X);D$:PRINT TAB(X);B$
564 PRINT TAB(X);B$:PRINT TAB(X);D$:PRINT TAB(X);B$:PRINT TAB(X);B$
566 PRINT TAB(X);D$:PRINT TAB(X);B$:PRINT TAB(X);A$:RETURN
595 REM SUB FOR FIRST ROLL OF 7 OR 11
600 B=B+S : PRINT J$;SPC(160);J$;N$
610 GOSUB 930
620 RETURN
645 REM SUB FOR R1=2
650 B=B-S
655 PRINT J$;SPC(160);J$;O$
657 IF B=<0 THEN GOSUB 965
660 GOSUB 930
670 RETURN
695 REM SUB FOR R1=3
700 B=B-S
705 PRINT J$;SPC(160);J$;P$
707 IF B=<0 THEN GOSUB 965
710 GOSUB 930
720 RETURN
745 REM SUB FOR R1=12
750 B=B-S : PRINT J$;SPC(160);J$;O$
755 IF B=<0 THEN GOSUB 965
760 GOSUB 930
770 RETURN
795 REM SUB FOR RE-ROLL=7
800 B=B-S : PRINT J$;SPC(160);J$;R$
805 IF B=<0 THEN GOSUB 965
810 GOSUB 930
820 RETURN
845 REM SUB FOR RE-ROLL
```

```
850 PRINT J$;SPC(160);J$;U$;R1
860 PRINT V$
862 GET RR$:IF RR$="R" GOTO 880
864 GOTO 862
880 D1=INT(6*RND(1)+1) : D2=INT(6*RND(1)+1) : R2=D1+D2
885 GOSUB 400
900 IF R2=7 THEN GOSUB 800 : RETURN
910 IF R2=R1 THEN GOSUB 600 : RETURN
920 GOTO 850
925 REM SUB FOR NEW GAME(BANKROLL)
930 PRINT L$;B : PRINT T$ : INPUT AA$
940 IF AA$="Y" THEN PRINT"■": GOTO 1140
950 GOTO 1180
965 PRINT S$ :PRINTT$: INPUT AA$
970 IF AA$="Y" THEN PRINT "■" : GOTO 1120
980 GOTO 1180
1000 REM START OF GAME PROPER
1010 PRINT TAB(11);"****** CRAPS  ******":PRINT
1020 PRINT TAB(4);"THE RULES OF CRAPS ARE AS FOLLOWS"
1030 PRINT TAB(4);"──────────────────────────────────────"
1040 PRINT
1050 PRINT TAB(6);"First you roll two dice."
1055 PRINT
1060 PRINT"A score of 7 or 11 wins outright"
1065 PRINT
1070 PRINT"A score of 2,3,or12 and you lose":PRINT
1080 PRINT"Any other score becomes your POINT":PRINT
1090 PRINT"You■must then roll the dice repeatedly"
1100 PRINT"until you either make your POINT again"
1110 PRINT"or score 7 when you lose or CRAP OUT":PRINT:PRINT
1120 PRINT"WHEN YOU ARE READY KEY R
1130 GET RR$: IF RR$="R" GOTO 1138
1135 GOTO 1130
1138 PRINT "■": B=100
1140 PRINT J$;K$;B :PRINT M$ : INPUT S
1150 D1=INT(6*RND(1)+1): D2=INT(6*RND(1)+1)
1155 R1=D1+D2
1160 IF S>0 GOSUB 400
1170 ON R1 GOSUB 850,650,700,850,850,850,600,850,850,850,600,750
1180 PRINT "■";I$;W$
1190 END
```


START
INSTR. GET R$
R$ = "S" ?
N
Y
Z=1 , X=0
RAND. DET. OF M≠N
Z=Z+1, X=X+8
H(M,N)=-1 ?
Y
N
H(M,N) = -1
DISPLAY ROUTINE
N = 1 ?
N
Y
N > 10 ?
N
Y
P(Ø) = 11, P(Z)=1
P(Z)=10
P(Z) = N
P= P(1) + P(2) + P(3) + P(4) + P(5)
ANOTHER CARD ?
N
Y
Z = 2 ?
N
Y
P > 21 ?
Y
N
P(Ø) + P(Z) = 21 ?
Y
N
P(Ø) + P-1 > P = <22
Y
N
P(Ø) + P(Z) > P ?
N
Y
P = 25
P= P(Ø)+P-1
P = 0
"PONTOON"
P = P(Ø) + P(Z)
"BUSTED"
PRINT SCORE (P)
A


A
FOR Z = 1 TO 5
NEXT Z
RAND. DET. OF M & N
H(M,N) = -1
H(M,N) = -1
CARD DISPLAY ROUTINES
N = 1
T(0) = 11, T(Z) = 1
N > 10
T(Z) = 10
T(Z) = N
T = T(1) + T(2) + T(3) + T(4) + T(5)
Z = 1
Z > 2
T > 21
T = 21
T - 1 + T(0) = 21
T - 1 + T(0) => 18
T = 25
"PONTOON"
COMPARRISON ROUT.
T = T - 1 + T(0)
"BUSTED"
T = 0
SCORE
SCORE = 21
T - 1 + T(0) > 21
T > T - 1 + T(0)
Z = 5
T - 1 + T(0) => 18 & < 21
T => 18
T - 1 + T(0) = 21
T = T - 1 + T(0)



GAMES

```
10 REM**PONTOON - WITH CARD DISPLAY - WRITTEN FOR THE MZ-80K.JUNE/1981 **
20 REM**BY F.E.WOODWARD.8 VIOLET AV.RAMSGATE.KENT
30 PRINT"■" : GOTO 200
40 PRINT
50 PRINT "   ##########################################"
60 PRINT "   #                                        #"
70 PRINT "   # **   *  *  * ***  *   *  *  * #"
80 PRINT "   # * * * * ** *  *  * * * * ** * #"
90 PRINT "   # **  * * * **  *  * * * * * ** #"
100 PRINT "   # *     *  *  *  *   *  *  * #"
105 PRINT "   #                                        #"
110 PRINT "   ##########################################"
115 PRINT
120 PRINT "      PLAY THE COMPUTER AT PONTOON"
125 PRINT
130 PRINT "You play first and can draw up to 5"
135 PRINT "cards if you wish.The aim of PONTOON is"
140 PRINT "to get as near to 21 as you can.
145 PRINT " Ace counts as 1 or 11.An Ace and"
150 PRINT "Face card for your first two cards gives"
155 PRINT "you PONTOON,and you cant lose."
160 PRINT "The MZ-80K plays to the same rules as a"
165 PRINT "dealer at a casino,and does not know"
170 PRINT "your hand until it has played its cards"
175 PRINT
180 PRINT "WHEN YOU ARE READY TO PLAY KEY 'D'"
185 GET D$:IF D$="D" GOTO 195
190 GOTO 185
195 PRINT "■":GOTO 500
200 REM**INITIALISATION OF STRINGS**
210 A1$="A■■■■■■■":A2$="2■■■■■■■":A3$="3■■■■■■■":A4$="4■■■■■■■":A5$="5■■■■■■■"
220 A6$="6■■■■■■■":A7$="7■■■■■■■":A8$="8■■■■■■■":A9$="9■■■■■■■":A0$="10■■■■■■"
230 E1$="■■■■■■■A":E2$="■■■■■■■2":E3$="■■■■■■■3":E4$="■■■■■■■4":E5$="■■■■■■■5"
240 E6$="■■■■■■■6":E7$="■■■■■■■7":E8$="■■■■■■■8":E9$="■■■■■■■9":E0$="■■■■■■10"
320 E$="■■■■■■■■■":S$="         ":F1$="##########":F2$="■■■■■■■■■"
330 C1$="■":C2$="■■■■■■■■■■■■■":C3$="■■■■■■■■■■■■■"
335 C4$="■■■■■■■■■■■■■■■":C5$="■■■■■■■■■■■■■■■■■"
340 C6$="■■■■■■■■■■■■■■■■■"
360 M1$="NOW WATCH ME PLAY"
370 M2$="YOUR SCORE IS:-"
380 M3$="MY SCORE IS:-"
390 M4$="YOU WIN":M5$="HARD LUCK!! YOU LOST":M6$="DRAWN GAME"
400 M7$="DO YOU WANT TO PLAY AGAIN?   KEY Y/N "
410 GOTO 40
500 REM**PLAYERS GAME ROUTINE**
510 DIM H(4,13):DIM P(5):DIM T(5)
520 P(0)=0:P(1)=0:P(2)=0:P(3)=0:P(4)=0:P(5)=0:Z=1:X=0
530 PRINT "KEY 'S' TO START
540 GET R$
541 IF R$="S" GOTO 545
542 GOTO 540
545 PRINT "■";"     YOUR CARDS"
550 M=INT(4*RND(1))+1 : N=INT(13*RND(1))+1
600 IF H(M,N)=-1 GOTO 550
610 H(M,N)=-1
```

```
620 ON M GOTO 630,640,650,660,
630 T$="♥" : GOTO 670
640 T$="♠" : GOTO 670
650 T$="♦" : GOTO 670
660 T$="♣" : GOTO 670
670 GOSUB 1700
678 Y$=C1$:GOSUB 2000
680 ONNGOSUB 3100,3200,3300,3400,3500,3600,3700,3800,3900,4000,4100,4200,4300
690 IF N=1 GOTO 704
700 IF N>10 GOTO 706
702 P(Z)=N:GOTO 710
704 P(0)=11:P(Z)=1:GOTO 710
706 P(Z)=10
710 P=P(1)+P(2)+P(3)+P(4)+P(5)
715 IF P>21 GOTO 860
720 PRINT C2$;"DO YOU WANT ANOTHER CARD?  Y/N
730 GET A$
740 IF A$="Y" GOTO 770
750 IF A$="N" GOTO 765
760 GOTO 730
765 PRINT C2$;SPC(35):GOTO 780
770 Z=Z+1:X=X+8:PRINT C2$:SPC(35):GOTO 550
780 IF Z>2 GOTO 810
790 IF P-1+P(0)=21 THEN P=25:GOTO 900
810 IF P-1+P(0)=21 THEN P=21:GOTO 1000
820 IFP>P-1+P(0) GOTO 1000
830 IF (P-1+P(0))18)*(P-1+P(0)<22) THEN P=P-1+P(0):GOTO 1000
840 GOTO 1000
850 REM**MESSAGES AND SCORES FOR PLAYER**
860 PRINT C2$;"HARD LUCK!! YOU HAVE BUSTED" :P=0
870 PRINT "STILL,I MIGHT DO THE SAME"
890 PRINT C4$;M1$ : GOTO 1020
900 PRINT C2$;"P0NTOON!!!":PRINT M1$:GOTO 1020
1000 PRINT C2$;M2$;P:PRINT M1$:GOTO1020
1010 REM**COMPUTER GAME ROUTINE**
1020 X=0:T(0)=0:T(1)=0:T(2)=0:T(3)=0:T(4)=0:T(5)=0:Y$=C6$:Z=0
1025 PRINT C5$;"MY CARDS"
1030 FOR Z=1 TO 5
1040 M=INT(4*RND(1))+1:N=INT(13*RND(1))+1
1050 IF  H(M,N)=-1 GOTO 1040
1055 H(M,N)=-1
1060 ON M GOTO 1070,1080,1090,1100
1070 T$="♥":GOTO 1110
1080 T$="♠":GOTO 1110
1090 T$="♦":GOTO 1110
1095 USR(62)
1100 T$="♣":GOTO 1110
1110 GOSUB 1700
1115 Y$=C6$:GOSUB 2000
1120 ONNGOSUB 3100,3200,3300,3400,3500,3600,3700,3800,3900,4000,4100,4200,4300
1130 IF N=1 GOTO1145
1135 IF N>10 GOTO1150
1140 T(Z)=N:GOTO1155
1145 T(0)=11:T(Z)=1:GOTO1155
1150 T(Z)=10
```

```
1155 T=T(1)+T(2)+T(3)+T(4)+T(5):TT=T-1+T(0)
1160 IF Z=2 GOTO 1220
1170 IF T>21 GOTO 1350
1180 IF T=>18 GOTO 1340
1190 IF (TT=>18)*(TT<22) GOTO 1320
1200 IF Z=5 GOTO 1260
1210 X=X+8:NEXT Z
1220 IF TT=21 GOTO 1330
1230 IF TT=>18 GOTO 1320
1240 IF T=>18 GOTO 1340
1250 GOTO1210
1260 IF (TT=>18)*(TT<22) GOTO 1320
1270 GOTO 1330
1320 T=TT:PRINT C5$;TAB(17);M3$;TT :GOTO 1360
1340 PRINT C5$;TAB(17);M3$;T:GOTO 1370
1350 PRINT C5$;TAB(17);"I HAVE BUST":T=0:GOTO 1370
1360 REM**COMPARRISON OF SCORES**
1370 PRINT C3$;SPC(40)
1380 IF P=T GOTO 1410
1390 IF P>T GOTO 1420
1400 IF P<T GOTO 1430
1410 PRINT C4$;M6$ :PRINT M7$:GOTO1450
1420 PRINT C4$;M4$ :PRINT M7$:GOTO1450
1430 PRINT C4$;M5$ :PRINT M7$:GOTO1450
1450 GET A$
1460 IF A$="Y" GOTO 1490
1470 IF A$="N" GOTO 1500
1480 GOTO 1450
1490 PRINT "E" : GOTO 500
1500 PRINT "E":"SO LONG FOR NOW.I HOPE YOU ENJOYED THE'
1510 PRINT "GAME.PLAY AGAIN SOON"
1520 END
1700 REM**SUB FOR SUIT STRINGS**
1710 B$="▒"+T$+"▒▒▒"+T$+"▒"
1715 C$="▒▒▒"+T$+"▒▒▒"
1720 D$="▒▒▒▒▒▒▒▒▒"
1730 P$="▒"+T$+T$+"▒▒▒▒▒":P0$="▒"+T$+"▒"+T$+"▒▒▒▒▒":P1$="▒"+T$+"▒▒▒▒▒"+T$+"▒"
1740 P2$="▒▒▒▒▒"+T$+"▒▒▒"
1750 P3$="▒▒▒"+T$+T$+T$+"▒▒▒"
1760 P4$="▒"+T$+T$+T$+T$+T$+"▒"
1780 P5$="▒"+T$+"▒▒▒"+T$+"▒▒▒":P6$="▒▒▒"+T$+T$+"▒▒▒▒▒":P7$="▒"+T$+"▒"+T$+"▒"+T$+"▒"
1790 P8$="▒▒▒▒▒▒▒"+T$+"▒"
1800 RETURN
1880 P5$="▒"+T$+"▒▒▒"+T$+"▒▒▒":P6$="▒▒▒"+T$+T$+"▒▒▒▒▒":P7$="▒"+T$+"▒"+T$+"▒"+T$+"▒"
2000 REM**SPINNING CARD ROUTINE**
2010 FOR A=1 TO 2
2020 X$=F1$:GOSUB 2070
2030 X$=F2$:GOSUB 2070
2040 NEXT A
2050 RETURN
2060 REM**SPINNING CARD DISPLAY**
2070 PRINT V$:FOR I=1 TO 4:PRINT:NEXT I:PRINT TAB(X);E$
2080 PRINT V$:FOR I=1 TO 4:PRINT:NEXT I:PRINT TAB(X);X$
2090 PRINT V$:FOR I=1 TO 3:PRINT:NEXT I:FOR I=1 TO 3:PRINT TAB(X);X$:NEXT I
2100 PRINT V$:FOR I=1 TO 2:PRINT:NEXT I:FOR I=1 TO 5:PRINT TAB(X);X$:NEXT I
2110 PRINT V$:PRINT:FOR I=1 TO 7:PRINT TAB(X);X$:NEXT I
```

```
2120 PRINT Y$:FOR I=1 TO 9:PRINT TAB(X);X$:NEXT I
2130 PRINT Y$:PRINT TAB(X);S$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);S$
2140 PRINT Y$:FOR I=1 TO 2:PRINT TAB(X);S$:NEXT I:FOR I=1 TO 5:PRINT:NEXT I
2150 FOR I=1 TO 2:PRINT TAB(X);S$:NEXT I
2160 PRINT Y$:FOR I=1 TO 3:PRINT TAB(X);S$:NEXT I:FOR I=1 TO 3:PRINT:NEXT I
2170 FOR I=1 TO 3:PRINT TAB(X);S$:NEXT I
2180 PRINT Y$:FOR I=1 TO 4:PRINT TAB(X);S$:NEXT I:PRINT
2190 FOR I=1 TO 4:PRINT TAB(X);S$:NEXT I
2200 RETURN
3100 REM**DISPLAY FOR ACE**
3110 PRINT Y$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);P1$
3120 PRINT TAB(X);P4$:PRINT TAB(X);P1$
3130 PRINT Y$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);P1$
3140 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);P1$
3150 PRINTY$:PRINT:PRINT TAB(X);P3$:FOR I=1 TO 5:PRINT:NEXT I:PRINTTAB(X);P1$
3160 PRINT Y$:PRINT TAB(X);D$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);D$
3170 RETURN
3200 REM**DISPLAY FOR 2
3210 PRINT Y$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3220 PRINT TAB(X);D$:PRINT TAB(X);D$
3230 PRINT Y$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);C$
3240 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);C$
3250 PRINTY$:PRINT:PRINTTAB(X);D$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);D$
3260 PRINT Y$:PRINT TAB(X);A2$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);E2$
3270 RETURN
3300 REM**DISPLAY FOR 3
3310 PRINT Y$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3320 PRINT TAB(X);C$:PRINT TAB(X);D$
3330 PRINT Y$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);C$
3340 FOR I=1 TO 3:PRINT:NEXT I:PRINTTAB(X);C$
3350 PRINTY$:PRINT:PRINT TAB(X);D$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);D$
3360 PRINT Y$:PRINT TAB(X);A3$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);E3$
3370 RETURN
3400 REM**DISPLAY FOR 4**
3410 PRINT Y$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3420 PRINT TAB(X);D$:PRINT TAB(X);D$
3430 PRINT Y$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);D$
3440 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3450 PRINTY$:PRINT:PRINTTAB(X);B$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);B$
3460 PRINT Y$:PRINT TAB(X);A4$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);E4$
3470 RETURN
3500 REM**DISPLAY FOR 5**
3510 PRINT Y$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3520 PRINT TAB(X);C$:PRINT TAB(X);D$
3530 PRINT Y$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);D$
3540 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3550 PRINTY$:PRINT:PRINTTAB(X);B$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);B$
3560 PRINT Y$:PRINT TAB(X);A5$:FOR I=1 TO 7:PRINT:NEXT I:PRINTTAB(X);E5$
3570 RETURN
3600 REM**DISPLAY FOR 6**
3610 PRINT Y$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3620 PRINT TAB(X);B$:PRINT TAB(X);D$
3630 PRINT Y$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);C$
3640 FOR I=1 TO 3:PRINT:NEXT I:PRINTTAB(X);D$
3650 PRINTY$:PRINT:PRINT TAB(X);B$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);D$
3660 PRINT Y$:PRINT TAB(X);A6$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);E6$
3670 RETURN
```

```
3700 REM**DISPLAY FOR 7**
3710 PRINT V$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3720 PRINT TAB(X);C$:PRINT TAB(X);D$
3730 PRINT V$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);C$
3740 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);C$
3750 PRINTV$:PRINT:PRINT TAB(X);B$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);B$
3760 PRINT V$:PRINT TAB(X);A7$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);E7$
3770 RETURN
3800 REM**DISPLAY FOR 8**
3810 PRINT V$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);B$
3820 PRINT TAB(X);D$:PRINT TAB(X);B$
3830 PRINT V$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);D$
3840 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3850 PRINTV$:PRINT:PRINT TAB(X);B$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);B$
3860 PRINT V$:PRINT TAB(X);A8$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);E8$
3870 RETURN
3900 REM**DISPLAY FOR 9**
3910 PRINT V$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);B$
3920 PRINT TAB(X);C$:PRINT TAB(X);B$
3930 PRINT V$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);D$
3940 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);D$
3950 PRINTV$:PRINT:PRINT TAB(X);B$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);B$
3960 PRINT V$:PRINT TAB(X);A9$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);E9$
3970 RETURN
4000 REM**DISPLAY FOR 10**
4010 PRINT V$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);B$
4020 PRINT TAB(X);D$:PRINT TAB(X);B$
4030 PRINT V$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);C$
4040 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);C$
4050 PRINTV$:PRINT:PRINT TAB(X);B$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);B$
4060 PRINT V$:PRINT TAB(X);A0$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);E0$
4070 RETURN
4100 REM**DISPLAY FOR JACK**
4110 PRINT V$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);P2$
4120 PRINT TAB(X);P2$:PRINT TAB(X);P2$
4130 PRINT V$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);P2$
4140 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);P5$
4150 PRINTV$:PRINT:PRINTTAB(X);P4$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);P6$
4160 PRINT V$:PRINT TAB(X);D$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);D$
4170 RETURN
4200 REM**DISPLAY FOR QUEEN**
4210 PRINT V$: FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);P1$
4220 PRINT TAB(X);P1$:PRINT TAB(X);P7$
4230 PRINT V$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);P1$
4240 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);P7$
4250 PRINTV$:PRINT:PRINT TAB(X);P3$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);P3$
4260 PRINT V$:PRINT TAB(X);D$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);D$
4270 RETURN
4300 REM**DISPLAY FOR KING**
4310 PRINT V$:FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);P0$
4320 PRINT TAB(X);P$:PRINT TAB(X);P0$
4330 PRINT V$:FOR I=1 TO 2:PRINT:NEXT I:PRINT TAB(X);P5$
4340 FOR I=1 TO 3:PRINT:NEXT I:PRINT TAB(X);P5$
4350 PRINTV$:PRINT:PRINT TAB(X);P1$:FOR I=1 TO 5:PRINT:NEXT I:PRINT TAB(X);P1$
4360 PRINT V$:PRINT TAB(X);D$:FOR I=1 TO 7:PRINT:NEXT I:PRINT TAB(X);D$
4370 RETURN
```

```
2 REM 'ROULETTE' BY F.E.WOODWARD,8 VIOLET AV.RAMSGATE
4 REM WRITTEN MAY/1981 FOR MZ-80K
8 GOTO 110
10 REM MOVING BALL ROUTINE
11 POKE54021,64:POKE54020,71:POKE54019,71:IF E=11 THEN 480
12 POKE54020,64:POKE54019,64:POKE54018,71:IF E=12 THEN 480
13 POKE54018,64:POKE54017,71:IF E=13 THEN 480
14 POKE54017,64:POKE53976,71:IF E=14 THEN 480
15 POKE53976,64:POKE53935,71:IF E=15 THEN 480
16 POKE53935,64:POKE53894,71:IF E=16 THEN 480
17 POKE53894,64:POKE53853,71:IF E=17 THEN 480
18 POKE53853,64:POKE53813,71:IF E=18 THEN 480
19 POKE53813,64:POKE53773,71:IF E=19 THEN 480
20 POKE53773,64:POKE53733,71:IF E=20 THEN 480
21 POKE53733,64:POKE53693,71:IF E=21 THEN 480
22 POKE53693,64:POKE53653,71:IF E=22 THEN 480
23 POKE53653,64:POKE53613,71:IF E=23 THEN 480
24 POKE53613,64:POKE53574,71:IF E=24 THEN 480
25 POKE53574,64:POKE53535,71:IF E=25 THEN 480
26 POKE53535,64:POKE53496,71:IF E=26 THEN 480
27 POKE53496,64:POKE53457,71:IF E=27 THEN 480
28 POKE53457,64:POKE53458,71:IF E=28 THEN 480
29 POKE53458,64:POKE53459,71:IF E=29 THEN 480
30 POKE53459,64:POKE53460,71:IF E=30 THEN 480
31 POKE53460,64:POKE53461,71:IF E=31 THEN 480
32 POKE53461,64:POKE53462,71:IF E=32 THEN 480
33 POKE53462,64:POKE53503,71:IF E=33 THEN 480
34 POKE53503,64:POKE53544,71:IF E=34 THEN 480
35 POKE53544,64:POKE53585,71:IF E=35 THEN 480
36 POKE53585,64:POKE53626,71:IF E=36 THEN 480
37 POKE53626,64:POKE53666,71:IF E=37 THEN 480
38 POKE53666,64:POKE53706,71:IF E=38 THEN 480
39 POKE53706,64:POKE53746,71:IF E=39 THEN 480
40 POKE53746,64:POKE53786,71:IF E=40 THEN 480
41 POKE53786,64:POKE53826,71:IF E=41 THEN 480
42 POKE53826,64:POKE53866,71:IF E=42 THEN 480
43 POKE53866,64:POKE53905,71:IF E=43 THEN 480
44 POKE53905,64:POKE53944,71:IF E=44 THEN 480
45 POKE53944,64:POKE53983,71:IF E=45 THEN 480
46 POKE53983,64:POKE54022,71:IF E=46 THEN 480
47 POKE54022,64:POKE54021,71:IF E=47 THEN 480
50 GOTO 460
100 REM WHEEL DISP.
110 PRINT "□":PRINT
120 PRINT TAB(8);"▒▒▒▒▒▒▒▒▒▒"
130 PRINT TAB(7);"▒# 21 21#▒"
140 PRINT TAB(6);"▒ #830546# ▒"
150 PRINT TAB(5);"▒  #♠♥♣♥♠♥#  ▒"
160 PRINT TAB(4);"▒####       ####▒"
170 PRINT TAB(3);"▒ 30♥          ♠33 ▒"
180 PRINT TAB(2);"▒ 11♠     ▲     ♥1  ▒"
```

```
190 PRINT TAB(1);"▒ 36♥          ♠20 ▒"
200 PRINT TAB(1);"▒13♠           ♥14▒"
210 PRINT TAB(1);"▒27♥           ♠13▒"
220 PRINT TAB(1);"▒ 6♠           ♥9 ▒"
230 PRINT TAB(1);"▒34♥           ♠23▒"
240 PRINT TAB(1);"▒17♠           ♥18▒"
250 PRINT TAB(1);"▒25♥           ♠29▒"
260 PRINT TAB(1);"▒ 2♠           ♥7 ▒"
270 PRINT TAB(1);"▒ 21♥         ♠28 ▒"
280 PRINT TAB(2);"▒  4♠        ♥12 ▒"
290 PRINT TAB(3);"▒ 19♥       ♠35 ▒"
300 PRINT TAB(4);"▒####      ####▒"
310 PRINT TAB(5);"▒  #♠♥";CHR$(219);CHR$(220);"♠♥#  ▒"
320 PRINT TAB(6);"▒ #13";CHR$(234);CHR$(235);"23# ▒"
330 PRINT TAB(7);"▒#52  6 #▒"
340 PRINT TAB(8);"▒▒▒▒▒▒▒▒▒▒"
360 X$="■■■■■■■■■■■■■■■■":Y$="■■■■■■■■":PRINTY$;TAB(28);"ROULETTE"
365 PRINT TAB(28);"********"
370 PRINT X$;TAB(25);"KEY'S'TO START"
400 GET S$:IF S$="S" THEN 420:REM STARTSMOVING BALL DISP.
410 GOTO 400
420 R=INT(37*RND(1)+1):X=1:PRINT X$;TAB(25);SPC(15)
430 IF R>18 THEN 435
435 T=R-18
437 GOTO 450
440 ON R GOTO 11,12,13,14,15,16,17,18,19,20,21,23,24,25,26,27,28
450 ONT GOTO 29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47
460 X=X+1:IF X<9 GOTO 11
470 S=INT(37*RND(1)+1):E=10+S
475 GOTO 11
480 PRINT X$;TAB(24);"DID YOU WIN"
490 PRINT TAB(25);"DO YOU WISH TO"
500 PRINT TAB(25);"PLAY AGAIN? Y/N"
510 GET A$
520 IF A$="Y" GOTO 550
530 IF A$="N" GOTO 560
540 GOTO 510
550 PRINT "■":CLR:GOTO 110
560 PRINT "■" : END
```

SHARPSOFT Sharpsoft Ltd., 86-90 Paul Street, London EC2A 4NE
Printed by Oldham Press (T.U.), Chatham, Kent.