SHARPSOFT

Issue No. 10

C O N T E N T S

SHARPSOFT USER NOTES

ISSUE NO. 10

EDITORIAL

This year we will be publishing the User Notes six times, rather than three times. A number of readers have written to me expressing the opinion that issues with less pages, but on a more regular basis, would be a better way to communicate our and your ideas to readers.

1984 promises to be an exciting year for SHARP computer users. We now have a good range of tape-based software for the MZ-80K/A/B with new items constantly being added. On the language front the interest in Hisoft PASCAL and FORTH continues to grow. To help beginners get started with PASCAL we are including a single tutorial introduction to this language - Part 1 is in this issue. The series will continue throughout this year's issues. For the FORTH fans, I have added a page or two on graphics words. The BASIC article called "Quick on the Draw" (© Mr. John Simpson) should interest readers writing graphics programs.

Mr. Peter Sydenham has contributed an article on his experiences when interfacing a single disk drive to the MZ-80K. This is a long article and hence will be published in four consecutive issues of the User Notes.

Once again we feature your letters and listings. These you will find at the back of the Issue.

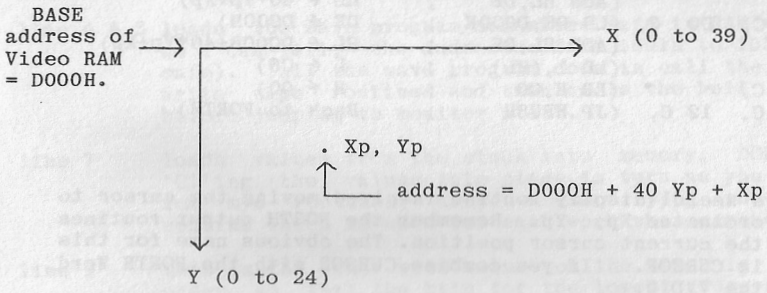
At this time of year I start to plan the themes for later Issues. If you have any material for inclusion in the User Notes, any programs, tips on PASCAL, FORTH, BASIC, machine code, or indeed more general points-of-view, please send them to me as soon as possible.

Issue 11 of the User Notes will be published in April 1984.

MIKE BRINSON
Editor

HIGH SPEED GRAPHICS WORDS - by Mike Brinson

Often in games programs one needs to write or read a display character from the Video RAM. The higher the speed of these operations the greater the degree of animation we can achieve. The main factor which slows down animation is the calculation of the address of the display position X,Y co-ordinated, for example:



Either a graphics "POKE" or "PEEK" operation written in high-level FORTH would involve one multiplication and two additions using the FORTH words * and +. The overhead associated with the FORTH threaded code would significantly slow down the POKE/PEEK operations. An impressive reduction in run time can be achieved if we code graphics routines in machine code. the two FORTH primitives shown below, called GPOKEXY and GPEEKXY, illustrate a shift technique for multiplication by a constant.

```
HEX
CREATE GPOKEXY   (C6. Xp Yp -> )

E1 C,                  (POP HL          ;     HL ← Yp)
29 C,                  (ADD HL,HL       ;     HL ← 2*Yp)
29 C,                  (ADD HL,HL       ;     HL ← 4*Yp)
29 C,                  (ADD HL,HL       ;     HL ← 8*Yp)
5D C,                  (LD   E,L                 )
54 C,                  (LD   D,H        ;     DE ← 8*Yp)
29 C,                  (ADD HL,HL       ;     HL ← 16*Yp)
29 C,                  (ADD HL,HL       ;     HL ← 32*Yp)
19 C,                  (ADD HL,DE       ;     HL ← 40*Yp)
D1 C,                  (POP DE          ;     DE ← Xp)
19 C,                  (Add HL,DE       ;     HL ← 40*Yp+Xp)
11 C,  00  C,  D0  C,  (LD DE,D000H     ;     DE ← D000H)
19 C,                  (ADD HL,DE       ;     HL ← D000H+40*Yp+Xp)
D1 C,                  (POP DE          ;      E ← C6)
73 C,                  (LD (HL),E       ;     (D000H+40*Yp+Xp) ← C6
C3 C,
45 C,
12 C,                  (JP NEXT         ;     Back to FORTH)
```

```
SMUDGE
CREATE GPEEKXY   (Xp Yp -→  C6.)

E1 C,                   (POP HL         ;     HL ← Yp)
29 C,                   (ADD HL,HL      ;     HL ← 2*Yp)
29 C,                   (ADD HL,HL      ;     HL ← 4*Yp)
29 C,                   (ADD HL,HL      ;     HL ← 8*Yp)
5D C,                   (LD   E,L                )
54 C,                   (LD   D,H       ;     DE ← 8*Yp)
29 C,                   (ADD HL,HL      ;     HL ← 16*Yp)
29 C,                   (ADD HL,HL      ;     HL ← 32*Yp)
19 C,                   (ADD HL,DE      ;     HL ← 40*Yp)
D1 C,                   (POP DE         ;     DE ← Xp)
19 C,                   (Add HL,DE      ;     HL ← 40*Yp+Xp)
11 C,  00  C,  DO  C,   (LD DE,D000H    ;     DE ← D000H)
19 C,                   (ADD HL,DE      ;     HL ← D000H+40*Yp+Xp)
6E C,                   (LD L,(HL)      ;      L ← C6)
26 C,  00  C,           (LD H,00        ;      H ← 00)
C3 C,  44  C,  12 C,    (JP HPUSH       ;     Back to FORTH)

SMUDGE
```

A third useful display routine involved moving the cursor to display co-ordinates Xp, Yp. Remember the FORTH output routines print from the current cursor position. The obvious name for this FORTH word is CURSOR. If you combine CURSOR with the FORTH Word speed!) to the V.D.U.

```
HEX
CREATE  CURSOR  ( Xp Yp -→  )

D1 C,                   (POP DE         ;     DE ← Yp)
21 C,
72 C,
11 C,                   (LD HL,1172H                )
73 C,                   (LD (HL),E      ; sets cursor ↑↓ 0-24)
D1 C,                   (POP DE         ;     DE - Xp)
21 C,
71 C,
11 C,                   (LD HL,1172H                )
73 C,                   (LD (HL),E      ; sets cursor ←- -→  0-39)
C3 C,
45 C,
12 C,                   (JP NEXT        ; Back to FORTH)
```

****

THE FRAGILE FORTH FIX

by

Mr. L.G. Miller of Cheshire

As people know by now, SHARPSOFT FORTH is quite fragile and as I had to use it quite a lot recently, I got rather upset when it crashed just because a shifted letter was typed in. So cursing the implementors (quite wrongly as it turned out) I dug out a FIG-FORTH assembly listing that's been here for two years and typed it in. After it was up and running, guess what I found? -- it crashed when I entered graphics characters and things! -- not very funny.

This brings me to the fix, the problem is in the word (FIND), this is the search dictionary routine that tries to match the input word. It gets very confused when the high bit is set in the input characters; as it thinks it has reached the end of a name in the dictionary, even though it has not.

The difficult part for me was getting the fix into the same space (I did say difficult for me, not necessarily others). The screen listing should be clear enough, but here is an expansion for those who are not sure what all the bits on it do:-

line 2 sets the tape work areas of the monitor to: file type, size of FORTH, load address and execution address.

line 4 & 5 loads the save program somewhere safe in RAM (as far as I can tell, the addresses from 1001H to 1037H are safe). All the save program does is call the monitor write tape routines and then calls the bell routine before jumping to monitor start.

line 7 loads values from the stack into memory, DON'T try 'C!'ing the values into place in turn as you may do for m/c routines, as you are modifying the search routine which is used for each work when executing!

line 9 this is the fix, with values on the stack in reverse order so that the byte for the lowest address is on top.

line 10 load fix and then cold start FORTH.

When you have loaded the screen, try typing shifted characters and entering them. If all is OK then type BYE, put your cassette in recorder, press record and play, then type in GOTO$1001.

All this may be obvious to all of you, in which case - sorry.

```
Block 8001
  0 ( KFIX - STOP K CRASHING IF GRAPHICS CHARACTERS INPUT            )
  1 HEX              ( SET TAPE PARAMETERS    )
  2  1 10F0 !  HERE 1200 - 1102 !  1200 1104 !  1200 1106 !
  3 ( LOAD SAVE ROUTINE AT 1001H     )
  4  0CD 1001 C!  0021 1002 !  0CD 1004 C! 0024 1005 !
  5  0CD 1007 C!  003E 1008 !  0C3 100A C!     0 100B !
  6
  7  : LOADER  1356 0E + 1356 DO I C! LOOP ;   ( LOAD PATCH )
  8
  9  0F4 30 87 1A 1C 20 0AE 7F 0E6 1A 13 23 23 20 ( PATCH )
 10  LOADER
 11  COLD ;S
 12 ( INSTRUCTIONS - LOAD THIS SCREEN, AFTER COLD START MESSAGE,  )
 13 ( TYPE BYE AND PUT A TAPE IN THE CASSETTE, TYPE GOTO$1001     )
 14 ( AFTER BEEP, STOP THE TAPE AS THE COPY WILL BE COMPLETE.     )
 15
```

BEGINNERS GUIDE TO PASCAL

PART I - Fundamentals

These notes are intended for users of Hisoft tape PASCAL. the material is presented as a reference introduction to PASCAL.

1. PASCAL is a structured programming language.

2. Programs are constructed using procedures and functions. Each procedure or function usally does one simple task.

3. When programming in PASCAL it is:-

   (a) easier to debug programs, and

   (b) easier to design programs.

4. PASCAL is a must for programmers who write large programs in BASIC and who have experienced debug problems.

5. A simple PASCAL program is:

```
PROGRAM Ex1;
BEGIN
  WRITELN('SHARPSOFT')
END.
```

6. To execute (run) program EX1

   (a) Enter the program text - using an EDITOR.

   (b) Compile the program - Hisoft command C.

   (c) Run the program - Hisoft command R.

7. Editor and compiler commands - see Hisoft Manual.

8. PASCAL Program structure.

```
PROGRAM name;

+-------------+
| Declaration |
| Section     |
+-------------+

BEGIN

+-------------+
| Body        |
+-------------+

END.
```

9. Declaration section

```
LABEL          )
CONST          )   must be in
TYPE           )   this order
VAR            )

PROCEDURE      )   any number of declarations
FUNCTION       )         in any order
```

10. What is a PASCAL program?
What is a string of tokens?

11. PASCAL language tokens

(a)	Basic symbols	- letters or digits
(b)	Special symbols	- + - / * etc. and reserved words, for example FOR IN CASE
(c)	Identifiers	- For example BREAD BUTTER P1 K342, must start with a letter - only the first 8 characters are treated as significant. Identifiers may contain lower or upper case letters - hence HELLO and hello are different. Reserved words may not be used as identifiers.
(d)	Numbers	- Integer for example -100 0 3276
		- Real for example 1.2 -1.7E-5
(e)	Strings	- 'SHARPSOFT' - note single quote
(f)	Comments	- (* This is a comment *)
(g)	Separators	- ; space and carriage return

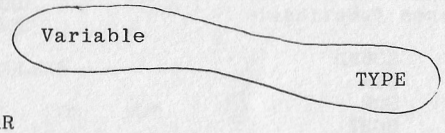
12. PASCAL constants.

```
CONST.

     pi    =  3.14159;
     ONE   =  1;
     CH    =  'A';
     STOP  =  FALSE
```

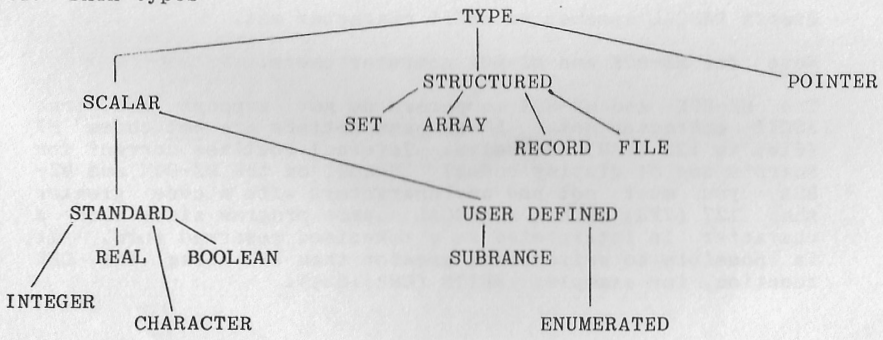
13. Variable declarations

All variables have a TYPE



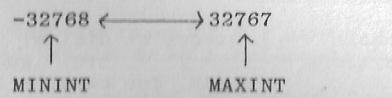

```
VAR
     COUNT, INDEX       :  INTEGER;
     LENGTH,WIDTH,AREA  :  REAL;
```

14. DATA types



15. Predefined types

(a) INTEGER



stored as two bytes

Operators

1. Assignment :=
2. Arithmetic + = * DIV MOD
3. Relational < <= = <> >= > gives boolean result
4. Predefined functions.

ABS	COS	ODD	SQRT
ARCTAN	EXP	PRED	SUCC
CHR	LN	SQR	

(b) REAL

[+ or -] <Whole part>.[<Fractional part>][E][[+ or -]exponent]

largest real is 3.4 E 38 stored as 4 bytes

smallest real is 5.9 E-39

The mantissa is 23 bites in length, yielding an accuracy of about 7 significant figures.

Operators

1. Assignment :=
2. Arithmetic + = * DIV MOD
3. Relational < <= = <> >= > gives boolean result
4. Predefined functions.

ABS	ROUND
ARCTAN	SIN
COS	SQR
EXP	SQRT
LN	TRUNC

(c) CHAR (+ characters *)

Hisoft PASCAL assumes an ASCII character set.

Note for MZ-80K and MZ-80A computer users.

The MZ-80K and MZ-80A computers do not support the true ASCII character set. Lower case letters are not codes 97 (61H) to 122 (7AH) inclusive. Internal routines correct for Sharp's use of display codes! Hence, on the MZ-80K and MZ-80A, you must not use any characters with a code greater than 127 (7FH) within a PASCAL source program since such a character is interpreted as a tokenised reserved word. It is possible to write codes greater than 7FH using the CHR function, for example: WRITE (CHR(132)).

Note for the MZ-80B Users - The above restriction does not apply because the MZ-80B uses a true ASCII character set.

Characters are stored as 1 byte, in pure, usigned binary.
Characters are denoted by single quotes.

Operators

1. Assignment :=
2. Relational < <= = <> >= > gives boolean result
3. Predefined functions

   ORD
   PRED
   SUCC

(d) BOOLEAN

FALSE OR TRUE stored as 1 byte, in pure, unsigned binary.

ORD(TRUE) = 1, so TRUE is represented by 1.
ORD (FALSE) = 0, so FALSE is represented by 0.

NOTE FALSE < TRUE

Operators

1. Assignment :=
2. Relational < <= = <> >= > gives boolean result
3. Predefined functions

   PRED
   SUCC

4. BOOLEAN

   AND OR NOT

CONTINUED NEXT ISSUE

MZ-80 CP/M NOTES

A number of CP/M users have written to us asking if there are any public domain Z80 assemblers. The answer is yes Volume 16 of the American CP/M Library contains a program called Z80ASM.COM and other similar software. The catalogue for this disk is given below:

VOLUME 16 ASSEMBLERS, OTHER UTILITIES AND FOCAL

NUMBER	SIZE	NAME	COMMENTS
		CATALOG.16	CONTENTS OF CP/M GROUP VOL 16
		VOLUME16.DOC	COMMENTS ON CERTAIN PROGRAMS
16.1	9K	ASMX.COM	ASSEMBLER WHICH RECOGNIZES Z-80 OPS SEE VOL.DOC [CAREFUL: WITH CORRECT SYNTAX (ASMX FILENAME.AA) THIS DOES WORK. WITH FAULTY SYNTAX THE PROGRAM TAKES REVENGE ON THE DISK DIRECTORY.] RUNS OK ON 8080
16.2	12K	COPYDSK.ASM	DISK COPY PROGRAM. SEE VOL.DOC
16.3	13K	COPYDSK.MAC	AS 16.2 FOR TDL ASSEMBLER
16.4	7K	CPMUTIL.ASM	CP/M SUBROUTINES USEFUL GENERALLY & EMPLOYED AS PART OF Z80ASM 16.17
16.5	3K	EDIT.COM	INTEL-LIKE EDITOR DOES L F B AND -B MUCH FASTER THAN ED.COM SEE VOL.DOC
16.6	8K	EDUCATOR.ASM	8080 INSTRUCTION SET TUTOR FROM BYTE JUL'76
16.7	57K	FOCAL.ASM	FOCAL LANGUAGE INTEPRETER. SEE VOL.DOC
16.8	8K	MACASM.COM	MACRO ASSEMBLER. SEE VOL.DOC
16.9	2K	MOVDOWN.ASM	PROGRAM TO LOAD FILE WHICH OPERATES BELOW 100H
16.10	2K	SEEK.ASM	SET DISK TRACK FROM FRONT PANEL DURING ALIGNMENT
16.11	9K	SPAT1.ASM	RE-WRITE OF 1.29 TO GENERALIZE CONSOLE FROM ORIGINAL VDM DEPENDANCY
16.12	2K	TASMIO.DOC	DOC FOR TASMIO PATCH TO PUT TDL TAPE ASSEMBLER UP ON CP/M
16.13	3K	TASMIO.HEX	SEE TASMIO.DOC
16.14	18K	TASMIO.MAC	SEE TASMIO.DOC
16.15	3K	TEST1A.ASM	SUCCESSFUL TEST FOR Z80ASM 16.17
16.16	1K	TEST2.ASM	UNSUCCESSFUL TEST FOR Z80ASM 16.17
16.17	9K	Z80ASM.COM	ZILOG MNEMONIC ASSEMBLER, RUNS ON 8080, SEE Z80DOC.DOC 16.18
16.18	4K	Z80ASM.DOC	DOC FOR 16.17
16.19	28K	Z80MAIN.ASM	SEE 16.17
19.20	4K	Z80OPCSA.ASM	SEE 16.17
16.21	11K	Z80SUBS.ASM	SEE 16.17

Unfortunately Z80ASM.COM is known to contain a number of bugs. These bugs have been documented by the CP/M UK User group in their journal - 8 ed.

CPMUGUK VOL 1 No 5, Feb 1982 (no page no.)
VOL 1 No 7, Sep 1982 p60-67

An improved Z80 assembler, called Z80ASM, has been included on Vol. 5 of the UK user group library. This versions is also reviewed by David Back on page 73 of CPMUGUK Vo. 1 No. 8, Nov 1982. MZ-80 Users interested in this assembler should contact the UK group.

QUICK ON THE DRAW!

© Mr. John Simpson

While developing a program on my Sharp MZ-80K I realised that it would need a great deal of graphics and that, if the program was to be effective, the drawing would have to be done as quickly and efficiently as possible. I then started to think about the different ways of putting graphics on to the screen and came to the conclusion that although I could think of many different ways of approaching the problem, I was by no means sure which method was fastest, and whether the fastest method was also economical in the amount of memory used. I decided, therefore, to run a series of test programs to compare the different methods.

The basic test was to draw a square of 10 x 10 characters in the centre of the screen and a FOR:NEXT loop was incorporated to run the routine 1000 times so as to show a measurable difference in the speeds.

The Sharp MZ-80K has a built-in clock facility, and this was used to time the program. Line 25 sets the clock to zero and lines 315-325 read the value after the test. Line 35 prints out the number of bytes used in the whole program - the elements which are common to all the test programs take up 181 bytes.

While the results listed hold good for the Sharp MZ-80K other computers may show a different performance. It should not be difficult to adapt the various programs so that you can tabulate results for your own computer. (If your computer does not have a clock facility you can time the programs with a stopwatch.) You can increase the value of I if you find that the differences are too small to measure easily (although having increased it you must then keep the same value for subsequent comparative tests), and print the results to the screen if you don't have a printer - to do this simply change the "PRINT/P" statements in the common elements to "PRINT".

The first program (No. 1) POKEs each character individually to the appropriate location on the screen. The next (No. 1A) adopts the same technique, but uses variables for the locations and display codes. This

requires slightly more memory, but gives a faster result.

Variant No. 1B again uses POKEs but this time makes use of FOR:NEXT loops to reduce the amount of memory required to 324 bytes, with only a slight penalty in terms of speed.

The final POKE method (No. 1C) sets up an array A(40) into which the POKE locations are read at the beginning of the program. This is less economical of memory but is reasonably fast.

Turning to the humble PRINT statement, this competes surprisingly well, and gives a cleaner picture without the 'snow' which seems to bedevil extensive use of POKE statements. No. 2 simply PRINTs each line, spaces included, and gives a reasonable time. No. 2A is the same as No. 2, but substitutes cursor control symbols for the spaces. As expected, the amount of memory required is the same, but surprisingly the program runs much more slowly. In view of this effect I did not test a further variation whereby a single PRINT line inserted each vertical line by taking the cursor down and back each time.

Program No. 2B saves memory by setting the PRINT position with a TAB statement. This also runs quite quickly. If, however, we set the position of the second vertical line by another TAB setting (as in No. 2C) this nearly doubles the time taken. Using SPC instead of TAB almost doubles the time again (see No. 2D).

From the experience of the previous test programs one would imagine that use of FOR:NEXT loops, and single TABs would be effective (No. 2E). While this saves memory it is surprisingly slow in execution.

I, therefore, returned to a variation on No. 2B, but this time using a FOR:NEXT loop to put in the vertical lines. This version No. 2F turned out to be economical in memory, and ran at a reasonable speed.

Program No. 2G puts the various components of the square into strings and then uses a FOR:NEXT loop and TAB to print the strings. This is quite successful, and version No. 2H uses the same approach but puts the spacing into the strings instead of using TAB settings. As might

be expected this uses a little more memory but saves on execution time.

The Sharp MZ-80K has the commands SET/RESET which give a double density display (50 x 80), allowing you to plot using a square which is one quarter of a normal character in size. Program No. 3 compares the use of this command. While it uses a relatively small amount of memory, it is not very fast and is only suitable for certain applications, as the squares do not touch each other and give a cruder effect than the excellent graphics characters available on the Sharp.

It is invidious to generalise from these results, and it must be reiterated that the following conclusions apply only to the Sharp MZ-80K - users of other machines should run the test programs (modified as appropriate) and draw up a reference table for their own computer.

A certain amount of personal preference also enters the matter, as some users may prefer to sacrifice some memory or speed for the sake of a routine which they find more convenient to use. In some circumstances one method may lend itself more naturally as the solution to a particular problem. However, on balance, the use of strings containing the components for incorporation into PRINT statements would seem to emerge as a good all round method, particularly if the display is split up in such a way that the same strings can be used in different parts of the display. (For example, the repetition of B$ in Test Program 2G.)

For this simple task of drawing a square I have illustrated 14 different methods, and many more slight variations could have been included. I have not, however, included the use of the "PRINT @" command, as this does not feature in all versions of BASIC for the Sharp.

It is hoped that these tests and notes will help novice programmers to realise that for a given problem there are very often several solutions, and the first one which comes to mind may not necessarily be the best. The test techniques illustrated should also be helpful when carrying out comparisons of other programming methods.

COMPARATIVE TABLE

Program No.	Description	Bytes	Time Minutes : Seconds
1	Individual POKEs to locations	656	3 : 41
1A	As No. 1, but using variables for locations and display codes	671	3 : 11
1B	FOR:NEXT loops to POKE display codes	324	3 : 36
1C	Array of POKE locations and variables for display codes	600	3 : 28
2	ALL PRINT statements with spaces	607	3 : 20
2A	ALL PRINT statements with cursor symbols	607	6 : 06
2B	ALL PRINT statements with 1 TAB setting per line	497	3 : 36
2C	ALL PRINT statements with 2 TAB settings per line	477	6 : 23
2D	ALL PRINT statements with 2 SPC settings per line	477	12 : 12
2E	PRINT statements using FOR:NEXT loops and TAB	309	9 : 09
2F	PRINT statements with 1 TAB setting per line using FOR:NEXT loop	287	3 : 54
2G	Components in strings - FOR:NEXT loop with 1 TAB setting to PRINT	305	3 : 53
2H	Components in strings with spaces. PRINT using FOR:NEXT loop	342	3 : 20
3	SET command	292	5 : 53

```
5 REM***COMMON ELEMENT 1***
15 PRINT"■"
25 TI$="000000"
35 PRINT/P:PRINT/P:PRINT/P34680-SIZE;" BYTES"
100 REM***TEST PROGRAMS***
200 REM***ARE  INSERTED***
300 REM********HERE********
305 REM***COMMON ELEMENT2***
315 TI=VAL(TI$)
325 PRINT/P"TIME ELAPSED:";TI
335 END
```

```
95 REM***TEST PROGRAM #1***
100 FORI=1TO1000
110 POKE53423,60:POKE53424,60:POKE53425,60:POKE53426,60
120 POKE53427,60:POKE53428,60:POKE53429,60:POKE53430,60
130 POKE53431,60:POKE53432,60
140 POKE53462,61:POKE53502,61:POKE53542,61:POKE53582,61
150 POKE53622,61:POKE53662,61:POKE53702,61:POKE53742,61
160 POKE53782,61:POKE53822,61
170 POKE53473,113:POKE53513,113:POKE53553,113:POKE53593,113
180 POKE53633,113:POKE53673,113:POKE53713,113:POKE53753,113
190 POKE53793,113:POKE53833,113
200 POKE53863,112:POKE53864,112:POKE53865,112:POKE53866,112
210 POKE53867,112:POKE53868,112:POKE53869,112:POKE53870,112
220 POKE53871,112:POKE53872,112
300 NEXTI
```

```
95 REM***TEST PROGRAM #1A***
100 SC=53248:A=60:B=61:C=113:D=112
110 FORI=1TO1000
120 POKESC+175,A:POKESC+176,A:POKESC+177,A:POKESC+178,A
130 POKESC+179,A:POKESC+180,A:POKESC+181,A:POKESC+182,A
140 POKESC+183,A:POKESC+184,A
150 POKESC+214,B:POKESC+254,B:POKESC+294,B:POKESC+334,B
160 POKESC+374,B:POKESC+414,B:POKESC+454,B:POKESC+494,B
170 POKESC+534,B:POKESC+574,B
180 POKESC+225,C:POKESC+265,C:POKESC+305,C:POKESC+345,C
190 POKESC+385,C:POKESC+425,C:POKESC+465,C:POKESC+505,C
200 POKESC+545,C:POKESC+585,C
210 POKESC+615,D:POKESC+616,D:POKESC+617,D:POKESC+618,D
220 POKESC+619,D:POKESC+620,D:POKESC+621,D:POKESC+622,D
230 POKESC+623,D:POKESC+624,D
300 NEXTI
```

```
95 REM***TEST PROGRAM #1B***
100 FORI=1TO1000
110 FORJ=53423TO53432
120 POKEJ,60:NEXTJ
130 FORJ=53462TO53822STEP40
140 POKEJ,61:NEXTJ
150 FORJ=53473TO53833STEP40
160 POKEJ,113:NEXTJ
170 FORJ=53863TO53872
180 POKEJ,112:NEXTJ
300 NEXTI
```

```
95 REM***TEST PROGRAM #1C***
100 DIMA(40)
110 FORJ=1TO40:READA(J):NEXT
120 A=60:B=61:C=113:D=112
130 FORI=1TO1000
140 FORJ=1TO10:POKEA(J),A:NEXTJ
150 FORJ=11TO20:POKEA(J),B:NEXTJ
160 FORJ=21TO30:POKEA(J),C:NEXTJ
170 FORJ=31TO40:POKEA(J),D:NEXTJ
200 NEXTI
250 DATA53423,53424,53425,53426,53427,53428,53429,53430,53431,53432
260 DATA53462,53502,53542,53582,53622,53662,53702,53742,53782,53822
270 DATA53473,53513,53553,53593,53633,53673,53713,53753,53793,53833
300 DATA53863,53864,53865,53866,53867,53868,53869,53870,53871,53872
```

```
95 REM***TEST PROGRAM #2***
100 FORI=1TO1000
110 PRINT"■■■■■■"
120 PRINT"                 _____"
130 PRINT"                |     | "
140 PRINT"                |     | "
150 PRINT"                |     | "
160 PRINT"                |     | "
170 PRINT"                |     | "
180 PRINT"                |     | "
190 PRINT"                |     | "
200 PRINT"                |     | "
210 PRINT"                |     | "
220 PRINT"                |_____| "
230 PRINT"
300 NEXTI
```

```
95 REM***TEST PROGRAM #2A***
100 FORI=1TO1000
110 PRINT"▒▒▒▒▒"
120 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒____________"
130 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
140 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
150 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
160 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
170 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
180 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
19C PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
200 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
210 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
220 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒ ▒▒▒▒▒▒▒▒▒▒| "
230 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒‾‾‾‾‾‾‾‾‾‾‾‾"
300 NEXTI
```

```
95 REM***TEST PROGRAM #2B***
100 FORI=1TO1000
110 PRINT"▒▒▒▒▒"
120 PRINTTAB(16);"__________"
130 PRINTTAB(15);" |          | "
140 PRINTTAB(15);" |          | "
150 PRINTTAB(15);" |          | "
160 PRINTTAB(15);" |          | "
170 PRINTTAB(15);" |          | "
180 PRINTTAB(15);" |          | "
190 PRINTTAB(15);" |          | "
200 PRINTTAB(15);" |          | "
210 PRINTTAB(15);" |          | "
220 PRINTTAB(15);" |          | "
230 PRINTTAB(16);"‾‾‾‾‾‾‾‾‾‾"
300 NEXTI
```

```
95 REM***TEST PROGRAM #2C***
100 FORI=1TO1000
110 PRINT"▒▒▒▒▒"
120 PRINTTAB(16);"__________"
130 PRINTTAB(15);" |";TAB(26);"| "
140 PRINTTAB(15);" |";TAB(26);"| "
150 PRINTTAB(15);" |";TAB(26);"| "
160 PRINTTAB(15);" |";TAB(26);"| "
170 PRINTTAB(15);" |";TAB(26);"| "
180 PRINTTAB(15);" |";TAB(26);"| "
190 PRINTTAB(15);" |";TAB(26);"| "
200 PRINTTAB(15);" |";TAB(26);"| "
210 PRINTTAB(15);" |";TAB(26);"| "
220 PRINTTAB(15);" |";TAB(26);"| "
230 PRINTTAB(16);"‾‾‾‾‾‾‾‾‾‾"
300 NEXTI
```

```
95 REM***TEST PROGRAM #2D***
100 FORI=1TO1000
110 PRINT"▒▒▒▒▒"
120 PRINTSPC(16);"__________"
130 PRINTSPC(15);" |";SPC(10);"| "
140 PRINTSPC(15);" |";SPC(10);"| "
150 PRINTSPC(15);" |";SPC(10);"| "
160 PRINTSPC(15);" |";SPC(10);"| "
170 PRINTSPC(15);" |";SPC(10);"| "
180 PRINTSPC(15);" |";SPC(10);"| "
190 PRINTSPC(15);" |";SPC(10);"| "
200 PRINTSPC(15);" |";SPC(10);"| "
210 PRINTSPC(15);" |";SPC(10);"| "
220 PRINTSPC(15);" |";SPC(10);"| "
230 PRINTSPC(16);"‾‾‾‾‾‾‾‾‾‾"
300 NEXTI
```

```
95 REM***TEST PROGRAM #2E***
100 FORI=1TO1000
110 PRINT"■■■■■■■■"
120 PRINTTAB(16);
130 FORJ=1TO9:PRINT"_";:NEXTJ:PRINT"_"
140 FORJ=1TO10:PRINTTAB(15);" |          | ";:NEXTJ
150 PRINTTAB(16);
160 FORJ=1TO9:PRINT"¯";:NEXTJ:PRINT"¯"
300 NEXTI
```

```
95 REM***TEST PROGRAM #2F***
100 FORI=1TO1000
110 PRINT"■■■■■■■■"
120 PRINTTAB(16);"____________"
130 FORJ=1TO10:PRINTTAB(15);" |          | "
140 NEXTJ
150 PRINTTAB(16);"¯¯¯¯¯¯¯¯¯¯¯¯"
300 NEXTI
```

```
95 REM***TEST PROGRAM #2G***
100 A$="____________":B$=" |          | ":C$="¯¯¯¯¯¯¯¯¯¯¯¯"
110 FORI=1TO1000
120 PRINT"■■■■■■■■"
130 PRINTTAB(16);A$
140 FORJ=1TO10:PRINTTAB(15);B$:NEXTJ
150 PRINTTAB(16);C$
300 NEXTI
```

```
95 REM***TEST PROGRAM #2H***
100 A$="            ____________"
110 B$="           |            | "
120 C$="            ¯¯¯¯¯¯¯¯¯¯¯¯"
130 FORI=1TO1000
140 PRINT"■■■■■■■■"
150 PRINTA$
160 FORJ=1TO10:PRINTB$:NEXTJ
170 PRINTC$
300 NEXTI
```

```
95 REM***TEST PROGRAM #3***
100 FORI=1TO1000
110 Y=10:FORX=30TO50:SETX,Y:NEXTX
120 X=30:FORY=11TO30:SETX,Y:NEXTY
130 X=50:FORY=11TO30:SETX,Y:NEXTY
140 Y=31:FORX=30TO50:SETX,Y:NEXTX
300 NEXTI
```

SHARPSOFT

Sharpsoft Ltd.,
86-90 Paul Street,
London EC2A 4NE
Tel: 01-739 8559

S P E C I A L O F F E R S

FOR MEMBERS OF

SHARPSOFT USER NOTES

ISSUE No. 10

Dear Member, March, 1984

As you know, we intend to produce 6 Issues of SUN for 1984. This increased number of Issues makes it even more important that you continue to support the User Notes by sending in your articles, notes and program listings, together with your letters, containing both questions and, in many cases, answers. As you can see from the Contents Page of this Issue, many of the most interesting ideas and developments come from you. For articles or program listings published we will give a five pound Sharpsoft Voucher that may be exchanged for either hardware or software.

If you are thinking about purchasing a printer, please ring or write asking for a quote, as we have been successful in doing a few 'good deals' with some new suppliers and as a SUN Subscriber we offer you that little extra discount.

As we are running the Beginners Guide to Pascal, based on the Hisoft Pascal Compiler, those of you who wish to follow this over the next few issues and don't already have Hisoft PASCAL, we are offering it for 36.50 instead of the recommended 40.25 (overseas Members ADD 2.00 for postage). Any PASCAL listings, or tips from current Hisoft users, would be most helpful in supporting this feature, so please send them in.

Overleaf are two brand new arcade games, also on a special introductory offer, if you wish to take advantage of these - they are well worth it!

Sharp MZ-80B Users, sorry there isn't much on offer for you in this Issue, but there will be something in Issue 11.

That's all from us.

Happy computing.

THE SHARPSOFT TEAM

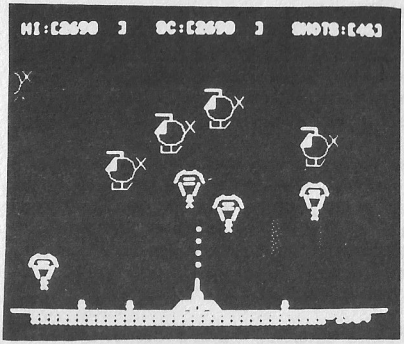
H I S O F T P A S C A L (A and K)

HISOFT have been producing PASCAL compilers for many years. Now they are proud to announce PASCAL 4, the lastest and most powerful of these implementations. Previous versions of HISOFT PASCAL have been in use with a wide variety of users for more than three years, so you can be sure that you are buying a proven, reliable piece of software. HISOFT PASCAL 4 is believed to be the most powerful and fastest PASCAL compiler available for microcomputers at such a low price.

HISOFT PASCAL 4 has been developed to provide an almost full implementation of Standard PASCAL as detailed in the PASCAL User Manual and Report by Kathleen Jensen and Niklaus Wirth (the originator of PASCAL). HISOFT have been careful to adhere to the definition of PASCAL as outline in this document while providing extra facilities to reflect the changing environment in which PASCAL is used.

PASCAL 4 will run on any system that has a Z80 microprocessor at its heart and at least 32K of RAM - the compiler has been written in Z80 and produces Z80 object code directly (no P-codes); this means that the compiler package, including an extensive line editor, occupies very little space in memory (less than 20K) and produces object code that runs very quickly indeed.

SEE OVERLEAF FOR MORE DETAILS ON HISOFT PASCAL

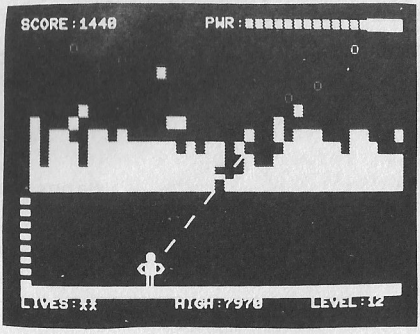


R A I D

The object of this nail biting, machine code game, is to prevent enemy forces parachuting from invading helicopters who are attempting to take over your guided missile control centre. Control the direction of the missiles to smash the choppers from the sky and kill the invading paratroopers before they land. Should too many land, the siege on your H.Q. will commence!!

P R O T E O U S I

This fantastic machine code program really fires the imagination and tests your skill to the full. The object of the game - SIMPLE - but the mission IMPOSSIBLE! Both adult and child alike will find this game absorbing and compulsive. PROTEOUS I progresses with 4 different levels of play with 4 waves in each level. If you enjoy arcade games don't miss it!



M E M B E R' S O R D E R F O R M

S.U.N. ISSUE NO. 10

Membership No: / / / /

NAME: .....

ADDRESS: .....

.....

.....

.....

(PLEASE COMPLETE IN BLOCK CAPiTALS)

Model MZ-80.....
A, K or B

PLEASE SEND ME:

		R.R.P.	MEMBER'S PRICE
HISOFT PASCAL			
MZ-80A, MZ-80K OR MZ-80B TAPE VERSION		40.25	**36.50**
MZ-80A, MZ-80K OR MZ-80B DISC (CP/M)		45.00	**40.50**
RAID	MZ-80A OR MZ-80K only	5.95	5.00
PROTEOUS I	MZ-80A OR MZ-80K only	6.95	5.50
		TOTAL	----- =====
S U P E R S P E C I A L	(A OR K users) only	**ALL 3 for**	**45.00**

I enclose cheque/PO made payable to SHARPSOFT LTD for .....

Charge my Access/Barclaycard No: _/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/

Expiry Date / /

Signature .....

OVERSEAS MEMBERS PLEASE ADD 2.00 POSTAGE

SEND TO: SUN SPECIAL OFFERS
SHARPSOFT LIMITED,
86/90 PAUL STREET,
LONDON, EC2A 4NE

The compiler supports all the data structures of Standard PASCAL (except files) including Arrays, Sets, Records, Pointers and user-enumerated types. Strings are supported as ARRAY (1..n) OF CHAR. Assignment of structured types, such as array to array assignment, is allowed and sets, strings and pointers may be compared. The CASE statement may optionally be terminated with ELSE. Procedures and functions may be fully recursive and procedures may take value or variable parameters.

I/O is provided through the standard procedures READ(LN) and WRITE(LN) integers, reals and strings may be read and integers, reals, strings and Booleans may be written. Formatting of output is supported through the use of write parameters; reals may be output in scientific form (mantissa, exponent) or fixed-point.

Records of any type may be created and accessed by field, optionally using the WITH construct for added convenience.

Dynamic variables may be created using the procedure NEW and accessed usinc pointers - this enables link-list structures to be created. Further, the procedures MARK and RELEASE allow control over the dynamic variable heap.

Interfacing with Z80 machine code is made easy through USER, PEEK, POKE, INLINE, SIZE and ADDR. PEEK and POKE may take any type as their arguments, except sets (for POKE). INLINE allows machine code to be included in the program at runtime. SIZE returns the amount of memory occupied by a variable and ADDR returns the address of a PASCAL program variable.

Interfacing to the Z80 ports is provided through the function INP(P) which returns the result of reading the Z80 port P and the procedure OUT(P,C) which outputs value C through the Z80 port P.

Variables may be saved to tape using the procedure TOUT and loaded back from tape using the procedure TIN. Note that variables of <u>any</u> type may be saved and loaded using these procedures.

The object code produced may save on tape, together with the runtimes, by a simple command and subsequently reloaded and executed inependently of the compiler thus saving memory space.

Many compiler options are available allowing, inter alia, output to be directed to a printer and runtime checks to be switched off for extra speed of execution.

For most systems the option 'F' is supported to allow inclusion of source text from tape at compile time - this allows large programs to be compiled as well as provided the flexibility of storing program segments on tape.

The compiler comes with extensive documentation in a 60 page manual with many example programs given.

FITTING SINGLE SIDED DRIVES INTO A SHARP DRIVE UNIT CASE

by

Peter Sydenham

Having purchased a pair of second hand Shugart SA 400 disc drives and a Sharp disc drive case, with power supply, it was decided to fit the drives into the case.

As soon as the drives were offered up to the case it was realised that they were mechanically interchangeable with the Sharp drives. Not only were the drives a perfect fit in the front panel but, even the fixing holes were correctly aligned. Two plates were made from 20 swg aluminium, to hold the drives together. A local ironmonger had some long No. 6 x 32 countersunk screws which were cut down and used to hold the plates to the drives. The heads needed to be only barely proud of the plates if they were not to foul on the sides of the front aperture. Some of the same screws together with 3/8" spacers were used to fix the lower drive to the base of the case. This completed the mechanical side of the job.

Checking the 4-way power plug showed that this was correct for the Shugart drives. Similarly, the 34 way printed circuit board edge connector, was pinned in the same way for both drives. The side select connector (pin 32) being open circuit on the Shugart drives. A 34 way ribbon cable was made up to connect directly between the Sharp disc controller card and the drives. The connectors used at the drives bore the part number RS 467-419. One mounting leg had to be cut off each socket to clear the other edge connector on the card. For correct connection the end of the connector nearest the edge of the Sharp controller board connects to the high numbers of the disk drive card. (If they are reversed the drives will be turning all the time and not respond to any commands). This then leaves the terminating resistors and links to be fitted.

The rule for terminating resistors is that only one drive must have then fitted and this must be the last drive in the chain. Thus if only 2 drives are in use they will be fitted to drive 2. The resistors are 150 ohms and may be in a DIL resistor pack or discreet resistors soldered to a header. A total of 5 resistors are needed, one each for:

	Track No.
WRITE DATA	22
WRITE GATE	24
MOTOR ON	16
DIRECTIONM SELECT	18
STEP	20

An IC socket at position IE was present on the drives for these resistors, with the spare places in the socket left disconnected. (Checking the track routing from the edge connector being worthwhile if in any doubt about the location of the correct socket).

A second socket on the disc drive card is marked for links to be fitted. Details in the Sharp Disc Unit Service Manual were followed for these connections, they are:

HS	OPEN	BOTH DRIVES
DSO	SHORT	DRIVE 1 ONLY
DS1	SHORT	DRIVE 2 ONLY
MX	OPEN	BOTH DRIVES
HM	OPEN	BOTH DRIVES

Once this had been done the drives connected to the Sharp controller card, a double sided Sharp Master Disc inserted and 'FD' typed.

Of course, as the drives were only single sided, the computer did not boot correctly. However, the code loaded into RAM was checked and was correct for even numbered tracks.

This now leaves the problem of a software system to run on the drives and that is the problem yet to be solved. To aid understanding of how the Sharp software manages double sided discs some of the code was disassembled and is presented here to give anybody who wishes to write their own DOS, a starting point.

It should of course be pointed out that had double sided drive units been available, these would have been used to make the Author's MZ-80K system up to a 4 drive system and such an option would be open to anybody who could buy double sided drives cheaply!

PAGE 1

```
                        ;INITIAL BOOT AT F000
                        ;THIS CODE IS IN ROM ON THE
                        ;DISC CONTROLLER CARD

                        ;WHEN FD IS TYPED EXECUTION
                        ;STARTS AT F000

                        ;27-4-83 P SYDENHAM
                        ;DISASSEMBLER BY R TANSWELL AND
                        ;'ZEN' ASSEMBLER USED FOR THIS
                        ;LISTING

                                ORG   0F000H
F000 00                         NOP
F001 F3                         DI
F002 AF                         XOR   A
F003 329C11                     LD    (119CH),A         ;CLOCK OFF
F006 3EC3                       LD    A,0C3H            ;JP CODE FOR ERROR TRAP
F008 320B10                     LD    (100BH),A
F00B 215AF0                     LD    HL,0F05AH
F00E 220C10                     LD    (100CH),HL        ;ERROR TRAP
F011 11F09F                     LD    DE,9FF0H          ;TRANSFER 9 BYTES FROM
F014 2187F0                     LD    HL,0F087H         ;ROM TO RAM FOR USE
F017 010900                     LD    BC,0009H          ;BY (IX+D) IN READER
F01A EDB0                       LDIR
F01C CD0900     IBT1:           CALL  CRLF
F01F 117AF0                     LD    DE,MESS1
F022 CD1500                     CALL  MESSAGE
F025 11009F                     LD    DE,BUFF2
F028 CD0300                     CALL  USER
F02B 210C00                     LD    HL,000CH          ;SKIP MESSAGE
F02E 19                         ADD   HL,DE             ;PICK UP ANSWERE TO
F02F 7E                         LD    A,(HL)            ;PROMPT
F030 FE0D                       CP    0DH               ;CR ?
F032 2002                       JR    NZ,0F036H         ;Z=CR ASSUME DRIVE 1
F034 3E31                       LD    A,31H             ;ASCII FOR 1
F036 47         IBT2:           LD    B,A
F037 E6F0                       AND   0F0H              ;TAKE ASCII AND CONVERT
F039 FE30                       CP    30H               ;TO NUMERIC HAVING
F03B 20DF                       JR    NZ,IBT1           ;CHECKED >1 & <=4
F03D 78                         LD    A,B
F03E E60F                       AND   0FH
F040 3D                         DEC   A
F041 FE04                       CP    04H
F043 30D7                       JR    NC,IBT1           ;DUD KEY TRY AGAIN
F045 32F09F                     LD    (9FF0H),A         ;KEEP DRIVE NO.
F048 321110                     LD    (1011H),A         ;AND AGAIN
F04B DD21F09F                   LD    IX,9FF0H          ;READY FOR DISC READ
F04F CD3BF1                     CALL  READER
F052 3A0098                     LD    A,(9800H)         ;LOOK AT 1ST BYTE
F055 FEC3                       CP    0C3H              ;JP CODE ?
F057 CA0098                     JP    Z,9800H           ;YES JP TO 9800H
F05A 31F010     IBT3:           LD    SP,10F0H          ;RESET STACK
F05D CD0900                     CALL  CRLF
F060 116CF0                     LD    DE,MESS2
F063 CD1500                     CALL  MESSAGE
F066 CDA7F0                     CALL  MOTOFF
F069 C38200                     JP    MAINLP            ;RETURN TO MONITOR

                        ;*********************************
```

PAGE 2

```
 61 F06C 45523A43 MESS2:    DB   "ER:CAN'T BOOT"
 61 F070 414E2754
 61 F074 20424F4F
 61 F078 54
 62 F079 0D                 DB   0DH
 63 F07A 424F4F54 MESS1:    DB   "BOOT DRIVE ?"
 63 F07E 20445249
 63 F082 5645203F
 64 F086 0D                 DB   0DH
 65 F087 0000     DDATA:    DB   00H,00H
 66 F089 0100               DB   01H,00H          ;TRACK 00 SECTOR 01
 67 F08B 0700               DB   07H,00H          ;0700H=NO. BYTES TO READ
 68 F08D 9800               DB   98H,00H          ;9800H=LOAD ADDR
 69 F08F 00                 DB   00H
 70
 71 F090 C5       MOTON:    PUSH BC               ;STARTS MOTORS
 72 F091 01F808             LD   BC,08F8H
 73 F094 ED78               IN   A,(C)            ;START MOTOR
 74 F096 010000             LD   BC,0000H
 75 F099 0B       WAIT1:    DEC  BC               ;WAIT FOR MOTOR TO
 76 F09A 00                 NOP                   ;GET UP TO SPEED
 77 F09B 00                 NOP
 78 F09C 78                 LD   A,B
 79 F09D B1                 OR   C
 80 F09E 20F9               JR   NZ,WAIT1
 81 F0A0 3E01               LD   A,01H
 82 F0A2 320210             LD   (MOTFLG),A       ;01=ON  00=OFF
 83 F0A5 C1                 POP  BC
 84 F0A6 C9                 RET
 85
 86
 87
 88 F0A7 C5       MOTOFF:   PUSH BC               ;STOP MOTORS
 89 F0A8 CDAEF1             CALL LNGDEL
 90 F0AB 01F800             LD   BC,00F8H
 91 F0AE ED78               IN   A,(C)
 92 F0B0 C1                 POP  BC
 93 F0B1 C9                 RET
 94
 95
 96 F0B2 CDBDF0   SKZERO:   CALL DREADY           ;SEEK TRACK 0
 97 F0B5 AF                 XOR  A
 98 F0B6 D3F9               OUT  (0F9H),A         ;CLEAR TRACK REG.
 99 F0B8 32E803             LD   (1000),A
100 F0BB D3FA               OUT  (0FAH),A         ;SEND SEEK ZERO CODE
101 F0BD C5       DREADY:   PUSH BC
102 F0BE 010000             LD   BC,0000H
103 F0C1 DBF9     DRY1:     IN   A,(0F9H)         ;GET DRDY,CRDY,RQM
104 F0C3 E603               AND  03H              ;LEAVE DRDY,CRDY
105 F0C5 FE02     DRY2:     CP   02H              ;WAIT FOR DRDY & CRDY
106 F0C7 2002               JR   NZ,WAIT2
107 F0C9 C1                 POP  BC
108 F0CA C9                 RET
109
110
111 F0CB 0B       WAIT2:    DEC  BC
112 F0CC 78                 LD   A,B
113 F0CD B1                 OR   C
114 F0CE 20F1               JR   NZ,DRY1
115 F0D0 C1                 POP  BC
```

PAGE 3

```
 116 F0D1 3E32              LD   A,32H          ;ERROR CODE
 117 F0D3 320810            LD   (1008H),A
 118 F0D6 C30B10            JP   100BH
 119
 120                ;--------------------
 121
 122 F0D9 DBFA      STATUS: IN   A,(0FAH)       ;READ STATUS
 123 F0DB E6F0              AND  0F0H
 124 F0DD 07                RLCA
 125 F0DE 30F9              JR   NC,STATUS      ;WAIT FOR CRDY
 126 F0E0 E6F0              AND  0F0H           ;MASK LEAVE CRDY,S1,S2,S3
 127 F0E2 0F                RRCA                ;MOVE RIGHT UNTIL S3
 128 F0E3 0F                RRCA                ;IS IN B0
 129 F0E4 0F                RRCA
 130 F0E5 0F                RRCA
 131 F0E6 B7                OR   A              ;CLEAR FLAGS
 132 F0E7 C8                RET  Z              ;Z=OK
 133 F0E8 FE0C              CP   0CH            ;0C=DRIVE NOT READY ETC.
 134 F0EA 2004              JR   NZ,STS1
 135 F0EC 3E32              LD   A,32H          ;ERROR CODE
 136 F0EE 180A              JR   STS3
 137 F0F0 FE04      STS1:   CP   04H            ;04=ID NOT FOUND
 138 F0F2 2004              JR   NZ,STS2
 139 F0F4 3E36              LD   A,36H          ;ERROR CODE
 140 F0F6 1802              JR   STS3
 141 F0F8 3E29      STS2:   LD   A,29H          ;ERROR CODE
 142 F0FA 320810    STS3:   LD   (1008H),A      ;KEEP ERROR CODE
 143 F0FD 37                SCF
 144 F0FE C9                RET
 145
 146
 147 F0FF C5        PRMDRV: PUSH BC             ;PRIME DRIVE
 148 F100 E5                PUSH HL
 149 F101 CD90F0            CALL MOTON
 150 F104 DD7E00            LD   A,(IX+00H )    ;GET DRIVE NO.-1
 151 F107 E603              AND  03H            ;FORM DRIVE CODE
 152 F109 F61C              OR   1CH            ;SET TND,MOTOR,SELECT BIT
 153 F10B 320110            LD   (1001H),A      ;KEEP DRIVE CODE
 154 F10E E60F              AND  0FH            ;MASK OUT TND
 155 F110 47                LD   B,A
 156 F111 0EF8              LD   C,0F8H
 157 F113 ED60              IN   H,(C)          ;SELECT DRIVE
 158 F115 3E32              LD   A,32H
 159 F117 CDAEF1    PRM1:   CALL LNGDEL         ;WAIT FOR HEAD
 160 F11A 3D                DEC  A              ;TO LOAD
 161 F11B 20FA              JR   NZ,PRM1
 162 F11D 010000            LD   BC,0000H
 163 F120 DBF9      PRM2:   IN   A,(0F9H)       ;GET DRDY,CRDY,RQM
 164 F122 E607              AND  07H            ;MASK OUT RUBBISH
 165 F124 FE06              CP   06H            ;DRDY & CRDY ?
 166 F126 2006              JR   NZ,PRM3        ;NZ=NO KEEP TRYING
 167 F128 CDB2F0            CALL SKZERO
 168 F12B E1                POP  HL
 169 F12C C1                POP  BC
 170 F12D C9                RET                 ;CORRECT EXIT
 171
 172 F12E 0B        PRM3:   DEC  BC
 173 F12F 78                LD   A,B
 174 F130 B1                OR   C
 175 F131 20ED              JR   NZ,PRM2
```

PAGE... 22

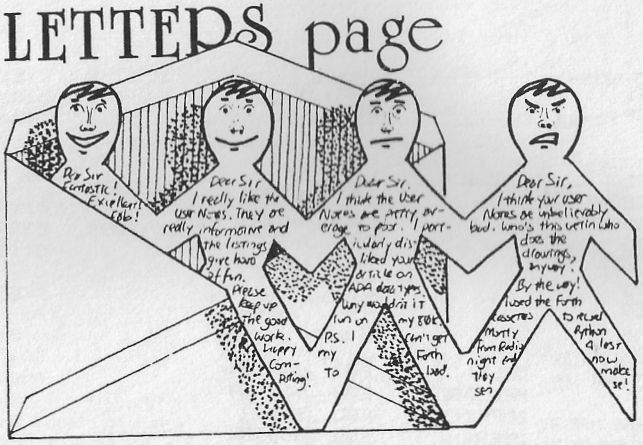
PAGE 4

```
 176 F133 3E32                  LD   A,32H              ;ERROR CODE
 177 F135 320810                LD   (1008H),A
 178 F138 C30B10                JP   100BH              ;ABORT
 179                    ;++++++++++++++++++++++++++++
 180
 181 F13B 3E0A          READER: LD   A,0AH              ;NO. OF TRIES
 182 F13D 320710                LD   (1007H),A
 183 F140 CDFFF0        RDR1:   CALL PRMDRV
 184 F143 3A0110                LD   A,(1001H)          ;KEEP DRIVE IN USE
 185 F146 47                    LD   B,A
 186 F147 0EF8                  LD   C,0F8H
 187 F149 D9                    EXX
 188 F14A 0EFB                  LD   C,0FBH
 189 F14C DD5E03                LD   E,(IX+03H )        ;GET LENGTH TO READ
 190 F14F DD5604                LD   D,(IX+04H )        ;IN BYTES INTO DE
 191 F152 CB13                  RL   E
 192 F154 CB12                  RL   D                  ;FORM NO. OF SECTORS IN D
 193 F156 1E03                  LD   E,03H              ;FOR MASK
 194 F158 DD6E05                LD   L,(IX+05H )        ;GET LOADING ADDR.
 195 F15B DD6606                LD   H,(IX+06H )        ;INTO HL
 196 F15E CDBDF0                CALL DREADY
 197 F161 AF                    XOR  A
 198 F162 DD7E01                LD   A,(IX+01H )        ;GET TRACK TO READ
 199 F165 1F                    RRA                     ;DOUBLE IT
 200 F166 D3F9                  OUT  (0F9H),A           ;SEND TRACK TO FDC
 201 F168 DD7E02                LD   A,(IX+02H )        ;GET SECTOR TO START
 202 F16B 3002                  JR   NC,RDR2
 203 F16D F680                  OR   80H                ;ODDS/EVENS FOR SIDE CODE
 204 F16F D3F8          RDR2:   OUT  (0F8H),A           ;SEND SECT+SIDE
 205 F171 CDA6F1                CALL SHTDEL             ;SHORT DELAY
 206 F174 3E70                  LD   A,70H              ;SEEK & READ CODE
 207 F176 320010                LD   (1000H),A          ;KEEP IT
 208 F179 F3                    DI
 209 F17A D3FA                  OUT  (0FAH),A           ;SEND S&R CODE TO FDC
 210 F17C 0680          RDR3:   LD   B,80H              ;BYTES/SECTOR
 211 F17E DBF9          RDR4:   IN   A,(0F9H)           ;GET DRDY,CRDY,RQM
 212 F180 A3                    AND  E                  ;MASK WITH 03
 213 F181 28FB                  JR   Z,RDR4             ;WAIT FOR EITHER CRDY/RQM
 214 F183 0F                    RRCA                    ;RQM INTO CARRY
 215 F184 300C                  JR   NC,RDR5            ;NC=NO RQM
 216 F186 EDA2                  INI                     ;TAKE IN BYTE
 217 F188 C27EF1                JP   NZ,RDR4            ;DO WHOLE SECTOR
 218 F18B 15                    DEC  D                  ;DEC SECTOR COUNTER
 219 F18C C27CF1                JP   NZ,RDR3            ;NZ=MORE TO DO
 220 F18F D9                    EXX
 221 F190 ED78                  IN   A,(C)              ;SEND TND HIGH
 222 F192 CDD9F0        RDR5:   CALL STATUS
 223 F195 D0                    RET  NC                 ;NC=GOOD READ
 224 F196 3A0710                LD   A,(1007H)
 225 F199 3D                    DEC  A                  ;A TRY GONE
 226 F19A 320710                LD   (1007H),A
 227 F19D CA0E10                JP   Z,100BH            ;CAN'T READ AT ALL ABORT
 228 F1A0 CDB2F0                CALL SKZERO
 229 F1A3 C340F1                JP   RDR1
 230
 231
 232 F1A6 F5            SHTDEL: PUSH AF
 233 F1A7 3E0A                  LD   A,0AH
 234 F1A9 3D            SDY1:   DEC  A
 235 F1AA 20FD                  JR   NZ,SDY1
```

PAGE... 23

PAGE 5

```
 236 F1AC F1                    POP  AF
 237 F1AD C9                    RET
 238
 239
 240 F1AE F5        LNGDEL:     PUSH AF                    ;LONG DELAY
 241 F1AF 3E0A                  LD   A,0AH
 242 F1B1 CDA6F1    LDY1:       CALL SHTDEL
 243 F1B4 3D                    DEC  A
 244 F1B5 20FA                  JR   NZ,LDY1
 245 F1B7 F1                    POP  AF
 246 F1B8 C9                    RET
 247
 248
 249                CRLF:       EQU  00009H
 250                MESSAGE:    EQU  00015H
 251                BUFF2:      EQU  9F00H
 252                USER:       EQU  00003H
 253                MAINLP:     EQU  00082H
 254                MOTFLG:     EQU  1002H
 255                            END
```



Dear Sharpsoft,

I refer to the member's Letter on page 37 in Issue No.7 - 1983 from B. Harvey, Police Officer of Lincolnshire and would wish to ascertain whether the "ULTIMUM" Interface System from Watford Electronics can be used to upgrade the MZ-80K to 64K of memory.

Would you kindly put me in touch with this member who wrote in response to an earlier member's letter published in Issue No. 6 for advice in addressing 64K on more of RAM on the SHARP MZ-80K.

Watford Electronics appear to have been supplied with full circuit diagrams of the MZ-80K. I would like to have first hand information from Mr. Harvey of his experience with this project.

Could you also let me know of any other commercial attempts to increase the RAM size above 48K..

J.H. MERES
HEREFORDSHIRE

It appears from our List that Mr. Harvey is no longer a Member and unfortunately, due to shortage of storage space, we only retain the original of Reader's Letters from the last Issue. Therefore, if you are out there Mr. Harvey, please let us know the results of your project?

SHARPSOFT

****

Fellow Members,

In a REM statement in the beginning of each listing, it would be a great help if the exact version of the language being used were added.

MR. WARD
MEMBER 0148

Dear Sharpsoft,

I would appreciate your help in solving my problem.

1. The Prog(s) sent in are the same except line 10004-5. Prog 2 was done just to see if it make any difference. **It does not.**

2. The Prog was written using Sharp Disk BASIC. **Line 10004 shows SYNTAX ERROR.**

3. The Prog was rewritten using Sharp Disk BASIC + Knights Commander. **Same result**

4. The Prog was rewritten using Tape BASIC SP-5025 **Same Result**

5. The Prog rewritten using Speed BASIC **Guess What - It worked!!**

I can't see why the Prog should not work on Disk or Std BASIC (A) Could I have corrupted Disk & Tape BASIC? (B) Could my Disk Drive /CPU be faulty? I need this type of routine for a new Adventure Game I am writing and I would appreciate your early reply.

```
10000 PRINT"[Q]"
10002 INPUT"WHAT IS TIME:"A1$
10004 TI$=A1$:IFTI$<="065959"THENTI$="070000"
10006 PRINT"[C]"
10007 IFA1$<TI$THENGOTO1100
10009 PRINT"[H]"
10010 PRINTTAB(17);LEFT$(TI$,2);":";MID$(TI$,3,2);":";RIGHT$(TI$,2)
10020 GOTO10009
11000 PRINT"[↓↓↓↓↓↓] ";TAB(15);"WHOSE SLEPT IN ?"
11001 REM MUSIC
11002 REM MUSIC
11003 FORG=1TO20 :NEXT :GOTO10009
12000 END
```

```
10000 PRINT"[C]"
10002 INPUT"WHAT IS TIME:"A1$
10004 TI$=A1$
10005 IFTI$<="065959"THEN TI$="070000"
10006 PRINT"[C]"
10007 IFA1$<TI$THENGOTO1100
10009 PRINT"[H]"
10010 PRINTTAB(17);LEFT$(TI$,2);":";MID$(TI$,3,2);":";RIGHT$(TI$,2)
10020 GOTO10009
11000 PRINT"[↓↓↓↓↓↓]";TAB(15);"WHOSE SLEPT IN ?"
11001 REM MUSIC
11002 REM MUSIC
11003 FORG=1TO20 :NEXT :GOTO10009                    G.F. GATISS
12000 END                                            GLOUCESTER
```

Standard Sharp BASIC's do not support the utility to test for string inequalities, they can only test for a string equality. Speed BASIC on the other hand adds a string inequality function to SP5025 BASIC, thus your program runs correctly. The way round this problem when using a standard Sharp BASIC is to convert the TI$ string into a numerical variable.

```
10002 INPUT"WHAT IS THE TIME?";TI$
10004 T=VAL(TI$):IFT<65959 THEN TI$="070000"
```

Pages 56 and 83 of the Orange Sharp BASIC Manual for the MZ-80K and pages 48 and 75 of the MZ-80A Manual should make this clearer.

SHARPSOFT

****

Dear Sharpsoft,

In response to your request for contributions from MZ-80A Users, I am including a few tips which you may like to pass on to other users of BASIC SA-5510.

POKE $19F7,$00	Allows a comma to be entered in an INPUT string. Poke$19F7,$2C restores to normal.
POKE $2C4C,$00 POKE $2CAB,$00	Allows (CR) or (ENT) alone to be pressed in an INPUT string without BASIC starting a new line with a ? and waiting for a fresh input. Poke $2C4C,$0D : Poke $2CAB,$0D restores to normal.
POKE $1EA7,$69 POKE $1E88,$1E	Allows an expression to be used following a GOTO statement. E.G. GOTO X*100. Poke $1EA7,$F6 : Poke $1ECC,$17 restores to normal.

For users with Disk BASIC SA-6510 the following are the equivalent of the above.

POKE $1BBA,$00	Allows a comma to be entered in an INPUT string. Poke $1BBA,$2C restores to normal.
POKE $313C,$00 POKE $3266,$00	Allows a null entry in an INPUT string. Poke $313C,$0D : Poke $3266,$0D restores to normal.
POKE $208F,$48 POKE $20C0,$20	Allows an expression to be used following a GOSUB statement. Poke $20BF,$B9 : Poke $20C0,$19 restores to normal.

Congratulations on your excellent magazine, I eagerly await the 1984 editions.

P. RAWSON
MANCHESTER

****

Dear Editor,

Your help information, literature, 'Users' has in 12 months enabled me to use my MZ-80K to some effect.

Many years ago I knew that even a thermionic valve could be used as a switch, Transitors, since, make switching easier.

The development to the use of the binary into a decimal language and then through hexadecimal onto structured languages is basically easy to understand.

The difficulty of putting this into practise is somewhat frightening. Yet when in the S.U.N. you see a program "made" by a 10 year old those fears you'll never understand, go.

The child's open mind proving it can be done easily. So we only have to open ours.

More of these youngsters' programs, please, they not only give us incentive, but give us some hope for the future.

I hope this is not too long, but to me it has value and could help many, struggling to come to terms with modern technology.

S. FROST
LINCS.

****

Dear Sir,

You might be interested to know that I have made very good use of PA-MON to relocate SHARP-5060 BASIC, and Hi-soft PASCAL 3, and interface them to FDOS. The ability to compile BASIC code directly, without learning FDOS commands, has proved very useful. If readers are interested, I could give a fuller description of what I've done.

J.F. MACQUEEN
SURREY

Yes please. Readers are always keen to see fellow Members' ideas and techniques. We look forward to seeing it in a future Issue.

SHARPSOFT

****

Dear Sir,

Why do you print letters such as that from Mr. Barnes on Page 45 of the Autumn 1983 S.U.N. (only recently received) without publishing the answers to queries raised? I can see no point in printing them unless to "pad out" the issue.

If you also published the answers then those of us who might have similar queries would also have the answers.

I hope, no double correctly, that you did give Mr. Barnes a speedy answer and, of course, I am not criticising him as I can appreciate the reason for his enquiry but why not let me know the answer as well?

C.C. ALGAR
LANCS.

T O R E F R E S H Y O U R M E M O R I E S
LETTER FROM MR.W.W. BARNES FROM S.U.N. AUTUMN 1983

I have your Sharp MX-80K Computer and am a Raw Amateur at the moment. I have produced a copy of the BASIC SP5025 tape, and have added the speed BASIC toolkit.

I have a problem with the ?@ command.

When I type in ?@4,2;"K" it correctly performs the function required, but it will not list. When I give it a line number, say 100, and type in 100 ?@4,2;"K" it will not perform the function, but it will list. (The result is I cannot transfer to a fresh tape).

I have your book Peeking and Pokeing the Sharp MZ-80K and have experimented with the 2 programs on page 16 and the one on page 17. The program at the bottom of page 16 reveals a syntax error in line 50, and the program on page 17 reveals a syntax error in Line 110. I have no success with the program with line nos: 1 2 3 4 5 6 7 8 on page 16.

I have your Sharpsoft Notes 1 to 7 and various other books purchased from you but I am afraid I cannot find any solution.

W.W. BARNES

When we received Mr. Barnes' letter we had no ready answer to his problem but sent it on to the Author of the Peeking & Pokeing Books, Mr. G.P. Ridley who has since kindly sent this reply.

SHARPSOFT

Dear Sharpsoft,

In reply to W.W. Barnes, and in defence of my book Peeking and Pokeing the MZ-80K, I can perhaps throw some light onto the PRINT@ confusion.

The PRINT@ mod on page 16 is intended to be added to the unmodified SP-5025 BASIC, which works very well. If the reader has purchased a copy of the Speed BASIC toolkit one assumes that it came with an instruction sheet which if he looks on the other side will see that this toolkit already contains a PRINT@ routine, so there seems little point in adding a further one.

These additional routines, whether entered directly from the keyboard or loaded from a tape containing a toolkit, alter the contents of the SP-5025 so that one can enter these extra commands and the interpreter will accept them without giving a Syntax error. The listing on page 16 does exactly that, it alters memory. SPEED BASIC contains several additional commands, not just?@, in fact PRINT can be followed by a few different characters for altering the printing effect to both screen and printer. Obviously altering memory has to be exactly correct for these functions to work, so adding a toolkit and altering memory is fine but then re-altering it will probably mean that some commands may not work.

In fact if the reader loads SP-5025 in the normal way and then loads the SPEED BASIC and follows this with the routine on page 16 he will find it will work, saving a copy of the altered BSIC with the toolkit, and so will the demonstration listings on apges 16 and 17, but doing it the other way, entering the routine first followed by the toolkit, it won't. The only difference appears to be in the parameters which follow ?@. SPEED BASIC requires the column number first and then the line number on which to print. My routine is the opposite in that the line number is first followed by the column, which explains the syntax error in line 110, as the second parameter would be out of range as SPEED BASIC likes it to be in the range 0-24.

I hope this has helped someone somewhere.

G.P. RIDLEY
AUTHOR PEEKING AND POKEING THE MZ-80K/MZ-80A/700

In the interim we received another reply to Mr. Barnes' problems as follows:

Dear Sharpsoft,

1. Re ?@ No problem 100 ?@4,2;"K" will list and if you type RUN it will produce the K on screen.

2. There is a small problem with "Peeking and Pokeing the SHARP MZ-80K" and the use of SPEED BASIC.

SPEED BASIC) use X for COLS 0-39
and)
KNIGHTS COMMANDER) Y FOR LINES 0-24

whereas print @ in the book uses X and Y the other way round. This does not appear to be a convention.

For use with programs in the P & P Book I have produced SP-5025 + Print @ per lines 1-8, although normally I use SP-5025 + SPEED BASIC without problems! Lines 1-8 if correctly entered and added to SP-5025 (without SPEED BASIC) WILL WORK.

I suspect your SYNTAX on line 50 P16 and line 100 P17 is due to use of SPEED BASIC. Have just checked theprogram don P17 does work if run using SP-5025 + PRINT@ Lines 1-8.

Other Programs in P & P Book:-

3. ALIENS - OK if line 370 changed to POKE 53627,191
and line 670 changed to POKE 53627
4. HANGMAN - OK

5. P29 at bottom D = EAST

6. CONNECT FOUR - OK but again coordinates wrong for SPEED BASIC

7. P42 - OK and P43

8. P46-53 - OK using P & P ?@ For printer change to
PRINT/P line 760-70 : 840-870.
INSERT " on line 1120 after ↑↑↑"

Trust this puts you on the right lines, I had difficulty until I rumbled the co-ordinate problem.

E.A.A. KIMBER
MIDDX.

Many thanks to Mr. Kimber for showing the true spirit of Membership for, as we all know:- "A Member in need is a Member indeed".

SHARPSOFT

****

NOTE
Poke listed by N. WYLDE on P41 of Issue 9 can be POKE 5414,100 or 5414,7.

To muddle listed programs. To revert back to normal listing: POKE 5414,126.

R.W. NEWTON
STH. HUMBERSIDE

Dear Sharpsoft,

A tip for SUN worshippers; the PRINTSIZE command can be split if a note of the memory left is required during processing. The following example will do the trick:-

```
PRINT"Size of memory left is ";SIZE;"bytes."
```

This enables an eye to be kept on the possibility of memory overflow where a large program is being run, and the data is being added to.

J. QUARMBY
THREE COUNTIES STEAM PRESERVATION SOCIETY
SURREY

****

Dear S.U.N.,

In reference to J. Kuttes claim to the highest score on Asteroids m/c game, I would like to claim a score of 56,190. I have never bothered to write before because I though it would have been beaten. I have 1 witness to my claim. Could you ask the other S.U.N. Asteroid players if they can beat this score?

ZEE GATISS (Age 11)
GLOUCESTER

Has Zee Gatiss the top score in playing Asteroids? Not according to B. Hallett on page 34. Let us know if anyone has even higher scores.

SHARPSOFT

****

Dear Sharpsoft,

In writing, I am enclosing a program that explains how to send control codes to the MZ-80BP6 printer by giving an example of the control codes for setting up line spacing (control codes are ESC + 10H + n which are listed in page 17 in the owners manual). The example used is considerably more flexible that the PRINT/P CHR$(17) mode and P6 users may wish to experiment with the various values of n and to adapt the program to take advantage of the other control codes which can improve the capability of the P6 printer.

```
1   REM Program to set up Line Spacing on the MZ-80BP6 Printer
2   REM Control Codes are ESC + 10H + n
3   REM By K F Francis & E. Bremner (Hong Kong)
5   CLR:R=1
10  CONSOLEN,C80:PRINTCHR$(6)
20  ON R GOSUB 230,350
30  INP@254,A          :REM check printer status (ie printer on)
40  IFA<>242 THEN 30   :REM and hold until printer on
50  FOR J= 1 TO 3
60  OUT@255,27         :REM send ESC (27 decimal) to data out
70  OUT@254,128        :REM set RDP to high level so printer will
                        read data
80  GOSUB 330
90  OUT@255,16         :REM send 10H (16 decimal) to data out
100 OUT@254,128        :REM set RDP to high level so printer will
                        read data
110 GOSUB 330
```

```
120 OUT@255,LS          :REM send n(line feed pitch number)to data
                         out
130 OUT@245,128         :REM set RDP to high level so printer will
                         read data
140 GOSUB 330
150 PRINT/P "       Line spacing is now set to :";LS;"       Line
                         spacing is now set to :";LS
160 NEXT J
170 FOR P=1 TO 5 PRINT/P:NEXT P:MUSIC"2BRO":PRINTCHR4(6)
180 CURSOR29,10:PRINT"TRY AGAIN ? (Y/N)"
190 GET A$:IF A$="" THEN 190
200 IF A$="N" THEN PRINCTCHR$(6):END
210 IF A$="Y" THEN R=2:GOTO 10
220 GOTO 180
230 CURSOR13,4:PRINT"INITIALISATION OF LINE SPACING FOR THE MZ-80
    P6 PRINTER"
240 PRINTTAB(13);"-----------------------------------------------"
250 CURSOR 24,9:PRINT"ENTER ANY NUMBER FROM 0 TO 255"
260 CURSOR 18,12:PRINT"(20 WILL COMPLETELY CLOSE UP LINE SPACING)"
    :PRINT
270 PRINTTAB(21);"(AND 75 WILL PROVIDE DOUBLE SPACING)"
280 CURSOR20,20:INPUT"LINE SPACING REQUIRED : ";LS:PRINTCHR4(6)
290 IF (LS>255)+(LS<0) THEN 280
300 CURSOR15,10:PRINT2ENSURE PRINTER IS ON AND ENTER ANY KEY
    TO CONTINUE"
310 GET A$:IF A$="" THEN 310
320 PRINTCHR4(6):CURSOR21,10:PRINT"INITIALISING LINE SPACING TO :
    ";LS:MUSIC"2BRO":RETURN
330 OUT@254,00          :REM reset RDP to low level to receive next
                         code
340 RETURN
350 CURSOR20,10:INPUT"LINE SPACING REQUIRED : ";LS:PRINTCHR4(6)
360 IF (LS>255)+(LS<1) THEN 350
370 GOTO 320
```

K.F.FRANCIS
HONG KONG

****

Dear Sharpsoft,

For months I have bemoaned the neglect of the MZ-80B by the magazines. Imagine my joy when I was given Mike Brinson's book for Christmas to read of SHARPSOFT USER NOTES.

Now that the B.B.C. have started their CHIP SHOP and are broadcasting programs in BASICODE for home micros can you please do something about a BASICODE cassette for the MZ-80B? Its infuriating that they are providing it for the 80A but not the 80B!

I have had to wrote most of my own programs for the 80B and have one or two which might interest other 80B fans. Having produced a full screen machine code LIFE, I noticed that it naturally produced, occassionally, a group of 'cells' which, by changing its form in a cyclic manner, moved about the screen until it met other 'cells'. Until then my wife had found LIFE only mildly interesting. But this was different. So I have no added the facility to create one of these 'amoeba' forms at the touch of a key, though the location and direction of travel are determid by theprogram in a 'random' manner. My wife now says it is my most interest program! She used to prefer my simulation of the orbital motions of the inner planets of the Solar System (with our Moon of course). Thats only in BASIC so far.

In appreciation of Mike Brinson's book (A Practical Guide to Graphics on the MZ-80B) I enclose for his and your interest an 'improvement' I find rather useful. It is self explanatory but its ease of manipulation only becomes apparent when you try it.

Finally my only criticism of the book. The diagram on page 70 does not help one to undersand the rotation of the Point P. Can it end up at P' by a rotation? Even my wife (about to embark on the O.U. honours degree courses in maths) finds it confusing. We think itdetracts from the book becauseof this. Please understand, I only mention this in the hope of being helpful. Thank you all for a most interesting book. I look forward to more like it.

PERSPECTIVE DRAWINGS. BY M.E.BRINSON

Addition "Continuous Rotation" by M.FRY Jan 1984.

In this printout "continuous" rotation is given in steps of 15 degrees for Heading, Pitch, or Bank with instant change-over as required to any one of these parameters or (by keying the .) to the Command mode. The size of the step is easily changed in lines 1070/1080. The + and - keys give instant redrawing, with the addition or subtraction of the programmmed step to the currently selected parameter. This mode of operation is selected by an additional Command added in line 930.

```
920 IFCO$="L"THENGOSUB 9200:GOTO 500
930 IFCO$="@"THEN 1000
940 GOTO 500
1000 CURSOR0,24:FOR I=0 TO 78:PRINT" ";:NEXT
1001 CURSOR0,24:PRINT"ROTATE H,P,B,or. ? >>";
1002 GETA$:IFA$=""THEN 1002
1003 CURSOR25,24:PRINTA$
1010 IFA$="H"THEN RR=HE:GOTO 1060
1020 IFA$="P"THEN RR=PI:GOTO 1060
1030 IFA$="B"THEN RR=BA:GOTO 1060
1040 IFA$="."THEN 500
1050 GOTO 1000
1060 CURSOR0,23:FOR I=0 TO 78:PRINT" ";:NEXT
1061 CURSOR0,24 :PRINT"ROTATE ";A$;"  + or - ?>> ";
1062 PRINT"  HEADING= ";HE*180/π;"  PITCH= ";PI*180/π;"  BANK= ";BA*180/π
1063 GETB$:IFB$=""THEN 1063
1070 IFB$="+"THEN RR=RR+π/12:GOTO 1150
1080 IF B$="-"THEN RR=RR-π/12:GOTO 1150
1090 IF B$="H"THEN A$="H":GOTO 1010
1100 IF B$="P"THEN A$="P":GOTO 1020
1110 IF B$="B"THEN A$="B":GOTO 1030
1120 IFB$="."THEN 500
1130 GOTO 1060
1150 IFA$="H"THEN HE=RR:GOSUB 30000:GOTO 1060
1160 IFA$="P"THEN PI=RR:GOSUB 30000:GOTO 1060
1170 IFA$="B"THEN BA=RR:GOSUB 30000:GOTO 1060
```

M. FRY
EASTBOURNE

Many, many thanks for your kind letter, it is surely uplifting when anyone takes such a keen, lively interest in computing. Please send us more about your fascinating projects.

EDITOR

Dear Sirs,

After typing in several games programs from yours, and other magazines, I have found that sometimes, routines can be speeded up quite considerably, leading to greater excitement. I therefore enclose a few tips for future game writers who have not yet discovered them.

1. It is often advisable to use variables as often as possible, these are processed quicker than numbers in many cases.

2. Instructions and other routines not used during the action part of a game should be placed near the end of a program, as the interpreter "reads" through all the line numbers in order when searching for the line after a GOTO or GOSUB routine.

3. A useful and quick routine for reading the standard W,A,D,X direction keys from the keyboard is as follows:

```
10  POKE10407,0:T=22:X=53248
20  GETX$:IFX$=""THEN80
30  IFASC(X$)>T THEN80
40  IFX$="A" THEN X=X-1:GOTO80
50  X=X+1:GOTO80
60  IFX$="W"THEN X=X-40:GOTO80
70  X=X+40
80  REM REST OF PROGRAM
```

This routine, although allowing other keys to be pressed unchecked, means eacy key depression passes through 3 IF...THEN statements, and so each direction of travel of the object at position X is about as fast as another. Also, if no key is depressed, this is detected early, and the program can quickly get on with moving aliens etc.

I hope these notes may prove useful for those who have just begun to write games, and hopefully their game's speed may be improved.

P.S. I regret to inform Master Kutte, whose letter was published in User Notes 9, that I have, as reported in Notes Issue 5 attained a score of over 67,000 at Asteroids.

B. HALLETT
SOMERSET

****

Dear Sharpsoft,

Thank you for your phone call, and your very prompt (only one day after you estimated!) action with the parcel. That is DEFINITELY what I call service!

Please find enclosed cheque for the supply of DYBUGR and 80 Column Conversion Kit which I would like to receive with similar promptness! Other orders, for other software will undoubtably follow.

I have looked briefly at the back issues with interest, fired up the DYBUGR, but the rest will have to wait. I have noted that there aren't any DEstructions with the RESOURCE DISK, should there be?

I have, as you can probably tell, got a Word Pro, plus various other software, but I am very interested in a Disk Based Spreadsheet, can you supply one, or do you know of one that I can purchase for my micro? (CP/M or other!)

Is it possible for you to supply me the addres of any other MZ-80A users in my area (BRISTOL-AVON)? If this is not possible would you print my Name and Telephone Number., expressing an interest in starting a local User Group (Ken..0272-719952 evenings).

My faith in you and your organisation has been boosted tremendously, and I will be happy to renew my subscription for next year.

I wish you every success with S.U.N. in the future.

KEN ROOK
232 WICK ROAD
BRISTOL BS4 4HN

Unfortunately there are no DEstructions for any of the CP/M Volumes as these are classed as free public domain software programs. At this point in time there are no Disk or Tape based Spreadsheet Programs that we can recommend for the MZ-80A. However we are hoping that this will be rectified shortly! We will notify all our Members of any developments on this front. Finally, we wish you every success in organising a local Group, as we will definitely be expecting one or more BRISTOL GROUP Contributions, after all, we'll be its Godparent!

SHARPSOFT

ADDRESS YOUR SHARPSOFT USER NOTE CONTRIBUTIONS TO:

S.U.N. LETTERS
SHARPSOFT LIMITED
86-90 PAUL STREET
LONDON, EC2A 4NE

TEL: 01-739-8559

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