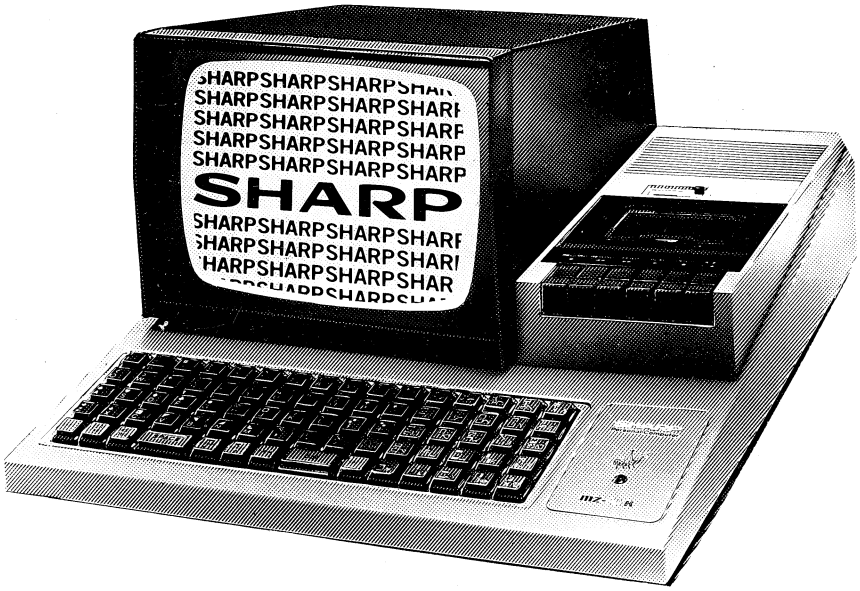
PEEKING and POKEING

THE

SHARP MZ-80K



A Beginners Guide to;

STYLING YOUR PROGRAMS

PEEKS and POKES

VIDEO RAM AREA

DATA STORAGE on TAPE

plus PROGRAMS

by G.P. Ridley

PEEKING and POKEING THE SHARP MZ-80K

First published 1982
Second impression 1982

PUBLISHED BY:-
D.C.BRENNAN Eng.
14 North Western Avenue
Watford Herts. WD2 6AE

Foreword

In this book I hope to aid the beginner and average hobbyist, in progamming and understanding the SHARP MZ-80K.

All the popular microcomputers on sale today besides their operating manual, which is supplied at time of purchase, have other publications written about them to help the beginner to computing extend his knowledge of that particular micro. If one visits computer exhibitions or bookshops he will see that several books abound for all the popular makes of home computers with the exception of one, the SHARP MZ-80K.

There appears to be a host of books aimed at those higher up the ladder of knowledge regarding home computers, but surprisingly little written for those who have read their BASIC manuals and wish to climb that same ladder but simply need a leg up. I hope this book will go a little way at least to help you on that first rung, and maybe other books will follow based on the MZ-80K.

Throughout this book references will be made to the BASIC manual supplied with your SHARP, so please keep it handy.

The cursor control characters used in listings throughout this book will be shown as follows:-

[C]	will be clear screen and cursor to top left
[H]	" " cursor home without clear screen
[↓]	" " cursor down
[↑]	" " cursor up
[←]	" " cursor left
[→]	" " cursor right

Contents

1
ROM & RAM

Some knowledge of how a computer works is helpful if one is going to understand something about PEEK's and POKE's that one sees written in programs. If you find this chapter is heavy going the first time round, do not be put off as you can always refer back to it later.

As you know the computer's memory is made up of a number of bytes, some of which are in ROM, and the remainder in RAM. On the Sharp MZ-80K the Monitor SP1002 is in ROM.
ROM stands for Read Only Memory, and that is precisely all the computer can do with it, read the instructions contained within the area that the ROM resides.

On the MZ-80K, when you switch on, the screen displays **MONITOR SP-1002** , this ROM takes up 4 Kilobytes of memory, which is equal to 4096 bytes.

Now that the computer is switched on, you LOAD your BASIC tape, all 14 Kbytes of it. This loads into RAM, which stands for Random Access Memory, which unlike ROM means that not only can the computer read what is in this part of memory, but it can be changed to suit ones needs.

This loading of Basic each time one switches on the Micro may appear to be laborious, especially as you do not have to do it with other popular Micros as their ROM contains all the Basic language instructions already. But bear in mind that it cannot be altered easily, as in our case with the Sharp.
If you want to add extra instructions, as in the case of the "PRINT @" statement described later on, or want to run in a different language other than Basic, such as FORTH, PASCAL, FORTRAN etc., or simply want to use one of the many TOOLKITS which are available, you have to physically take out or add chips to the printed circuit board within the computer. And then you can only add what is commercially available, at extra cost, each time. You certainly cannot add what you want to, only what others would like you to.

If you really get hooked on computing and would like to write programs using another language, all you have to do is LOAD a tape containing that particular language. There is no need to change parts of your Sharp, it is simply written into the RAM area of memory, which is empty and waiting each time you switch on. So do not become anxious about the 1½ minutes it takes to load Basic each time, because you have a certain advantage over other users of micros.

Perhaps a word on "TOOLKITS" is called for. Unlike their name suggests, these are not mixture of small screwdrivers and pliers etc., but tapes of useful additional commands added to the Basic ones you already have. These tapes are loaded after Basic, and usually you can SAVE both the BASIC and the TOOLKIT back on to one tape, as this then means you only load one tape at switch on which would contain both the Basic and the extra commands in TOOLKIT.

Some of the additional commands that are available are as follows:-

APPEND. Allows you to join one program to another, providing the second is higher line numbered.

AUTO. This automatically numbers your program as you write each line. The starting line number and the increments can be altered. 100,10 would start numbering at line 100 and continue with 110,120 etc.

RENUMBER. You probably know that it is good practice to number your lines in increments of 10, so as to leave space for additional lines to be squeezed in. But even then you may find that you need more lines than you have room for. With this command you can RENUMBER at any time such as:- RENUMBER100,10 will renumber the program starting at 100 and incrementing in 10's. RENUMBER200,10 would start at line 200.

Even when you have finished writing your program and debugged it, the line numbers may not be evenly incremented, so a quick RENUMBER, and it is very quick, will make it look professional and neat.
TRACE. While debugging, which we all have to do, some more than others, a useful function is the TRACE command. This slows down the running of the program and displays the current line number on the screen. You can see where you are in the program at all times.

Most Toolkits contain many more useful functions than I have mentioned, and are certainly worth considering, as it makes the programmers life a lot easier.

The layout of all the bytes in your MZ-80K is listed on page 118 of the Sharp Manual, it's called the Memory Map. If you look at the map you will see numbers down the left side of the diagram. These numbers are in Hexadecimal, which if you did not know, are numbers to the base of 16. The everyday numbers we normally use are Decimal, to the base of 10. I won't dwell too much on this, as throughout this book whenever a Hex number is mentioned the Decimal equivalent will be shown. There is a conversion table in the Appendix at the end of the book, and it will certainly help you if you can get acquainted with this system of numbering, especially if you intend to go on to using Assembler Language. Nevertheless it will not stop you understanding this book if you feel it is too complicated at this stage.

As you will see, the first section of the Map is for the bytes numbered 0000 to 1000 Hex. (4096 decimal), these bytes are in ROM and is where the Monitor sits in memory, and is coupled to the next area from 1000 to 1200 Hex. (4608 decimal), which is the work area for the Monitor. These two sections spring into life as soon as you switch on, and are the main control areas for whatever language you are using.

The third section is from 1200 to 6000 Hex. (24576 decimal), is a little vague, as it should be shown as three differing sections. In actual fact the Basic tape when loaded, occupies memory from 1200 to 4200 Hex. (16896 decimal), and it's work area from 4200 to 4805 Hex. (18437 decimal).

Your programs in Basic, either those you type in directly or ones loaded from tape, start loading from the end of the work area at 4806 Hex (18438 dec) and grow upwards. If your memory is 48Kbytes, this area extends up to CFFF Hex 53247 (dec).
Above this figure is the Video Ram area which is used for mapping out the screen. In a lot of programs which contain fast moving graphics, you will see a lot of POKEing going on in the area between 53248 and 54247 Decimal.
Above these areas are sections which deal with Input and Output controls and the Disc operating system.

On page 119 of the manual is a brief description of linkage to Machine language. It actually looks quite confusing so we shall try to clarify it somewhat. To use the LIMIT command is to block off the top of memory for the use of a subroutine in Machine code, which you may wish to write into a program, but more on that later.
Your Basic programs use the uppermost part of memory for storing variables that are being used during the running a program. They gradually grow downwards and would eventually hit the end of the program listing, which then results in an MEMORY error being displayed. Therefore any Machine code routine must be protected from being overwritten by the Basic program running. So all that happens is, that you alter the top of memory to a lower position and leave room right at the top for your Machine code routine.
These routines are not the easiest of programs to write. You will have to gain some knowledge of Assembler language. You cannot POKE any number you wish, it has all got to be coded. But the facility is there if and when you need it, and very useful it is, we will see the LIMIT command used later.
Many of the ready to run games programs contain this LIMIT command with a machine code routine, but problems can arise when you wish to load the next program. If you see a Memory error message displayed the probable cause is that the top of memory has been protected by the previous program. The best method of checking this, is to type PRINT SIZE, and if

a lower figure than usual is shown then this is probably the cause. You must now clear memory. Type BYE, and the display will return to the Monitor, and MONITOR**SP1002 should be on the screen. Now enter GOTO$1200(remember the dollar sign it is important) and the correct number of Bytes should be displayed. You can now LOAD the next program as usual and this time it should load with no difficulty.

MEMORY MAP 48 K (not to scale)

HEXADECIMAL			DECIMAL
	FFFF	DISC OPERATING SYSTEM	65535
	F000		61440
	EFFF	INPUT OUTPUT CONTROLS	61439
	E000		57344
	DFFF	VIDEO RAM AREA	57343
	D000		53248
	CFFF	PROGRAM IN BASIC	53247
	4806		18438
	4805	BASIC WORK AREA	18437
	4200		16896
	41FF	BASIC SP 5025	16895
	1200		4608
	11FF	MONITOR WORK AREA	4607
	1000		4096
	0FFF	MONITOR	4095
	0000		0000

2

Features of BASIC SP 5025

2.1 Basic Flags

The MZ-80K is supplied with one copy of SHARP BASIC SP 5025, but tapes can break or corrupt if placed too near a magnet, such as a television or Hi-Fi speaker, so it's a good idea to make a security back-up copy.

In the Basic SP 5025 tape is a protection flag which *** is set so that one cannot PEEK (look at) the values of bytes in memory. It is a good idea when making a back-up copy of Basic, to POKE this flag to off. It is a lot easier than it sounds.

Load Basic as usual, now place a new cassette in the drive, the screen will be displaying 'READY'.
Enter POKE 1Ø167,1 and press 'CR'.
The display will show 'READY' again, but you have just turned the protection flag off, and now the value of any byte will be accessable to you.
Now enter this one line program:-
1Ø USR(33):USR(36)
Now enter 'RUN' followed by 'CR'. The message on the screen will ask you to press RECORD and PLAY. Do this and in about three minutes you will have a security copy of Basic.
It is probably good practise to use this tape when loading Basic, and keep the original in a safe place, away from any magnetic fields.

When you SAVE a program on tape there is additional information recorded at the beginning of the tape before your actual program. You are not made aware that it is happening, but all programs have such data recorded into the header of the program, such as the starting address and the length of that particular program. There is also a facility to make the program RUN automatically as soon as it has finished loading the next time you use it.

*** A flag is just like a Traffic signal, stop or go.
It is only a byte really that has a value just like any other. By POKEing it with another number changes it's value just like a signal from red to green.

This is achieved by entering POKE 1Ø682,1 followed by 'CR' before you SAVE the program to tape. After this you should SAVE the program as described on page 92 of your Sharp Manual, in the normal way. It will then RUN whenever you load that program.

Some of you may have purchased programs on tape and wanted to look at the listing, or simply changed some of the effects on the screen, but have been unable to do so, as when you enter LIST all that happens is a return to the READY message. This is because another flag has been switched on and recorded onto the header of the program. To switch this flag off enter POKE 1Ø68Ø,Ø and 'CR'. Now the program should LIST in the normal way. The reason this protection was entered on the tape was to stop people buying a program and running off copies to sell to others, thus deprieving the authors and software firms from royalties and profits. This flag also stops one SAVEing the program too.
SO DO NOT RUN OFF COPIES OF PROGRAMS SOLD TO YOU, AS IT IS A BREACH OF COPYRIGHT.

2.2 Personalising your Basic

A routine which may be of interest is that all the sreen messages such as:- READY, BREAK, SYNTAX ERROR etc., can be changed to anything you wish, providing that you do not exceed the number of characters in the message you wish to alter.
Let us take a look at READY.
If you have not already done so turn off the PEEK protect flag by entering POKE 1Ø167,1 followed by 'CR'.
Now enter this program:-

```
1Ø    FOR X = 4857 TO 4861
2Ø    PRINT X,PEEK(X),CHR$(PEEK(X))
3Ø    NEXT X
RUN
```

On the screen should be printed:-

```
4857      82        R
4858      69        E
4859      65        A
486Ø      68        D
4861      89        Y
READY
```

This is where the message READY resides in memory, in locations 4857 to 4861 inclusive.
Let us assume that your initials are CP. You could alter the message from READY to OK CP.

Look up the table of ASCII codes on page 121 of the Sharp manual. Do not confuse these with the DISPLAY codes on page 117, they are different. The code for O is 79, K is 75, SPACE is 32, C is 67 and P is 8Ø. If you POKE locations 4857 to 4861 with the above values the OK CP will be displayed each time instead of READY.

Enter NEW and 'CR' then enter this program:-

```
1Ø    FOR X = 4857 TO 4861
2Ø    INPUT A
3Ø    POKE X,A
4Ø    NEXT X
RUN
```

The computer will require the first number to be entered.
On the screen should be:-

```
?
```

So enter number 79 followed by Carriage Return,'CR'.
Do this for the five values you wish to enter, the screen should be displaying:-

```
? 79
? 75
? 32
? 67
? 8Ø
OK CP
```

And now each time the computer returns to READY, OK CP will be printed instead. In fact you have actually changed five Bytes of the Basic memory. Obviously one can change the five Bytes to anything one wishes, but you will have to look up the ASCII codes you require. Try it, you will not corrupt the memory if you only POKE to these particular addresses.
If you wish to keep this feature, you will have to record it onto your back-up copy of the Basic, as described earlier by entering the one line program, after clearing the last program by entering NEW.

```
1Ø    USR(33):USR(36)
```

NEVER RECORD OVER THE ORIGINAL BASIC TAPE SUPPLIED WITH THE SHARP.

For obvious reasons it should be kept in it's original state, use your back up copy for any changes you make.

All the screen messages are held in memory locations from 4834 to 4950, and all you have to do is alter line 1Ø in the program which PEEKed the READY message to see all of them. After each word you will find the next Byte has a value of 13, although this code is not listed in the ASCII codes, it actually tells the computer to perform a carriage return after printing the message, so never change the value of this Byte. Remember by using PEEK you cannot change any part of memory, so no harm will be done. Even if you POKE the wrong location, or POKE the incorrect value, no permanent damage will occur. The worst that can happen is that the program hangs or crashes, which means it simply will not run correctly. If this happens reload Basic, and all of memory will return to it's original condition, do not be afraid to experiment.

After listing the screen messages you will notice that most of them refer to errors such as:- SYNTAX, MEMORY, MISMATCH and the like. You can alter them if you wish, but remember that their original meaning should be reflected in whatever you change them to, otherwise you may not understand them at some future time.

2.3 Cursor Positioning

The 'PRINT AT' statement, which positions the cursor anywhere on the screen can be found in Sharp Disc Basic, but is unfortunately absent from Tape SP 5025 Basic. At the moment to print a prompt at the bottom of the display one would use a line such as:-

```
1ØØ   PRINT"↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓→→→→→→PRESS ANY KEY"
```

Which equals 23 arrows down and 6 arrows right.

If you use the alternative "PRINT@" statement, which can be entered as "?@" , it becomes much simpler and saves on memory.

```
1ØØ   PRINT@22,5;"PRESS ANY KEY"
```

The reason 22 was entered as the line position is that the top line is considered as Ø and the bottom line on the screen is 24. Similarly the columns are from Ø to 39, so column 6 would be 5. This is similar to the TAB function explained on page 49 of the Sharp Manual.

To add this PRINT@ statement to your Basic the following program should be entered, afterwhich you should rewind your back up copy Basic and RUN, and the new command will be recorded for future use. You haven't made a back up tape yet? Well you can now, this routine will prove very useful.

When you have typed in this program, go over it checking that it is correct, especially the numbers in the DATA lines along with all the commas. Also ensure that you have not placed any commas after the <u>last</u> item in each DATA line.

```
1     POKE 1Ø167,1
2     DATA2Ø5,139,22,64,69,28,2Ø5,169,25
3     DATA123,5Ø,114,17,2Ø5,154,22,44,2Ø5
4     DATA169,25,123,5Ø,113,17,195,69,28
5     FOR X = 154Ø5 TO 15431:READ A
6     POKEX,A:NEXTX
7     POKE7221,45:POKE7222,6Ø
8     USR(33):USR(36)
RUN
```

You will be prompted to press RECORD.PLAY.

After about 3 minutes the tape should stop and you will have added this extra command to Basic. To check how it works clear the screen and enter:-

?@2Ø,18;"OK" followed by the CR key

OK should be printed at the bottom of the screen in the middle.

Please remember that the first number after the ?@ refers to the line number, and must be in the range Ø to 24. Similarly the second number after the comma, is the column and it's range is Ø to 39.

These two numbers do not have to be actual numbers, they can be variables such as X and Y, which would have a value allocated to them.

Type in NEW and enter this program:-

```
1Ø    ?"C"
2Ø    FOR Z = 1 TO 1ØØ
3Ø    Y = RND(1)*12
4Ø    X = RND(1)*39
5Ø    ?@Y,X;"*"
6Ø    NEXT Z
7Ø    ?@13,Ø
RUN
```

You will see stars printed at random over the top half of the screen. The printing is done in line 5Ø at positions designated by variables Y and X. These two variables are given values each time the program runs through lines 3Ø and 4Ø , a total of 1ØØ times because of the loop in lines 2Ø and 6Ø. All line 7Ø does is to move the cursor down past the displayed area, so the READY message does not get printed in the top half of the screen. Had the last value of Y been randomly selected as 2 or 3 on the last time through the loop, the READY message would have been printed one line lower in the displayed area.
Another point about the ?@ statement is that it truncates the value given to the variables used, in this case X and Y. If the random number given to Y was 11.021497, the ?@ would consider the value as a whole number equal to 11. Therefore the INT statement that you often see used in RND program lines such as, X = INT(RND(1)*12), is not necessary.

Enter NEW and type in this program which generates patterns:-

```
1Ø   PRINT"⌂"
2Ø   FOR A = 1 TO 5
3Ø   READ D
4Ø   FOR B = 1 TO 1ØØ
5Ø   A$ = CHR$(D)
6Ø   FOR C = 1 TO 2
7Ø   Y = RND(1)*12+1
8Ø   X = RND(1)*2Ø+1
9Ø   PRINT@Y,X;A$
1ØØ  PRINT@24-Y,X;A$
11Ø  PRINT@Y,4Ø-X;A$
12Ø  PRINT@24-Y,4Ø-X;A$
13Ø  A$ = " "                (note space between quotes)
14Ø  NEXT C,B,A
15Ø  RESTORE:GOTO2Ø
16Ø  DATA167,172,149,112,124
RUN
```

You will need to enter SHIFT and BREAK to stop the program. You may prefer to omit the space in quotes in line 13Ø, so that A$= "". Try experimenting by changing the numbers in the DATA statements in line 16Ø . They are listed in the ASCII codes on page 121 of the Sharp manual.

2.4 Tone Generator

Another variation on Basic which may be of interest, is when the READY message is displayed, the Sharp plays some notes. In the direct mode type in:-

POKE4684,Ø and CR

You should have heard eleven notes played in succession. The 2 bytes of memory controlling this feature are 4684 and 4685. You can experiment by typing in different numbers after the comma. Type in:-

POKE4685,26 and CR

That was different again.

Mind you I think these notes might drive the other members of your household slowly mad, being played each time READY is displayed. If you like the idea of an audible as well as a visual prompt for READY try this:-

POKE4684,Ø:POKE4685,Ø and CR

That should give you a single note, which sounds each time the computer returns to READY. This tone also sounds when error messages are screened, which could prove useful.

You can keep this feature on your Basic back up tape by entering the one line program:-

USR(33):USR(36)

Do not forget to rewind the tape first.

To turn off the tones and return the 2 bytes to their original values enter:-

POKE4684,254:POKE4685,18

While on the subject of sound, the tone generator on the MZ-80K is very powerful. Besides the MUSIC statement and the USR(62) command, a large variety of noises can be executed.

Enter this short program, it should sound similar to an old American Police siren:-

```
1Ø   B = 3
2Ø   FOR C = 1 TO 5
3Ø   FOR A = 1ØØ TO 1 STEP -1
4Ø   POKE4513,A
5Ø   POKE4514,B
6Ø   USR(68)
```

```
7Ø    NEXT A
8Ø    FOR A = 1 TO 1ØØ
9Ø    POKE4513,A
1ØØ   POKE4514,B
11Ø   USR(68)
12Ø   NEXT A,C
```

RUN

Try changing line 1Ø to B = 8 and hear the difference.

You will see that line 5Ø uses the value of B to POKE into location 4514, as this byte is responsible for the main pitch of the note.

A more modern Police siren sounds like this:-

```
1Ø    B = 3
2Ø    FOR C = 1 TO 1Ø
3Ø    FOR A = 25Ø TO 1 STEP-7
4Ø    POKE4513,A
5Ø    POKE4514,B
6Ø    USR(68)
7Ø    NEXT A,C
```

RUN

Try changing the STEP value in line 3Ø for a longer or shorter note.

I have deliberately used one statement per line for clarity. When writing your own programs, it will save time and memory if you enter several statements on each line separated with colons :

The next example could be added to the STARTREK program, it simulates the Short-wave transmissions often associated with space noise.

```
1Ø    FOR A = 1 TO 2ØØ
2Ø    B = INT(RND(1)*2Ø)
3Ø    POKE4514,B
4Ø    USR(68)
5Ø    FOR C = 1 TO 25:NEXT C,A
```

RUN

There are numerous combinations that can be used to create different tones. Experiment using the 2 addresses 4513 and 4514, but don't exceed 30 on 4514, as it becomes a very low note indeed.

2.5 Video Ram Area

As you already know, the POKE command allows one to place a value in a memory location. These values can be any number between Ø and 255.
Part of the Sharp memory is allocated the Video Ram area, and the first 1ØØØ bytes of this area are used for mapping out the screen, that is 25 lines of 4Ø characters, all 1ØØØ of them is stored in locations 53248 (top left of the screen) to 54247 (bottom right). If you clear the screen, so that the cursor is on the top line, and enter POKE54247,26 followed by CR you will see the letter Z has been printed in the bottom right hand position of the screen. However the cursor has returned to near the top of the screen, which demonstrates that by POKEing to the screen area the position of the cursor remains unchanged. The characters, along with their codes are listed under the Display Code Table on page 117 of the Sharp manual, you will see that the letter corresponding to code 26, which we just used, is the letter Z. The Display Code is only used when you are directly POKEing to the screen within the range 53248 to 54247, do not confuse it with the ASCII codes on page 121. To begin with, when writing your own programs, it might be a sensible idea to use graph paper to map out the screen and it's positions.
The following program uses direct addressing to the screen, it is fairly simple and should give you an example of how the POKEing is carried out.
It is a typical Alien type program, but REM statements have been included, so that you will know which part of the program does what. These can of course be omitted in your listing without affecting the running of it.

```
1Ø    GOSUB11ØØ
2Ø    DG = Ø
3Ø    DIM M(8,3)
4Ø    REM
5Ø    REM---STARTING POSITIONS OF ALIENS---
6Ø    REM
7Ø    DATA533Ø7,Ø,4Ø,53393,Ø,39,53642,Ø,-1
8Ø    DATA53873,Ø,-41,539Ø7,Ø,-4Ø,53861,Ø,-39
9Ø    DATA53613,Ø,1,53381,Ø,41
1ØØ   FOR I = 1 TO 8
11Ø   FOR J = 1 TO 3
12Ø   READM(I,J)
```

```
13Ø  NEXT J,I
14Ø  INPUT"▯▯▯SPEED  (1TO9)";S
15Ø  IF(S<1)+(S>9)THEN GOTO 14Ø
16Ø  SP = INT(2ØØ/S)
17Ø  PRINT"▯"
18Ø  REM
19Ø  REM---PRINT SQUARE ON SCREEN---
2ØØ  REM
21Ø  FOR X = 6 TO 72
22Ø  SETX,Ø
23Ø  SETX,34
24Ø  NEXT X
25Ø  FOR Y = Ø TO 34
26Ø  SET6,Y
27Ø  SET72,Y
28Ø  NEXT Y
29Ø  H = Ø
3ØØ  IF RND(1)>.9THEN GOTO 32Ø
31Ø  GOTO 3ØØ
32Ø  F = Ø
33Ø  X = RND(1)
34Ø  REM
35Ø  REM---POSITION OF YOUR SHIP---
36Ø  REM
37Ø  POKE53627,199
38Ø  REM
39Ø  REM---ANGLE OF ALIENS APPROACH---
4ØØ  REM
41Ø  D = INT(X*8+1)
42Ø  POKE M(D,1),199
43Ø  M(D,2)=M(D,1)
44Ø  REM
45Ø  REM---WHICH KEY IS PRESSED---
46Ø  REM
47Ø  GET A$
48Ø  IF A$=""THEN 67Ø
49Ø  IF A$="W" THEN F=1
5ØØ  IF A$="E" THEN F=2
51Ø  IF A$="D" THEN F=3
52Ø  IF A$="C" THEN F=4
53Ø  IF A$="X" THEN F=5
54Ø  IF A$="Z" THEN F=6
55Ø  IF A$="A" THEN F=7
56Ø  IF A$="Q" THEN F=8
57Ø  REM
58Ø  REM---FIRING SOUND---
59Ø  REM
6ØØ  B = 4
61Ø  FOR C = 24Ø TO 1 STEP -4Ø
62Ø  POKE4513,C:POKE4514,B
63Ø  USR(68)
64Ø  NEXT C
65Ø  USR(71)
66Ø  IF F = D THEN 76Ø
67Ø  IF M(D,2) =53627  THEN 88Ø
68Ø  POKEM(D,2),Ø
69Ø  POKEM(D,2)+M(D,3),199
```

```
7ØØ  M(D,2) = M(D,2)+M(D,3)
71Ø  FOR S =1 TO SP:NEXT S
72Ø  GOTO 47Ø
73Ø  REM
74Ø  REM---ALIEN HIT---
75Ø  REM
76Ø  POKEM(D,2),1Ø7
77Ø  B = 6
78Ø  FOR C = 1 TO 24Ø STEP 2
79Ø  POKE4513,C:POKE4514,B
8ØØ  USR(68)
81Ø  NEXT C
82Ø  USR(71)
83Ø  H = H + 1
84Ø  PRINT"H↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓HITS ON ALIENS";H
85Ø  POKEM(D,2),Ø
86Ø  IF H = 1Ø  THEN PRINT"C":GOTO 97Ø
87Ø  GOTO 3ØØ
88Ø  MUSIC"C5"
89Ø  PRINT"H↓↓↓↓↓↓↓↓↓→→→→→→→→→→DAMAGE SUSTAINED"
9ØØ  FOR Y = 1 TO 1ØØØ:NEXTY
91Ø  PRINT"↑→→→→→→→→→→→                  "          (n.b.16 spaces)
92Ø  DG= DG+1:IF DG=1Ø THEN 95Ø
93Ø  PRINT"H↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓→→HITS ON SHIP";DG
94Ø  GOTO 3ØØ
95Ø  PRINT"C↓↓↓→→→→→→→OH DEAR"
96Ø  PRINT"↓LUCKY IT WASN'T FOR REAL"
97Ø  PRINT"↓↓↓→→→→ANOTHER GO ? (Y/N)"
98Ø  GET Z$:IF Z$=""THEN 98Ø
99Ø  IF Z$="Y"THEN RUN 2Ø
1ØØØ IF Z$="N"THEN END
1Ø1Ø GOTO 98Ø
11ØØ PRINT"CTHE OBJECT OF THE GAME IS TO FIRE AT THE"
111Ø PRINT"↓ATTACKING ALIENS BY USING THE KEYS"
112Ø PRINT"↓Q--W--E"
113Ø PRINT"↓↓A-----D"
114Ø PRINT"↓↓Z--X--C"
115Ø PRINT"↓DEPENDING ON THE ANGLE OF APPROACH"
116Ø FOR DL= 1 TO 3ØØØ:NEXT DL
117Ø RETURN
```

This program, while being very simple in it's output, lends itself to modification, and in doing so you will learn different methods of programming. As the listing is split into sections by the REM statements, it should be easy to change the output to your own requirements.

The next listing is HANGMAN which also uses DIRECT SCREEN ADDRESSING through the POKE statement. The gallows graphics are contained in DATA statements in lines 28Ø to 37Ø.

HANGMAN

```
10   DIM A$(26),H$(26)
20   PRINT"CENTER YOUR WORD"
30   PRINT"↓→→(dont let your opponent see)"
40   INPUT Y$
50   B=LEN(Y$)
60   PRINT"C*****THE WORD HAS";B;"→LETTERS*****"
70   PRINT:PRINT"UNUSED:-→→ABCDEFGHIJKLMNOPQRSTUVWXYZ"
80   PRINT:PRINT"→→USED:-"
90   FOR X = 1 TO B:A$(X)=MID$(Y$,X,1):NEXT X
100  PRINT"H→→→→→→↓↓↓↓↓↓↓↓";:FOR X=1 TO B:PRINT"*";:NEXT X
110  G = 1:T = 0
120  PRINT"H↓↓↓↓↓↓↓↓↓↓→→PLEASE PRESS LETTER No";G;"    ←←";
130  GET N$:IF N$=""THEN130
140  IF(ASC(N$)<65)+(ASC(N$)>90)THEN130
150  PRINT N$
160  POKE 53273+ASC(N$),0
170  POKE 53353+ASC(N$),ASC(N$)-64
180  GOSUB 230
190  IF V=0 THEN GOSUB 280
200  IF L=99 THEN 430
210  IF T=B THEN 420
220  G=G+1:GOTO120
230  V=0
240  FOR X=1 TO B
250  IF H$(X)=N$ THEN V=1:GOTO270
260  IF N$=A$(X) THEN PRINT"H↓↓↓↓↓↓↓↓";TAB(X+5);N$:V=V+1:T=
                                            T+1:H$(X)=N$
270  NEXT X:RETURN
280  DATA40,28,41,120,42,120
290  DATA0,121,-40,121,-80,121
300  DATA-120,121,-160,121,-200,92
310  DATA-199,120,-198,120,-197,120
320  DATA-196,95,-156,121,-116,121
330  DATA-76,207,43,120,44,120
340  DATA-37,227,-36,173,-35,227
350  DATA3,221,5,217,-76,206
360  DATA43,93,44,64,45,92
370  DATA0,99
380  Z=54078
```

```
39Ø  FOR X=1 TO 3:READK,L
4ØØ  IF L=99 THEN RETURN
41Ø  POKEZ+K,L:NEXT X:RETURN
42Ø  PRINT"HHHHHHHHHHHHHHHWELL DONE..YOU DID THAT IN";G;" GOES":
                                                        GOTO 52Ø
43Ø  POKE54ØØ2,121:POKE54Ø41,Ø:POKE54Ø42,121
44Ø  POKE54Ø43,Ø:POKE54Ø81,Ø:POKE54Ø82,121:POKE54Ø83,Ø:POKE54122,121
45Ø  B=3
46Ø  FOR A=1 TO 24Ø STEP5
47Ø  POKE4513,A:POKE4514,B
48Ø  USR(68)
49Ø  NEXT A
5ØØ  USR(71)
51Ø  PRINT"HHHHHHHHHHHHHHNO IT WAS ";Y$
52Ø  PRINT"HHANOTHER GO? (Y/N)"
53Ø  GET Z$:IF Z$=""THEN53Ø
54Ø  IF Z$="N"THEN END
55Ø  IF Z$="Y"THEN RUN
56Ø  GOTO 53Ø
```

The unused letters of the alphabet are printed on third line of the screen, in fact the letter A is printed in position 53338 in the Video Ram area. This is calculated by remembering that the top left position is 53248, so the start of the second line must be 53248 plus 40, which is 53288, and likewise the third line will start with position 53328, being 40 more than line two. The letter A is printed on line three, eleven characters in from the left, so it's position will be known to be 53338. When you run the program it will be seen that on selection of a letter, the letter moves from the unused row to the used row. We know the ASCII codes for the letters by using the table on page 121 of the manual, so the first letter A has an ASCII value of 65. If we deduct 65 from the position of A(53338) we get the number 53273. So you will see that in line 16Ø of the program we POKE position 53273 and add the ASCII value of the selected letter to it, in this case it would be 65 for the letter A, making a total of 53338, with a zero, which is a blank space.

In order to print the letter two lines lower on the screen in the used section the value of 53273 will have to have 80 added to it. This is achieved in line 17Ø, but here we run into a slight problem. The position can be calculated very easily as we did previously, but the value of the item we wish to print in this position is not calculated using the ASCII codes, but the DISPLAY codes on page 117, which have different values. If you compare the two codes you will see that the letter A has a display code value of 1 and not 65.In fact each letter of the alphabet is 64 less than it's equivalent in the ASCII code. Therefore by POKEing the calculated position with the ASCII value of the chosen letter and subtracting 64 the correct letter in the display code will be printed two lines lower on the screen.

If you are having difficulty understanding the principles in using the Memory Map of the screen this simplified short program should help in showing how characters are moved on the screen. This shows a man falling down the screen and finally walking away to the right. Line 1Ø uses a loop to move the man, starting at position 20 on the top line of the screen, and incrementing in steps of 40, which will reprint him directly below his previous position on the following line.

```
1Ø    FOR Q = 2ØTO98Ø STEP 4Ø
2Ø    PRINT"Q"
3Ø    A=INT(4*RND(1)+202)          (randomly selects 4 positions
4Ø    POKE53248+Q,A                   of man 202 to 206)
5Ø    FOR R = 1TO2ØØ:NEXT R        (time loop for display in
6Ø    NEXTQ                                each position)
7Ø    FOR Z = 1TO19
8Ø    PRINT"C"
9Ø    A=2Ø2                        (display code value of man
1ØØ   POKE53248+Q-4Ø+Z,A                   standing upright)
11Ø   FOR R = 1TO2ØØ:NEXT R
12Ø   NEXTZ
```

3
Style your Programs

3.1 Peeks and Pokes

On page 120 of the Sharp Manual there are a limited number of special control commands listed. One of these being POKE 4509,0 which sounds a note each time a key is pressed. While POKEing it with a value of 1 restores to normal. There are many more locations in memory that can have their contents changed to another value for different effects. These commands can be used in either the direct mode or written into programs.

POKE59555,Ø blanks the screen.

POKE59555,1 restores to normal.

Care should be taken whenever POKEing is carried out. Always ensure that you have got the correct address otherwise your program might not run correctly if at all. With the above address after the first POKE the screen will go blank and whatever you type in will not be displayed, so you will be typing blind. One use of this command could be to flash the titles of a program on and off at the start of the run. You will see it demonstrated in the CONNECT FOUR listing.

POKE57347,4 turns the LED on the right of the keyboard to red, but does not change to lower case letters.

POKE57347,5 turns it back to green.

This can be used to indicate an error by the user, and could be written as a subroutine coupled with the USR(62) command:-

```
1Ø   GOSUB1ØØØ
2Ø   END
1ØØØ FOR Z = 1TO1Ø
11ØØ POKE57347,4:USR(62)
111Ø FOR A = 1TO1ØØ:NEXT
112Ø POKE57347,5
113Ø FOR A = 1TO1ØØ:NEXT
114Ø NEXTZ
115Ø RETURN
```

POKE6636,Ø:POKE6637,Ø Renders the BREAK key inoperative when the program is running.

POKE6636,2Ø5:POKE6637,3Ø Restores to normal.

If it is important that the program is incapable of being prematurely halted, then this command should be written into the program early in the listing, along with the following command. As with only using the previous command the BREAK key could still be used while the program is waiting for an INPUT.

POKE79Ø6,Ø:POKE79Ø7,Ø:POKE79Ø8,Ø Renders the BREAK key inoperative on INPUT.

POKE79Ø6,2Ø2:POKE79Ø7,252:POKE79Ø8,3Ø Restores to normal.

POKE4464,1 Changes to lower case(SML CAP) letters.

POKE4464,Ø Changes back to upper case (NORMAL) letters.

POKE1Ø682,1 While in direct mode before saving a program will make the program RUN automatically each time it is loaded.

POKE1Ø68Ø,1 Also in direct mode before saving will stop the program being LISTed or SAVEd each time it is loaded.

POKEing these two addresses with a zero (Ø) restores to normal.

You may have seen in professionally written programs that the first few lines, which usually contain the authors name or copyright notices in REM statements, have a zero line number, and wondered how and why it was achieved. The advantage of this is that the line cannot be deleted or changed without renumbering, and therefore is semi permanent. It is useful for ensuring that your name etc. remain on the program. You cannot directly enter a line with a zero as it's number (try it), but you can change any amount of lines to zero after you have finished writing and debugging your program.

Load in any Basic program and LIST it making a note of the last line number used. The following routine when entered must be numbered higher than the program loaded. Let us assume the program finishes with a line number 31ØØ, then enter this routine starting with the line number 4ØØØ. As you know Basic programs load into memory from location 18438(decimal), but the first two locations(bytes) refer

to the start of the following line, so that the computer knows where the next line starts. Locations 18440 and 18441 hold the number of the actual first line number. If we POKE these locations with a zero (0) then we will have changed the first line to line 0. You can change as many lines as you wish to zero, the program will still run correctly.

Enter these five lines:-

```
4000 A=18440
4001 PRINTA;PEEK(A)
4002 FOR X =18442TO18542
4003 IF PEEK(X)=13THEN PRINTX+3;PEEK(X+3):X=X+4
4004 NEXT X
```

Now enter:-

GOTO4000 and 'CR'

The display on the screen should be similar to this:-

```
18440 10
18450 20
18483 30
18494 40
18504 50
18537 60
```

Apart from the first number (18440), your numbers will not necessarily be the same as listed here. The actual line numbers at the begining of your program are listed in the second column(in the above case from 10 to 60)and their respective locations in memory in the first column.

To change line 10 to 0 you would enter in direct mode:-

POKE18440,0 followed by CR

To change the second line in the above case from 20 to 0:-

POKE18450,0 and CR

Remember the first column is the location in memory and the second is the line number. Only use this routine for low numbered lines, as if they are numbered above 255 another byte is used to cope with the higher number.

If you now list the program, you will see that the first two lines have changed to 0. You can alter as many lines as you wish, but only do it after you have finished writing and debugging the program, and remember to delete lines 4000 to 4004 before you SAVE the program.

The cursor is controlled by locations 4465 and 4466.

POKE4465,2Ø Would move the cursor horizontally along a line to position 21(the positions are from Ø to 39)

POKE4466,1Ø Would move it down to line 11(the positions are from Ø to 24)

When using the GET statement the ASCII value of the key held down is placed in location 17828 in memory, and can be used in many ways. Enter:-

```
1Ø   GETA$:PRINT CHR$(PEEK(17828));
2Ø   GOTO1Ø
```

When you RUN this you will see that unlike the usual form that the GET statement takes, of just getting one character from the keyboard, and waiting for the next key to be pressed, it actually continues printing any key held down. This can be adapted very easily into games programs where fast moving graphics are required. Consider the following short routine which demonstrates this aspect.

```
1Ø   PRINT"Q"
2Ø   Z=17828:X=53748:Y=X
3Ø   POKEY,Ø:POKEY+1,Ø:POKEY-1,Ø
4Ø   POKEX,191:POKEX+1,231:POKEX-1,232
5Ø   GET Q$
6Ø   Y=X
7Ø   IF PEEK(Z)=81 THEN X=X-41
8Ø   IF PEEK(Z)=65 THEN X=X-1
9Ø   IF PEEK(Z)=9Ø THEN X=X+39
1ØØ  IF PEEK(Z)=88 THEN X=X+4Ø
11Ø  IF PEEK(Z)=67 THEN X=X+41
12Ø  IF PEEK(Z)=68 THEN X=X+1
13Ø  IF PEEK(Z)=69 THEN X=X-39
14Ø  IF PEEK(Z)=87 THEN X=X-4Ø
15Ø  IF X<53288 THEN X=Y
16Ø  IF X>54207 THEN X=Y
17Ø  IF PEEK(Z)=Ø THEN 5Ø
18Ø  GOTO3Ø
```

To move the object around the screen use these keys:-
Q=Northwest, W=North, E=Northeast, D=West, C=Southeast, X=South Z=Southwest and A=West.

3.2 User Prompts

You have probably seen in programs, or even written into your own programs the "PRESS ANY KEY" prompt, followed by the GET statement. This can cause confusion to non-computerist users of the program in whether they have pressed the wrong key. A far more explicit instruction would be to "PRESS SPACE TO CONTINUE" or even "PRESS CR".

All you need to do is test the GET input for the ASCII code of the key pressed. If they have pressed a wrong key the program waits for the correct key, and the display on the screen remains unchanged until the correct is pressed.

All that is required is to find the ASCII code of the specific key that you want them to press.

The ASCII code for SPACE is 32

The ASCII code for CR is 102 (although this is not in the manual)

So a typical routine would look like this:-

```
1ØØ  GOSUB 1ØØØ
11Ø  PRINT"YOU PRESSED CR":GOTO1ØØ  (program would continue
                                     from here)
1ØØØ PRINT"PRESS CR TO CONTINUE"
1Ø1Ø GET A$:IF A$=""THEN1Ø1Ø
1Ø2Ø IF ASC(A$)=1Ø2 THEN 1Ø4Ø
1Ø3Ø GOTO 1Ø1Ø
1Ø4Ø RETURN
```

Apart from the CR key having an ASCII code which is not mentioned in the manual, all the other yellow keys also have a code including the SML/CAP key.

Enter the next listing, and you will see the codes for each key pressed. When asked for PRESS KEY, press the cursor keys with or without the SHIFT, and the codes will be displayed:-

```
1Ø   PRINT"PRESS KEY"
2Ø   GET A$: IF A$=""THEN2Ø
3Ø   PRINT ASC(A$)
4Ø   GOTO 1Ø
```

CONNECT FOUR

```
10   GOSUB1130
20   DIM NA$(2),X(9),B(3),FW(4),S(2)
30   FOR N = 1TO2
40   X=53858
50   PRINT"[C]WHAT IS YOUR NAME PLAYER";N:USR(62)
60   INPUT NA$(N)
70   N$="HELLO@"+NA$(N)                          (REMEMBER THE @
80   FOR L = 1TOLEN(N$)                           AFTER HELLO)
90   A=ASC(MID$(N$,L,1))
100  P=X+L-20*(A-64)
110  M=X+L
120  POKEM+80,A+64
130  FOR J = M TO P STEP-40
140  IF J=P THEN POKEJ,248:GOTO160
150  POKEJ,127
160  NEXT J,L
170  USR(62):FOR DL = 1TO3000:NEXTDL
180  NEXT N
190  N=1
200  PRINT"[C]"
210  FOR A = 1TO9
220  READ X
230  FOR B = 1TO8
240  POKEX,121
250  X=X+40
260  POKEX,189
270  X=X+40
280  NEXT B
290  NEXT A
300  DATA53426,53428,53430,53432,53434,53436,53438,53440,53442
310  RESTORE
320  FOR A = 1TO8
330  READ X
340  FOR B = 1TO8
350  POKEX+41,120
360  X=X+80
370  NEXT B,A
380  RESTORE
390  AA=33
400  FOR A = 1TO8
410  READ X
420  POKEX-119,AA
430  AA=AA+1
440  NEXT A
450  PRINT@20,0;"-------------------1 2 3 4 5 6 7 8-----"
460  PRINT"[H]SCORE"
470  PRINT@1,0;S(1)
480  PRINT@3,0;S(2)
490  PRINT@1,3;NA$(1);" IS ";CHR$(241)
500  PRINT@3,3;NA$(2);" IS ";CHR$(247)
510  PRINT@22,0;"                                    " (33 spaces)
520  P=N+70
530  PRINT@11,0;"WHICH COLUMN(1to8)"
540  PRINT:PRINT"                    "
550  PRINT"[↑][→][→]";NA$(N):USR(62)
```

```
560  GET A$:IF A$=""THEN560
570  IF(ASC(A$)<49)+(ASC(A$)>56)THENGOTO560
580  PRINT"↑↑↑       COLUMN ";A$;"     "
590  RESTORE
600  FOR A = 1TO9
610  READX(A):NEXT A
620  AA=VAL(A$)
630  X=X(AA)-79
640  IF PEEK(X+80)<> 0THENPRINT@22,0;"***** THAT COLUMN IS
FULL ******":GOTO660
650  GOTO670
660  USR(62):FOR DL = 1TO500:NEXTDL:GOTO510
670  POKEX,P
680  FOR DL = 1TO100:NEXT DL
690  IF PEEK(X+80)=0 THEN POKEX,0:X=X+80:GOTO670
700  GOSUB730
710  IF N=1THEN N=2:GOTO510
720  N=1:GOTO510
730  CW=X:TL=1
740  FW(TL)=CW
750  DATA80,160,240,-80,-160,-240
760  DATA78,156,234,-78,-156,-234
770  DATA82,164,246,-82,-164,-246
780  DATA2,4,6,-2,-4,-6
790  FOR V = 1TO8
800  IF INT(V/2)<>V/2 THENTL=1
810  FOR A =1TO3
820  READB(A)
830  NEXT A
840  FOR A = 1TO3
850  IF PEEK(CW+B(A))<>P THEN900
860  TL=TL+1
870  FW(TL)=CW+B(A)
880  IF TL = 4THEN920
890  NEXT A
900  NEXT V
910  RETURN
920  PRINT@15,0;"THE WINNER IS"
930  PRINT@17,4;NA$(N)
940  S(N)=S(N)+1
950  F=0
960  Q=74
970  IF F>=10THEN1050
980  FOR A = 1TO4
990  POKEFW(A),Q:NEXT A
1000 USR(62)
1010 FOR DL = 1TO400:NEXT DL
1020 IF Q=74 THEN Q=P:GOTO1040
1030 Q=74
1040 F=F+1:GOTO970
1050 PRINT@23,0;"ANOTHER GAME? (Y/N)"
1060 GETSS$:IF SS$ =""THEN1060
1070 IF SS$="Y"THEN1100
1080 IF SS$="N"THEN1120
1090 GOTO1060
1100 IF N=1 THEN N=2:RUN200
```

```
111Ø N=1:RUN2ØØ
112Ø PRINT"THANKS ";NA$(1);" & ";NA$(2):MUSIC"C3D3F3":END
113Ø PRINT"C"
114Ø FOR A = 1TO5
115Ø POKE59555,Ø
116Ø PRINT@4,1Ø;"****************"
117Ø PRINT@6,1Ø;"  CONNECT FOUR"
118Ø PRINT@8,1Ø;"****************"
119Ø FOR DL = 1TO3ØØ:NEXT DL
12ØØ POKE59555,1
121Ø FOR DL = 1TO3ØØ:NEXT DL,A
122Ø PRINT"↓↓↓FOR 2 PLAYERS"
123Ø PRINT"↓THE FIRST TO GET 4 TOKENS IN ANY ROW"
124Ø PRINT"↓IS THE WINNER"
125Ø PRINT"↓EITHER HORIZONTAL, VERTICAL OR DIAGONAL"
126Ø FOR DL = 1TO2ØØØ:NEXT DL
127Ø PRINT"↓press 'CR' to play"
128Ø GET Q$:IFQ$=""THEN128Ø
129Ø IF ASC(Q$)=1Ø2THEN RETURN
13ØØ GOTO128Ø
```

This program whilst not being too professional, does use some good routines which you may wish to use in programs of your own.

The title is contained within lines 113Ø and 13ØØ, and is called as a subroutine in line 1Ø, it uses the POKE59555 mentioned earlier to flash the screen display on and off.

Placing the titles at the end of the program means that you are not confined for space, and can take time compiling them when the main program is completed.

The name printout is contained within lines 3Ø to 18Ø and can easily be used in other programs.

The printing of the grid is done in lines 2ØØ to 45Ø and has been kept to one statement per line for clarity.

The main part of the program is between 46Ø and 111Ø, with the checking for a winning line in lines 73Ø to 91Ø.

When entering the listing note that the @ after HELLO in line 7Ø is not an error, and in line 45Ø there is one space between each of the numbers, 1 to 8, also line 51Ø has 33 spaces between the quotes.

4

Program Tips

4.1 Decimal Aligning

The most annoying aspect of printing out a table of numbers, is getting them to align under each other. There is a routine for achieving this on page 72 of the Sharp manual, but it does not allow for decimals, such as when one is trying to print Pounds and Pence in a cheque book program or budget account.

The following routine may look complicated, but it works, and should be entered as a subroutine, and called whenever printing is required.

The variable N is the figure to be printed.

```
1Ø    DEF FND(X)=INT(LOG(ABS(N))/LOG(1Ø))
2Ø    PRINT"C"
25    INPUT"INPUT DECIMAL NUMBER→";N
4Ø    GOSUB1ØØØ
5Ø    GOTO25
1ØØØ  Z = 3Ø
1Ø1Ø  IF N<1 THEN Z=Z-1:IF N<Ø.1 THEN Z=Z-1
1Ø2Ø  PRINT TAB(Z-FND(X));N
1Ø3Ø  RETURN
RUN
```

The variable Z in line 1ØØØ is the position of the first character to be printed. Change it to your own requirements. Line 5Ø has been inserted to demonstrate, and would not be needed in a program.

If you input various numbers when prompted such as:- .Ø9, 1.87, 8.987, 12, 4.05, etc., you will see that they are all aligned.

Whilst this format is adequate for most applications, there are additional lines which, when added, give a formatted output which makes monetary output etc., look more correct.

For example if the input was the number 12, 12.ØØ would be printed, similarly 1.5, would become 1.5Ø.

Try adding the following lines to the above program.

```
5     Q$="Ø"
3Ø    N=INT(N*1ØØ+Ø.5)/1ØØ
31    N$=STR$(N)
32    A=LEN(N$)
33    FOR B=1 TO A
34    IF MID$(N$,B,1)="." THEN GOTO 37
35    NEXT B :N$ =N$+"."+Q$+Q$
36    GOTO 4Ø
```

```
37    D=VAL(MID$(N$,B,A))
38    IF D*1Ø<>INT(D*1Ø) THEN 4Ø
39    N$=N$+Q$
```

ALSO CHANGE THE VARIABLE N IN LINE 1Ø2Ø TO N$

The program also rounds up any figures after the second decimal on input, if you do not want this delete the +Ø.5 in line 3Ø.

4.2 Logical Operators AND/OR

In many variants of Basic, and program listings, you may see the logical operators AND OR used such as:-

IF Z>47 AND Z<58 THEN GOTO (line number)

Sharp Basic does support these operators, but it isn't too easy to find in the Manual. Their descriptions are on page 113. The line above would have to be changed, but could still be entered as one line.

The Sharp equivalent to AND is the * symbol.

And the equivalent to OR is the + symbol.

The line above written for the MZ-8ØK should be entered thus:-

IF (Z>47)*(Z<58) THEN GOTO (line number)

The two arguments must be enclosed in brackets.

*(AND) with +(OR) can be used in the same line such as:-

IF Z>47 AND Z<58 OR Z>64 AND Z<71 THEN GOTO (line number)

Should be altered to:-

IF (Z>47)*(Z<58)+(Z>64)*(Z<71) THEN GOTO (line number)

This feature cuts down on the otherwise larger amount of lines that would have to be used, and also reduces on the amount of memory.

4.3 Protecting Programs

We have already seen the orthordox method of protecting programs, in order to stop them being LISTed and SAVEd.
There are other ways of achieving the same result. The commands or keywords that the computer uses can be altered.
The command LIST is held in memory locations from 5343 to 5346. On all the keywords, you can alter any letter except the last in each command, i.e. on the word LIST, LIS can be changed but the T must remain as it is. In fact if you PEEKed the location 5346 expecting to see the ASCII code for the letter T, you would find a completely different character, as the computer uses the final letter in each command to differentiate between them, so never alter the last letter.

If, when you have completed debugging your program and made sure it runs effectively, you wish to alter the first letter of the command LIST, you could enter as your first line:-

```
1     POKE 5343,77
```

This will in fact only LIST the program, when it is subsequently run, if you enter the word MIST, and the word LIST will no longer be recognised and will produce a SYNTAX ERROR message if it is entered. In fact you have changed the first letter of the word by POKEing the ASCII code of M in it's place.
To ensure that the program runs as soon as it has been loaded, before SAVEing the program on tape enter in direct mode:-

```
POKE 1Ø682,1:POKE 1Ø68Ø,1
```

This will effect the AUTO-RUN flag, along with the anti LIST-SAVE flag. Please remember to note any changes you make in order to be able to return to normal. Always PEEK the location before making any alteration and remember the value it contains.

The command SAVE is located in memory from 5402 to 5405, and could be altered as with the LIST example above, making sure not to alter the last letter, and could be entered in the first line of the program.

Two other locations can be changed within the interpreter, one is in the LIST operator handler routine, and the second in the SAVE routine. These should also be entered early in the program, but not until it has been debugged.

```
POKE 6829,195:POKE 10764,195
```

POKEing these two addresses with the two values will cause the computer to return to the READY message whenever LIST or SAVE are entered.
POKE 6829,2Ø2:POKE 1Ø764,194 will return them to normal.
Quite obviously, anyone who has read this will also know of these ways of protecting programs, and by using PEEK to look at the various addresses, will be able to see which locations have been altered. But if you altered the statement PEEK to something quite different such as:- SEEK, LOOK or even SSSS, each time they entered PEEK a SYNTAX ERROR would occur, as the computer would no longer recognise PEEK as being a keyword. Unlike LIST and SAVE, all the letters of PEEK can be altered, not just the first three letters. The amount of different combinations which could be substituted is vast, similar to a small combination lock. Look up the ASCII values of the letters you wish to replace PEEK with, in the codes on page 121 of the Sharp manual.

The keyword PEEK is located in memory from 5582 to 5585. Let us assume you wished to alter it from PEEK to FIND. The ASCII codes for F.I.N.D are 7Ø, 73, 78 and 68. The second line in your program could read:-

```
2     POKE 5582,7Ø:POKE5583,73:POKE5584,78:POKE5585,68
```

After the program has AUTO executed only you will know the new word which has replaced PEEK. To look at a location in memory, and see what value it contains, you would not enter PEEK(5585) but FIND(5585) and the computer would understand the new statement and respond.

If one couples these alterations with the BREAK disable operations mentioned in section 3.1, and enters them after the program has tested within the first few lines of the listing, a fair amount of protection will exist. So remember to leave the first lines free to accomodate them.

Always keep a record of the changes you make otherwise complete chaos could occur, and remember that all memory will return to normal when you reload Basic so no harm will be done if you do come across difficulties.

5

Converting Programs

The Commodore Pet appears to be a popular microcomputer (why I do not know), and on glancing through the many micro magazines you will always find program listings written for the Pet. In actual fact it is fairly straightforfard to convert them to run on the MZ-80K.

One of the first things you will notice is the peculiar symbols used in strings, or after PRINT statements. These are usually cursor commands, and are printed inversely (white on black).

An inversed HEART sign	=	CLEAR SCREEN	C
" " " letter S	=	HOME	H
" " " letter Q	=	CURSOR DOWN	↓
" " " CLUB sign	=	CURSOR UP	↑
" " " vertical line	=	CURSOR LEFT	←
" " "] (looks like ▌)	=	CURSOR RIGHT	→

The Pet can also print black letters on a white background. To achieve this they enter an inversed letter R, and to revert to normal an inverse horizontal line. If you come across this just ignore it.

Furthermore they do not have a SML/CAP key, and lower case letters have to be POKEd in by using POKE59468,14 and returned to normal capitals by POKE59468,12. If you come across this just use the SML/CAP key instead.

The Memory map of the MZ-80K is from 53248 (top left) to 54247 (bottom right) as was explained earlier. The Pet's Memory map is 20480 bytes lower. So any POKEing written in the range 32768 to 33767 that you see will have to be converted for the SHARP by adding 20480. Sometimes they are found at the beginning of a program listing such as:- M=32768, and later on you will see a line :- POKE M,48. So just alter the original value of M to 53248, or the equivalent.

The Display codes for the graphics used with POKEing directly to the screen are different, and in the above example POKE M,48 would print a Ø at that position, whereas the Ø in the display code for the SHARP is 32. The table that follows

shows the equivalents. The table only goes up to 127 as the Pet does not have the many varied graphics that we enjoy on the SHARP. The codes from 128 to 255 are the inverse of the codes from Ø to 127, so if you saw POKE M,176 it will be to POKE a reverse Ø onto the screen at that position.

If you find that after running the converted program, the display does not look correct, you will see that altering some of the display codes of the graphics will probably improve the effects. The main concern is to get it running, these small adjustments can be quite fun as you have more graphic characters to experiment with.

PET to SHARP Conversion Table (Display Codes)

PET	SHARP	PET	SHARP	PET	SHARP
Ø	85	72	121	100	60
1to26	1to26 (AtoZ)	73	76	101	113
27	82	74	111	102	208
28	89	75	110	103	61
29	84	76	50	104	212
30	80	77	119	105	78
31	69	78	118	106	63
32	0 (space)	79	114	107	30
33to41	97to105	80	115	108	248
42	107	81	71	109	28
43	106	82	56	110	93
44	47	83	83	111	62
45	42	84	117	112	92
46	46	85	75	113	31
48to57	32to41 (0to9)	86	109	114	95
58	79	87	72	115	94
59	44	88	70	116	55
60	81	89	57	117	123
61	43	90	68	118	127
62	87	91	189	119	122
63	73	92	210	120	122
64	52	93	121	122	51
65	65	94	96	123	244
66	53	95	97	124	242
67	120	96	64	125	29
68	116	97	123	126	241
69	48	98	58	127	108
70	120	99	54		

The Pet also uses AND and OR which has already been mentioned. A typical line could read:-

```
1ØØ   IF Z=4 AND Y=2 THEN GOTO2ØØ
```

This will present no problem, simply contain the arguments in brackets and change AND to* ,and OR to+.:-

```
1ØØ   IF(Z=4)*(Y=2) THEN GOTO2ØØ
```

Another line which you may come across is the following:-

```
3ØØ   Y = -((X<1Ø)*2+(X>1Ø)*5)
```

or

```
3ØØ   Y = -2*(X<1Ø)-5*(X>1Ø)
```

To the beginner these two lines, which actually mean the same thing, look frightening. In fact they can both be entered into the MZ-80K, and it is Boolean arithmetic, something that computers are very good at, but we are not going to try to unravel it's mysteries here. Suffice to say that the above lines are a shorter version of the IF...THEN statement.
If X is less than 10 then Y = 2, and if X is more than 10 then Y = 5. You are probably saying but what if X equals 10. Well then Y will equal 0 (zero).
By using this technique you will save two extra lines.
Enter this program:-

```
1Ø    INPUT"INPUT X ";X
2Ø    Y= -((X<1Ø)*2+(X>1Ø)*5)
3Ø    PRINT"Y=";Y
4Ø    GOTO1Ø
```

Now try entering different values for the variable X.
If you change the leading minus sign to a plus sign in line 20, Y will return negative numbers of -2 or -5. The reason for this is that when using these logical operators, if a statement is true the computer returns a value of -1, and if it is false returns a zero Ø. So if you negate a negative number it becomes positive. In the first instance above if X is less than 10 Y will equal --1*2 , which is +2.

Remember that all Pet POKE codes are different to the SHARPS, so never enter them exactly as they are written, if you are not sure what they do leave them out and try running the program without them. Occasionally you may see POKEs around 152,158 or 512 coupled with a WAIT statement, you can probably get round these by using the GET statement. GOOD LUCK.

6
Sorting Data

There comes a time when the game playing has to make way for some more serious applications, even if it is only to justify the purchase of the expensive item of hardware sitting in the corner to ones spouse, or to show visitors that your Micro can perform a variety of tasks, other than playing Star-trek continously.

The next chapters concentrate on two aspects which should prove useful, the first is concerned with sorting of data into numerical order, whereas the second explains how to store data on tape for future reference and updating.

On page 66 of the Sharp manual is a small program which sorts a series of numbers into order. The method used is called a Bubble-sort. Quite simply the program runs through the list of numbers and selects the highest number and places it at the top of the list, afterwhich it runs again and selects the next lowest number and places it in position two of the list. The program continues until all the numbers are in order. Whilst this Bubble-sort method is perfectly adequate for lists of numbers up to 50 or 60, it becomes exceptionally slower for longer lists. In fact if it had to place in order a list of 255 numbers it would take about 4 minutes. In the next examples I intend to show how we can cut that 4 minutes down to 1 minute by using an alternative method of sorting. Although this may be a reduction in time of 75%, some might say that it still is not fast, unfortunately the only way to improve on this figure is to program in Assembly language, which is another story, for the purposes of this book which is concerned with Basic, and showing various routines to enable us to write better programs, we will continue with Basic.

We are going to enter the first program which generates a series of random numbers and sorts them into order using the Bubble-sort method. We will then enter a second program which uses a faster method, and you can compare the results achieved. You will be prompted to enter the amount of numbers to be sorted.

```
1ØØ   INPUT"TOTAL OF NUMBERS TO SORT ";N
11Ø   DIM B(N)
12Ø   FOR X = 1 TO N
13Ø   B(X)=INT(RND(1)*1ØØ)+1
14Ø   PRINT B(X);
15Ø   NEXT X
16Ø   PRINT:PRINT"THESE ARE THE NUMBERS TO BE SORTED"
17Ø   PRINT"THE TIME STARTS NOW"
18Ø   TI$="000000"
19Ø   FOR T=N TO 1 STEP-1:M=Ø
2ØØ   FOR S=1 TO T
21Ø   IF B(S)<=M THEN 23Ø
22Ø   M=B(S):L=S
23Ø   NEXT S
24Ø   B1=B(L):B(L)=B(T):B(T)=B1
25Ø   NEXT T
26Ø   A$=TI$
27Ø   PRINT:PRINT"TIME TAKEN TO SORT";N;" NUMBERS WAS"
28Ø   PRINT A$:PRINT
29Ø   FOR X=1 TO N
3ØØ   PRINT B(X);
31Ø   NEXT X
32Ø   END
```

This is a typical Bubble-sort program and is adequate for sorting numbers up to a total of 50 or 60. The variable N is the total of numbers to sort, and the array B(X) contains the actual numbers. Lines 1ØØ to 15Ø generate these numbers and in a typical program these would be omitted. The actual sorting takes place in lines 19Ø to 25Ø, and are printed out in lines 29Ø to 31Ø.

If you now RUN the program, entering different values for N when prompted, you will see the time taken to sort. Begin by entering 10, and the time to sort will display 000001, which is very fast. Now RUN the program again using higher numbers start with 20 and keep increasing it up to, and no higher than 255, as a higher number than this will produce an error message, and make a note of the times taken. Now enter the next program.

Type in these additional lines:-

```
185   GOTO 33Ø
33Ø   M=12:DIM LH(M),RH(M)
34Ø   G=1:LH(1)=1:RH(1)=N
35Ø   L=LH(G):R=RH(G):G=G-1
36Ø   I=L:J=R:Y=B(INT((L+R)/2))
37Ø   IF B(I)<Y THEN I=I+1:GOTO37Ø
38Ø   IF Y<B(J) THEN J=J-1:GOTO38Ø
39Ø   IF I>J THEN42Ø
4ØØ   W=B(I):B(I)=B(J):B(J)=W
41Ø   I=I+1:J=J-1
42Ø   IF I<=J THEN37Ø
43Ø   IF I>=R THEN45Ø
44Ø   G=G+1:LH(G)=I:RH(G)=R
45Ø   R=J
46Ø   IF L<R THEN36Ø
47Ø   IF G<>Ø THEN35Ø
48Ø   GOTO26Ø
```

If you RUN this program and compare the subsequent times, it will be seen that the second program is faster for quantities over 60.

TYPICAL TIMES:-

No. of items to sort.	Bubble-sort	Quicksort
10	01	01
20	02	02
30	04	04
40	07	05
50	10	07
60	15	09
80	26	12
100	40	15
200	2.35	37
255	4.00	46

The actual sorting takes place in lines 33Ø to 47Ø, and this could be entered as a subroutine and called whenever sorting is required. It contains more code than the Bubble-sort, but as can be seen, is far quicker in processing if large amounts have to be sorted.

This sorting can be applied to strings, and this will be seen in the next chapter.

7

Data Tape Handling

7.1 Address List Program

This chapter contains an address list program which makes good use of the data file handling of the MZ-80K, which after explanation will show you the techniques used can be written into any type of program where lists and corresponding data records need to be made. The program is menu driven, which means the screen prints a list of options that the user can make.

```
NUMBER OF ENTRIES IS 30

        SELECT FROM LIST

    1   PRINTOUT
    2   NEW ENTRIES
    3   DELETIONS
    4   SORTING BY SURNAME
    5   CHANGE DETAILS
    6   SAVE DATA

  ***** ENTER 1 to 6 *****
```

On selection of an option the program branches to a routine to carry out that specific task, and on completion returns to the menu. It should be remembered that if you feel the urge to write a similar program, that each module of an option can be self contained, written and tested separately. This way you will not get tied up in knots if it is a long listing, because the whole program will be split into several small programs each being capable of being called from the main menu. One application of this program could be to list the names and addresses of a club membership, up to 255 separate entries can be made. Or if you have a printer linked to your Sharp it could be used to generate address labels, you would of course need to alter the lines containing PRINT statements to PRINT/P, but that is fairly simple once the program is up and running.

As Basic SP-5025 does not allow full string comparisons, a machine code routine is POKEd into the top section of memory at the begining of the program with a GOSUB 219Ø.

This routine enables the Sharp to sort by surname, in that it can decide that SMITH should be placed before SMYTH in a sorted list, something that it cannot normally achieve. This machine code routine is only encountered once during the program run, and for that reason is entered towards the end of the listing, along with the titles for the same reason. As in most micros whenever a GOSUB or GOTO statement is encountered the search for that particular line number begins at the start of the listing and works down line by line until it finds it. Normally on short programs this will not be of any consequence so far as execution time is concerned, but on longer listings, and in particular routines such as the sorting routine used here, the amount of times that a GOTO command is met means that the earlier in a listing it can be placed the shorter the execution time will be. The sorting routine, which was described in the last chapter, has been altered to accomodate strings and is in lines 110 to 330. Although we saw how actual sorting times could be reduced in the last chapter, this sorting as far as strings is concerned is not as fast as one would like, mainly due to the time in moving the strings related to each surname, such as the addresses and telephone numbers. When the correct position of the surname has been made the relevant strings associated with it have to be changed to the new position, and this takes time. A good machine code subroutine could be used here, as the time would reduce drastically. You will only notice the time delay on long lists, and if this occurs leave the sorting until last on your actual run.

The only alteration to the listing that might have to be made is in line 226Ø, where you must enter the size of your memory, 24,36,or 48, as this is where it calculates the top of memory to enable the machine code routine to be entered.

Line 38Ø asks if there is data already on tape. The first time you run the answer will be no, and the program skips the INPUT routine from tape and jumps to the menu. The INPUT/T lines must have their variables in the same order as the PRINT/T lines, (440 to 490 and 213Ø to 2180)otherwise errors will occur.

One point which should be mentioned is, if you ever BREAK a program during a ROPEN or WOPEN command has been executed, you will not be able to carry on by entering CONT, as the file has not been closed. You must enter in direct mode CLOSE before carrying on with a GOTO (line number) command.

The menu starts on line 5ØØ and depending on which option is selected (1to6), branches with an 'ON Z GOTO' command in line 65Ø. The option number is tested in line 64Ø, to ensure only a number between 1 and 6 was entered. Here we see good use of the logical 'OR' in the form of a '+' sign as described earlier.

```
64Ø   IF(Z<1)+(Z>6)THEN62Ø
```

which means simply if Z is less than 1 OR if Z is greater than 6 , go back to line 62Ø and wait for another input.

The printout of names and addresses begins in line 66Ø , and again you are asked to enter either all the names or only one group. If any of the names need to be checked, the scrolling can be halted by pressing any key, afterwhich it can be restarted by any key too.

All the other sections are preceeded by REM statements and can be found easily. Remember before you save any data on tape to place an empty cassette in the recorder, as soon as this option is selected you will be prompted to "RECORD PLAY".

The assembly language routine for sorting by surname starts in line 219Ø and is contained in DATA statements. Be careful to enter these lines correctly as an error here will result in the program crashing whenever sorting takes place.

The titles are contained in lines 235Ø to 247Ø and can be changed to your own requirements.

```
      ************** ADDRESS LIST PROGRAM *************
1ØØ   GOSUB 219Ø:GOTO 34Ø
11Ø   REM  SORT ROUTINE
12Ø   PRINT"▢SORTING":PRINT"▼▼▼▼PLEASE BE PATIENT"
13Ø   POKE88Ø5,BL:POKE88Ø6,BH
14Ø   S=1:SL(1)=1:SR(1)=EN
15Ø   L=SL(S):R=SR(S):S=S-1
```

```
16Ø   X=L:J=R:X$=SN$(INT((L+R)/2))
17Ø   IF SN$(X)<X$ THEN X=X+1:GOTO17Ø
18Ø   IF X$<SN$(J) THEN J=J-1:GOTO18Ø
19Ø   IF X>J THEN 27Ø
2ØØ   W$=SN$(X):SN$(X)=SN$(J):SN$(J)=W$
21Ø   E$=CN$(X):CN$(X)=CN$(J):CN$(J)=E$
22Ø   R$=AD$(X):AD$(X)=AD$(J):AD$(J)=R$
23Ø   T$=TN$(X):TN$(X)=TN$(J):TN$(J)=T$
24Ø   Y$=PC$(X):PC$(X)=PC$(J):PC$(J)=Y$
25Ø   O=C(X):C(X)=C(J):C(J)=O
26Ø   X=X+1:J=J-1
27Ø   IF X<=J THEN17Ø
28Ø   IF X>=R THEN3ØØ
29Ø   S=S+1:SL(S)=X:SR(S)=R
3ØØ   R=J
31Ø   IF L<R THEN 16Ø
32Ø   IF S<>Ø THEN 15Ø
33Ø   GOTO 51Ø
34Ø   M=12:DIM SL(M),SR(M)
35Ø   DIM CN$(15Ø),SN$(15Ø),AD$(15Ø)
36Ø   DIM TN$(15Ø),PC$(15Ø),C(15Ø),H(1ØØ)
37Ø   X=Ø:EN=Ø
38Ø   PRINT:PRINT:PRINT"IS THERE ANY DATA ON TAPE (Y/N)":USR(62)
39Ø   GETA$:IF A$=""THEN 39Ø
4ØØ   IF A$="Y" THEN 44Ø
41Ø   IF A$="N" THEN 51Ø
42Ø   GOTO 39Ø
43Ø   REM *****READ DATA TAPE****
44Ø   ROPEN"MAIL LIST DATA"
45Ø   INPUT/T EN
46Ø   FOR X =1 TO EN
47Ø   INPUT/T CN$(X),SN$(X),AD$(X),TN$(X),PC$(X),C(X)
48Ø   NEXT X
49Ø   CLOSE
5ØØ   REM *****MENU*****
51Ø   PRINT"C──────────────────────────────────────────────"
52Ø   PRINT"          NUMBER OF ENTRIES IS";EN
53Ø   PRINT"──────────────────────────────────────────────"
54Ø   PRINT"↓→→→→→→→→→→→→→SELECT FROM LIST":TC=EN
```

```
55Ø   PRINT"→→→↓1   PRINTOUT"
56Ø   PRINT"→→→↓2   NEW ENTRIES"
57Ø   PRINT"→→→↓3   DELETIONS"
58Ø   PRINT"→→→↓4   SORT BY SURNAME"
59Ø   PRINT"→→→↓5   CHANGE OF DETAILS"
6ØØ   PRINT"→→→↓6   SAVE DATA"
61Ø   PRINT"↓↓  ********* ENTER 1 to 6 *********":USR(62)
62Ø   GETA$:IF A$="" THEN62Ø
63Ø   Z=VAL(A$)
64Ø   IF (Z<1)+(Z>6) THEN62Ø
65Ø   ON Z GOTO 67Ø,1Ø2Ø,132Ø,11Ø,155Ø,2Ø9Ø
66Ø   REM ****** PRINTOUT ******
67Ø   PRINT"C       PRINTOUT ALL (ENTER A)"
68Ø   PRINT"OR ONLY ONE GROUP (ENTER 1to6)"
69Ø   GETZ$:IF Z$="" THEN69Ø
7ØØ   IF Z$="A" THEN N=1:GOTO83Ø
71Ø   Z=VAL(Z$)
72Ø   IF (Z<1)+(Z>6) THEN67Ø
73Ø   N=Ø
74Ø   FOR X=1 TO EN:IF C(X)<>Z THEN 81Ø
75Ø   N=1
76Ø   PRINT CN$(X);SN$(X);" ";AD$(X)
77Ø   PRINT TN$(X);" ";PC$(X)
78Ø   PRINT
79Ø   GETZ$:IF Z$="" THEN 81Ø
8ØØ   GETQ$:IF Q$="" THEN 8ØØ
81Ø   NEXT X
82Ø   GOTO 96Ø
83Ø   FOR X=1 TO EN STEP 2
84Ø   PRINT CN$(X);SN$(X);TAB(2Ø);CN$(X+1);SN$(X+1)
85Ø   PRINT AD$(X);TAB(2Ø);AD$(X+1)
86Ø   PRINT TN$(X);TAB(2Ø);TN$(X+1)
87Ø   PRINT PC$(X);TAB(17);C(X);TAB(2Ø);PC$(X+1);TAB(37);C(X+1)
88Ø   GETZ$:IF Z$="" THEN 93Ø
89Ø   PRINT"↓ANY KEY TO CONTINUE....ZERO TO QUIT"
9ØØ   GETQ$:IF Q$="" THEN 9ØØ
91Ø   IF Q$="Ø" THEN EN=TC:GOTO 51Ø
92Ø   PRINT"↑";SPC(38);"↑↑"
```

```
93Ø   PRINT
94Ø   NEXT X
95Ø   IF SN$(X-1)="" THEN EN=X-2:GOTO 98Ø
96Ø   EN=X-1
97Ø   IF N=Ø THEN PRINT"↓NO ENTRIES IN GROUP";Z
98Ø   PRINT"↓DONE..PRESS ANY KEY":USR(62)
99Ø   GETZ$:IF Z$="" THEN 99Ø
1ØØØ  GOTO 51Ø
1Ø1Ø  REM ****** NEW ENTRIES ******
1Ø2Ø  X=EN
1Ø3Ø  X=X+1
1Ø4Ø  PRINT"CIF THERE ARE NO INITIALS ENTER ZERO (Ø)"
1Ø5Ø  PRINT"↓↓NUMBER";X:USR(62)
1Ø6Ø  INPUT"↓FIRSTNAME/INITIALS   ";CN$(X)
1Ø7Ø  IF ASC(CN$(X))=48 THEN CN$(X)="":GOTO1Ø9Ø
1Ø8Ø  CN$(X)=CN$(X)+" "
1Ø9Ø  PRINT"*****IF ENTRY NOT REQUIRED TYPE Ø*****"
11ØØ  INPUT"↓      LASTNAME        ";SN$(X)
111Ø  IF ASC(SN$(X))=48 THEN GOTO 51Ø
112Ø  PRINT"↑↑↑";SPC(38);"↓↓↓"
113Ø  INPUT"↓  No.& STREET         ";AD$(X)
114Ø  IF LEN(AD$(X))>19 THEN PRINT"TOTAL 19 LETTERS PLEASE
                                   ABBREVIATE":GOTO 113Ø
115Ø  INPUT"↓TOWN & COUNTY         ";TN$(X)
116Ø  IF LEN(TN$(X))>19 THEN PRINT"TOTAL 19 LETTERS PLEASE
                                   ABBREVIATE":GOTO 115Ø
117Ø  INPUT"↓TELEPHONE             ";PC$(X)
118Ø  IF LEN(PC$(X))<2 THEN PC$(X)=""
119Ø  INPUT"↓USERS CODE (1 to 6)   ";C(X)
12ØØ  IF (C(X)<1)+(C(X)>6) THEN 119Ø
121Ø  PRINT"↓↓→→→→→IS ALL THE ABOVE CORRECT(Y/N)":USR(62)
122Ø  GETA$:IF A$="" THEN122Ø
123Ø  IF A$="Y" THEN 126Ø
124Ø  IF A$="N" THEN GOSUB 188Ø:GOTO 126Ø
125Ø  GOTO 122Ø
126Ø  PRINT"↓IS THERE MORE DATA TO ADD":USR(62)
127Ø  GETA$:IF A$="" THEN 127Ø
```

```
1280  IF A$="Y" THEN 1030
1290  IF A$="N" THEN EN=X:TC=EN:GOTO 510
1300  GOTO 1270
1310  REM ******DELETIONS******
1320  PRINT"CDELETIONS:-":X=0
1330  USR(62):GOSUB 1640:IF ASC(DT$)=48 THEN 510
1340  IF V=0 THEN 1330
1350  PRINT
1360  PRINT"-------------------------------------"
1370  INPUT"INPUT No. TO DELETE(0 returns to menu)";L
1380  IF L=0 THEN 510
1390  PRINT CN$(L);SN$(L)
1400  PRINT AD$(L)
1410  PRINT TN$(L);" ";PC$(L)
1420  PRINT"↓ARE YOU SURE (Y or N)":USR(62)
1430  GETZ$:IF Z$=""THEN1430
1440  IF Z$="N" THEN1320
1450  IF Z$="Y" THEN PRINT"↓↓WAIT..AM MOVING DATA":GOTO 1470
1460  GOTO 1430
1470  CN$(L)="":SN$(L)="":AD$(L)="":TN$(L)="":PC$(L)="":C(L)=0
1480  FOR X=L TO EN
1490  CN$(X)=CN$(X+1):SN$(X)=SN$(X+1)
1500  AD$(X)=AD$(X+1):TN$(X)=TN$(X+1)
1510  PC$(X)=PC$(X+1):C(X)=C(X+1)
1520  NEXT X
1530  EN=EN-1
1540  GOTO 1320
1550  REM ******DETAIL CHANGES******
1560  PRINT"CCHANGE OF DETAILS"
1570  GOSUB 1640:IF ASC(DT$)=48 THEN 510
1580  IF V=0 THEN 1570
1590  PRINT
1600  PRINT"-------------------------------------"
1610  INPUT"INPUT No. TO CHANGE (0 returns to menu)";X
1620  IF X=0 THEN 510
1630  GOSUB 1880:GOTO 1570
1640  INPUT"↓PLEASE ENTER SURNAME (0 returns to menu)";DT$
```

```
165Ø  IF ASC(DT$)=48 THEN RETURN
166Ø  V=Ø
167Ø  FOR X=1 TO EN
168Ø  IF SN$(X)=DT$ THEN V=V+1:H(V)=X
169Ø  NEXT X
17ØØ  IF V<>Ø THEN 181Ø
171Ø  PRINT"↓NO CORRESPONDING NAME FOUND"
172Ø  PRINT"PERHAPS THE SPELLING IS WRONG"
173Ø  PRINT"↓I WILL PRINTOUT SIMILAR ENTRIES"
174Ø  POKE 88Ø5,114:POKE 88Ø6,34
175Ø  FOR X=1 TO EN
176Ø  IF LEFT$(SN$(X),3)=LEFT$(DT$,3) THEN V=V+1:H(V)=X
177Ø  NEXT X
178Ø  IF V<>Ø THEN 181Ø
179Ø  PRINT"↓I CANNOT FIND IT PLEASE RE-ENTER"
18ØØ  RETURN
181Ø  PRINT"↓"
182Ø  PRINT"No.↓"
183Ø  FOR X=1 TO V
184Ø  P=H(X)
185Ø  PRINT P;"  ";CN$(P);SN$(P);" ";AD$(P)
186Ø  NEXT X
187Ø  RETURN
188Ø  REM ******CHANGES******
189Ø  PRINT TAB(8);"1 ";CN$(X)
19ØØ  PRINT TAB(8);"2 ";SN$(X)
191Ø  PRINT TAB(8);"3 ";AD$(X)
192Ø  PRINT TAB(8);"4 ";TN$(X)
193Ø  PRINT TAB(8);"5 ";PC$(X)
194Ø  PRINT TAB(8);"6";C(X)
195Ø  PRINT
196Ø  PRINT"-------------------------------------"
197Ø  INPUT"No. OF LINE TO CHANGE (IF O/K TYPE Ø)";N
198Ø  IF (N<Ø)+(N>6) THEN PRINT"↑":GOTO 197Ø
199Ø  IF N=Ø THEN RETURN
2ØØØ  PRINT"↑↑******PRESS 'CR' WHEN CORRECTED******"
2Ø1Ø  PRINT SPC(38)
2Ø2Ø  ON N GOTO 2Ø3Ø,2Ø4Ø,2Ø5Ø,2Ø6Ø,2Ø7Ø,2Ø8Ø
```

```
2Ø3Ø  INPUT"↑↑↑↑↑↑↑↑↑↑1.st NAME ";CN$(X):PRINT"C":CN$(X)=
                                        CN$(X)+" ":GOTO 189Ø
2Ø4Ø  INPUT"↑↑↑↑↑↑↑↑LASTNAME  ";SN$(X):PRINT"C":GOTO 189Ø
2Ø5Ø  INPUT"↑↑↑↑↑↑↑↑No. & ST. ";AD$(X):PRINT"C":GOTO 189Ø
2Ø6Ø  INPUT"↑↑↑↑↑↑TN. & CY. ";TN$(X):PRINT"C":GOTO 189Ø
2Ø7Ø  INPUT"↑↑↑↑↑TELEPHONE ";PC$(X):PRINT"C":GOTO 189Ø
2Ø8Ø  INPUT"↑↑↑↑USER CODE ";C(X):PRINT"C":GOTO 189Ø
2Ø9Ø  REM ******SAVE DATA******
21ØØ  PRINT"CPRESS 'CR' WHEN DATA TAPE IS REWOUND"
211Ø  GETZ$:IF Z$="" THEN 211Ø
212Ø  IF ASC(Z$)<>102 THEN 211Ø
213Ø  WOPEN"MAIL LIST DATA"
214Ø  PRINT/T EN
215Ø  FOR X=1 TO EN
216Ø  PRINT/T CN$(X),SN$(X),AD$(X),TN$(X),PC$(X),C(X)
217Ø  NEXT X
218Ø  CLOSE:GOTO 51Ø
219Ø  DATA121,213,245,2Ø5,174,34,2Ø5,15,24,241,145,48,1,175,
                                          129,71,227,2Ø5
22ØØ  DATA1,24,235,2Ø5,1,24,12Ø,176,4Ø,8,26,19Ø,35,19,32,2,
                                        16,248,245,8,214
221Ø  DATA176,4Ø,26,61,4Ø,23,61,4Ø,25,61,40,22,61,4Ø,24,61,
                                        4Ø,21,61,4Ø,23,61
222Ø  DATA4Ø,25,241,48,32,24,25,241,32,22,24,25,241,56,17,24,
                                          6,241,56,17,24
223Ø  DATA1Ø,241,4Ø,7,24,1Ø,241,56,7,24,232,17,3Ø,22,24,3,17,
                                         25,22,2Ø5,26,24
224Ø  DATA225,2Ø5,123,35,195,91,34
225Ø  REM **INPUT MEM SIZE ON NEXT LINE
226Ø  SZ=48
227Ø  LM=(SZ+4)*1Ø24-111
228Ø  LIMIT LM
229Ø  FOR N=LM TO LM+11Ø
23ØØ  READ A
231Ø  POKE N,A
232Ø  NEXT N
233Ø  BH=INT(LM/256):BL=INT(LM-256*BH+Ø.5)
234Ø  POKE 88Ø5,BL:POKE 88Ø6,BH
235Ø  PRINT"C↓↓↓↓"
236Ø  PRINT"→→→*    *   *   * *"
237Ø  PRINT"→→→**  **  * *  * *"
238Ø  PRINT"→→→* ** * ***** * *"
```

```
2390  PRINT"[→→→]*     * *    * * *"
2400  PRINT"[→→→]*     * *    * * *****"
2410  PRINT:PRINT
2420  PRINT"[→→→→→→→→]*      *  **** *****"
2430  PRINT"[→→→→→→→→]*      * *        *"
2440  PRINT"[→→→→→→→→]*      *  ***     *"
2450  PRINT"[→→→→→→→→]*      *      *   *"
2460  PRINT"[→→→→→→→→]***** * *****    *"
2470  RETURN
```

If you have requested a search for a name and the program has not found it, then it will search for similar names that start with the same three letters, and printout all the names that it finds. If you wish to alter this, to perhaps names which only start with the same letter instead of the first three, then change line 1760. At the moment it reads:-

```
IF LEFT$(SN$(X),3)=LEFT$(DT$,3)THEN V=V+1:H(V)=X
```

Change to:-

```
IF LEFT$(SN$(X),1)=LEFT$(DT$,1)THEN V=V+1:H(V)=X
```

The program could be tightened up by entering more statements on each line, but for clarity and legibility it should be typed in as shown in order to assist any changes that you wish to make.

7.2 Stock Control Program

The following program which deals with Stock Control, also demonstrates the file handling aspects of the Sharp, and is capable of maintaining records for up to 256 different categories of stock. It does not pretend to be capable of running a large business, but could easily be used in keeping smaller records and recording the relevant data on tape.
If one runs this program using data on tape just imagine how fast it would run if one was using Discs, which for any serious user is a natural progression. Nevertheless if one can be patient while the data is loading it works quite adequately.

The first time you run the program you will answer no 'N' to the 'IS THERE DATA ON TAPE' prompt, and the program will jump to the menu and display a group of options.
Selection of option 2 will allow you to commence entering the stock you hold, afterwhich any of the options can be selected.
At any time you may return to the menu by entering Ø (zero).
If you wish to finish and have not previously recorded the data you have entered, you will be reminded of the fact and asked if you wish to save data. If you reply 'Y' the program will return to the menu in order to allow you to select option 8, 'SAVE DATA' ,otherwise the program will finish.
If while testing the program you wish to BREAK and subsequently restart, enter GOTO250 , and you will skip the initialisation sequence and go directly to the menu.
When entering the TRADE price of a stock item always type in the price excluding VAT, you will be prompted for the current rate of VAT afterwards. If the item carries no VAT enter Ø when prompted.
If at any time you find the list contains an item of stock which you no longer carry, it can be changed to an alternative by entering option 7 'CHANGE NAME'.
The mobile cursor displayed on the menu is controlled in lines 39Ø to 4ØØ within a loop.
Change line 194Ø to a name of your choice.
The most used messages are contained in strings in lines 194Ø to 2Ø2Ø.

```
        ****STOCK CONTROL PROGRAM****
1ØØ   GOSUB 186Ø:GOTO 15Ø
11Ø   RS = Ø:GET Z$:IF Z$ = ""THEN 11Ø
12Ø   IF Z$ = "Y" THEN RS = 1:RETURN
13Ø   IF Z$ = "N" THEN RS = 2:RETURN
14Ø   GOTO 11Ø
15Ø   PRINT"□IS THERE DATA ON TAPE?(Y/N)":USR(62)
16Ø   GOSUB 11Ø
17Ø   ON RS GOTO 18Ø,25Ø
18Ø   PRINT"↓INSERT DATA TAPE"
19Ø   ROPEN "STOCK DATA"
2ØØ   INPUT/T DA$,EN
21Ø   FOR N = 1 TO EN
22Ø   INPUT/T SN(N),BN$(N),TP(N),SP(N),VT(N),SA(N)
23Ø   NEXT N
24Ø   CLOSE
25Ø   IF DA$ = ""THEN DA$ = "ØØ/ØØ/ØØ"
26Ø   PRINT A$:PRINT B$
27Ø   PRINT E$;DA$
28Ø   PRINT"--------------------------------------"
29Ø   PRINT"↓→→→→→key  1 for TOTAL STOCK PRINTOUT"
3ØØ   PRINT"↓→→→→→→'   2  '  NEW STOCK ENTRIES"
31Ø   PRINT"↓→→→→→→'   3  '  STOCK BOUGHT"
32Ø   PRINT"↓→→→→→→'   4  '  PRICE CHANGES"
33Ø   PRINT"↓→→→→→→'   5  '  STOCK SOLD"
34Ø   PRINT"↓→→→→→→'   6  '  STOCK TAKE CHECK"
35Ø   PRINT"↓→→→→→→'   7  '  CHANGE NAME"
36Ø   PRINT"↓→→→→→→'   8  '  SAVE DATA"
37Ø   PRINT"↓→→→→→→'   9  '  FINISH"
38Ø   PRINT"↓→→→→→→please enter option (1 to 9)":USR(62)
39Ø   PRINT"H↓↓↓↓":C = Ø
4ØØ   GET Z$:IF VAL(Z$)>Ø THEN 43Ø
41Ø   PRINT"↓→→→→→→→→→→→>";:FOR X = 1 TO 15Ø:NEXT X:PRINT"< ":
                                        C = C+1:IF C = 9 THEN 39Ø
42Ø   GOTO 4ØØ
43Ø   Z = VAL(Z$)
44Ø   IF (Z<1)+(Z>9) THEN 4ØØ
45Ø   ON Z GOTO 46Ø,78Ø,99Ø,123Ø,132Ø,145Ø,162Ø,17ØØ,181Ø
```

```
46Ø   PRINT A$:PRINT B$
47Ø   PRINT"DATE ";DA$
48Ø   TT = Ø:TR = Ø:TV = Ø:RV = Ø:VD =1
49Ø   PRINT SN$;TAB(22);TP$;SP$;SA$
5ØØ   FOR N = 1 TO EN
51Ø   PRINT SN(N);TAB(5);BN$(N);
52Ø   PRINT TAB(24-LEN(STR$(INT(TP(N)))));TP(N);
53Ø   PRINT TAB(3Ø-LEN(STR$(INT(SP(N)))));SP(N);
54Ø   PRINT TAB(38-LEN(STR$(INT(SA(N)))));SA(N)
55Ø   TT = TT+(TP(N)*SA(N)):TR = TR+(SP(N)*SA(N))
56Ø   IF VT(N) = Ø THEN 59Ø
57Ø   TV = TV+INT((VT(N)*TP(N)*SA(N))*1ØØ)/1ØØ
58Ø   RV = RV+INT((SA(N)*SP(N)*1ØØ)-(SA(N)*SP(N)/(VT(N)+1)*1ØØ))/1ØØ
59Ø   IF INT(VD/2Ø)=VD/2Ø THEN PRINT"↓THERE IS MORE PRESS
                                 ANY KEY":USR(62):GOTO 61Ø
6ØØ   GOTO 63Ø
61Ø   GET Z$:IF Z$ = "" THEN 61Ø
62Ø   PRINT"↑";SPC(38);"↑↑"
63Ø   VD = VD + 1:NEXT N
64Ø   PRINT L$
65Ø   PRINT"↓→→→→→→→→STOCK VALUE £";
66Ø   PRINT TAB(26-LEN(STR$(INT(TT))));TT
67Ø   PRINT"→→→→→TOTAL WITH VAT £";
68Ø   PRINT TAB(26-LEN(STR$(INT(TT+TV))));TT+TV
69Ø   PRINT"↓→→→→→→→RETAIL RETURN £";
7ØØ   PRINT TAB(26-LEN(STR$(INT(TR))));TR
71Ø   PRINT"→→→→→→→→VAT CONTENT £";
72Ø   PRINT TAB(26-LEN(STR$(INT(RV))));RV
73Ø   PRINT"↓NET PROFIT EXPECTED £";
74Ø   PRINT TAB(26-LEN(STR$(INT(TR-RV-TT))));TR-RV-TT
75Ø   PRINT"→→→→→→→press any key to continue":USR(62)
76Ø   GET Q$:IF Q$ = "" THEN 76Ø
77Ø   GOTO 25Ø
78Ø   PRINT"CNEW LINES"
79Ø   PRINT L$
8ØØ   N = EN+1:EX = Ø
81Ø   PRINT"STOCK No.      ";N:SN(N)=N
82Ø   PRINT"↓ZERO RETURNS TO MENU"
```

```
83Ø   INPUT"DESC.OF GOODS ";NN$
84Ø   IF LEN(NN$)>17 THEN PRINT"17 LETTERS IS MAX PLEASE
                                   ABBREVIATE":GOTO 83Ø
85Ø   IF NN$ = "Ø" THEN 25Ø
86Ø   BN$(N)=NN$
87Ø   INPUT"AMOUNT BOUGHT ";SA(N)
88Ø   INPUT"VAT RATE %     ";VT(N):VT(N)=VT(N)/1ØØ
89Ø   PRINT"↓→→→→→→→→→→→→→→in pence"
9ØØ   INPUT"TRADE PRICE    ";TP(N):TP(N)=TP(N)/1ØØ
91Ø   INPUT"SELLING PRICE ";SP(N):SP(N)=SP(N)/1ØØ
92Ø   IF SP(N)<TP(N) THEN PRINT D$:GOSUB 2Ø4Ø
93Ø   PRINT H$:GOSUB 11Ø
94Ø   IF (EX=1)*(RS=1) THEN 25Ø
95Ø   IF RS=1 THEN EN = EN+1:GOTO 97Ø
96Ø   IF RS=2 THEN 81Ø
97Ø   PRINT"↓↓ARE THERE ANY MORE TO ENTER?(Y/N)":USR(62):GOSUB 11Ø
98Ø   ON RS GOTO 78Ø,25Ø
99Ø   PRINT"CSTOCK BOUGHT"
1ØØØ  PRINT L$
1Ø1Ø  INPUT"INPUT STOCK No. (zero to quit)  ";N
1Ø2Ø  IF N = Ø THEN 25Ø
1Ø3Ø  IF N>EN THEN PRINT F$:GOTO 1ØØØ
1Ø4Ø  PRINT"↓";SN$;SN(N);"→";BN$(N)
1Ø5Ø  PRINT"↓if stock no. does not match type Ø"
1Ø6Ø  PRINT"STOCK HELD WAS    ";SA(N)
1Ø7Ø  INPUT"AMOUNT BOUGHT ?    ";AM
1Ø8Ø  IF AM = Ø THEN 1ØØØ
1Ø9Ø  SA(N)=SA(N)+AM
11ØØ  PRINT"STOCK HELD IS NOW ";SA(N)
111Ø  PRINT"↓↓IS TRADE PRICE STILL £";TP(N);" (Y/N)":USR(62):
                                                GOSUB 11Ø
112Ø  IF RS = 1 THEN 1ØØØ
113Ø  INPUT"NEW TRADE PRICE (in pence) ";J:J=J/1ØØ:PRINT"→£";J;
                                          "→";H$:GOSUB 11Ø
114Ø  IF RS = 2 THEN 111Ø
115Ø  TP(N) = J
116Ø  PRINT"↓IS SELLING PRICE STILL £";SP(N);"(Y/N)":USR(62):
                                                GOSUB 11Ø
117Ø  IF RS = 1 THEN 1ØØØ
```

```
1180  INPUT"NEW SELLING PRICE (in pence)";K:K=K/100
1190  IF K<TP(N) THEN PRINT D$:GOSUB 2040
1200  PRINT"£";K;"→";H$:USR(62):GOSUB 110
1210  IF RS = 2 THEN 1180
1220  SP(N)=K:GOTO 1000
1230  PRINT"CPRICE CHANGES"
1240  PRINT L$
1250  INPUT"↓STOCK No.OF ITEM (zero to quit) ";N
1260  IF N = 0 THEN 250
1270  IF N>EN THEN PRINT F$:GOTO 1250
1280  PRINT"↓";SN(N);BN$(N)
1290  PRINT"↓THE SELLING PRICE IS £";SP(N)
1300  INPUT"↓INPUT NEW PRICE (in pence)";J:IF J = 0 THEN 1250
1310  SP(N)=J/100:GOTO 1240
1320  PRINT"CSTOCK SOLD"
1330  PRINT L$
1340  INPUT"INPUT STOCK No. (zero to quit) ";N
1350  IF N = 0 THEN 250
1360  IF N>EN THEN PRINT F$:GOTO 1330
1370  PRINT SN(N);BN$(N)
1380  PRINT H$:GOSUB 110
1390  IF RS = 2 THEN 1330
1400  PRINT"↓STOCK WAS ";SA(N)
1410  INPUT"INPUT AMOUNT SOLD ";J
1420  IF J>SA(N) THEN PRINT C$:GOSUB 2040:GOTO 1410
1430  SA(N)=SA(N)-J
1440  PRINT"↓STOCK IS NOW ";SA(N):GOTO 1330
1450  PRINT"CSTOCK TAKE CHECK"
1460  PRINT L$
1470  PRINT SN$;TAB(23);"WAS   IS   SOLD"
1480  FOR N = 1 TO EN
1490  PRINT SN(N);TAB(5);BN$(N);TAB(25-LEN(STR$(INT(SA(N)))));
                                                SA(N);TAB(27);
1500  INPUT J:IF J>SA(N) THEN PRINT G$:GOSUB 2040:GOTO 1490
1510  CR=SP(N)*(SA(N)-J)+CR
1520  IF VT(N)=0 THEN 1540
1530  VA=VA+INT(((SA(N)-J)*SP(N)*100)-((SA(N)-J)*SP(N)/(VT(N)+1)
                                                    *100))/100
```

```
1540  PRINT"↑";TAB(34);SA(N)-J:SA(N)=J
1550  NEXT N
1560  PRINT"↓TOTAL SALES = £";TAB(17-LEN(STR$(INT(CR))));CR
1570  PRINT"↓VAT CONTENT = £";TAB(17-LEN(STR$(INT(VA))));VA
1580  PRINT"↓";L$
1590  PRINT"↓↓PRESS ANY KEY TO CONTINUE":USR(62)
1600  GET Q$:IF Q$ = "" THEN 1600
1610  GOTO 250
1620  PRINT"CCHANGE NAME"
1630  PRINT L$
1640  INPUT"STOCK No.OF ITEM TO ALTER (zero to quit) ";N
1650  IF N = 0 THEN 250
1660  IF N>EN THEN PRINT F$:GOTO 1630
1670  PRINT:PRINT BN$(N);H$:GOSUB 110
1680  IF RS = 1 THEN EX = 1:GOTO 830
1690  GOTO 1630
1700  PRINT"CBEFORE SAVING DATA PLEASE ENTER"
1710  INPUT"TODAYS DATE ";DA$
1720  PRINT H$;DA$:USR(62):GOSUB 110
1730  IF RS = 2 THEN 1710
1740  PRINT"CINSERT DATA TAPE & REWIND"
1750  WOPEN"STOCK DATA"
1760  PRINT/T DA$,EN
1770  FOR N = 1 TO EN
1780  PRINT/T SN(N),BN$(N),TP(N),SP(N),VT(N),SA(N)
1790  NEXT N
1800  CLOSE:DS = 1:GOTO 250
1810  IF DS<>1 THEN PRINT"YOU HAVE NOT SAVED DATA DO YOU WANT TO?":
                                                          GOTO 1830
1820  END
1830  USR(62):GET Z$:IF Z$ = "" THEN PRINT"↑";SPC(39);"↑":GOTO 1810
1840  IF Z$ = "N" THEN 1820
1850  GOTO 250
1860  PRINT"C"
1870  PRINT"→→→→→→→→→→→→→→→STOCK CONTROL"
1880  FOR X = 1 TO 5:USR(62)
1890  FOR Y = 1 TO 500:NEXT Y
1900  POKE 59555,0:FOR Y = 1 TO 200:NEXT Y
```

```
191Ø  POKE 59555,1:NEXT X
192Ø  DIM SN(255),BN$(255),TP(255),SP(255),VT(255),SA(255)
193Ø  EN=Ø:TT=Ø:TR=Ø:TV=Ø:CR=Ø
194Ø  A$="C→→→→→→→→→→MZ-8ØK SERVICES"
195Ø  B$="→→→→→→→→→→————————————————"
196Ø  C$="**HOW CAN YOU SELL MORE THAN YOU HELD"
197Ø  D$="**HOW CAN YOU SELL FOR LESS THAN YOU BUY"
198Ø  E$="DATE OF LAST ENTRY ":F$="NO SUCH ENTRY"
199Ø  G$="STOCK CANNOT INCREASE..RE-ENTER"
2ØØØ  H$="→IS THIS CORRECT (Y/N)"
2Ø1Ø  L$="————————————————————————————————————————"
2Ø2Ø  SN$="No.":TP$=" TRADE ":SP$=" SELL ":SA$=" AMT"
2Ø3Ø  RETURN
2Ø4Ø  FOR X = 1 TO 3
2Ø5Ø  POKE 59555,Ø:USR(62):FOR Y = 1 TO 5Ø:NEXT Y
2Ø6Ø  POKE 59555,1:FOR Y = 1 TO 2ØØ:NEXT Y,X:RETURN
```

Appendix

The following listing converts either Decimal numbers to Hexadecimal, or Hexadecimals to Decimal.

```
4000  INPUT"ENTER NUMBER FOLLOWED BY (H or D)";A$
4010  IF RIGHT$(A$,1)="H" THEN 4040
4020  IF RIGHT$(A$,1)="D" THEN 4120
4030  GOTO 4000
4040  A$=LEFT$(A$,4)
4050  J=0:FOR Z=1 TO LEN(A$)
4060  K=ASC(MID$(A$,Z,1))-48
4070  IF K<10 THEN 4100
4080  K=K-7
4090  IF K>15 THEN 4000
4100  J=J*16+K:NEXT
4110  PRINT"=";J;" DEC":GOTO 4000
4120  A$=LEFT$(A$,LEN(A$)-1):C$=""
4130  B=VAL(A$):IF B>65535 THEN 4000
4140  C=65536
4150  FOR A=1 TO 4:C=C/16:K=INT(B/C)
4160  GOSUB 4190:B=B-K*C:NEXT A
4170  K=B
4180  PRINT"=";C$;" HEX":GOTO 4000
4190  IF K>9 THEN K$=CHR$(K+55):GOTO 4210
4200  K$=STR$(K)
4210  C$=C$+K$:RETURN
```

There follows a conversion table for Hexadecimal to Decimal. The first column is the Hex code, and the second and third is the Decimal equivalent. To illustrate it's use we will look at the Hex number at the end of the Monitor work area, 11FF Hex as shown on page 11. To calculate the decimal equivalent we take the first two numbers, 11, known as the Most Significant Byte (M.S.B), and convert it. As can be seen the the eqivalent is 4352 dec. We then take the second pair of numbers, the Lowest Significant Byte(L.S.B) which is FF Hex, and on conversion will see that it represents the decimal number 255. Adding these two numbers together(4352+255) will give us the decimal number of 4607, which is equivalent to 11FF Hex.

HEX	DEC *256	DEC	H	D *256	D	H	D *256	D	H	D *256	D	H	D *256	D
00	00000	0	34	13312	52	68	26624	104	9C	39936	156	D0	53248	208
01	00256	1	35	13568	53	69	26880	105	9D	40192	157	D1	53504	209
02	00512	2	36	13824	54	6A	27136	106	9E	40448	158	D2	53760	210
03	00768	3	37	14080	55	6B	27392	107	9F	40704	159	D3	54016	211
04	01024	4	38	14336	56	6C	27648	108	A0	40960	160	D4	54272	212
05	01280	5	39	14592	57	6D	27904	109	A1	41216	161	D5	54528	213
06	01536	6	3A	14848	58	6E	28160	110	A2	41472	162	D6	54784	214
07	01792	7	3B	15104	59	6F	28416	111	A3	41728	163	D7	55040	215
08	02048	8	3C	15360	60	70	28672	112	A4	41984	164	D8	55296	216
09	02304	9	3D	15616	61	71	28928	113	A5	42240	165	D9	55552	217
0A	02560	10	3E	15872	62	72	29184	114	A6	42496	166	DA	55808	218
0B	02816	11	3F	16128	63	73	29440	115	A7	42752	167	DB	56064	219
0C	03072	12	40	16384	64	74	29696	116	A8	43008	168	DC	56320	220
0D	03328	13	41	16640	65	75	29952	117	A9	43264	169	DD	56576	221
0E	03584	14	42	16896	66	76	30208	118	AA	43520	170	DE	56832	222
0F	03840	15	43	17152	67	77	30464	119	AB	43776	171	DF	57088	223
10	04096	16	44	17408	68	78	30720	120	AC	44032	172	E0	57344	224
11	04352	17	45	17664	69	79	30976	121	AD	44288	173	E1	57600	225
12	04608	18	46	17920	70	7A	31232	122	AE	44544	174	E2	57856	226
13	04864	19	47	18176	71	7B	31488	123	AF	44800	175	E3	58112	227
14	05120	20	48	18432	72	7C	31744	124	B0	45056	176	E4	58368	228
15	05376	21	49	18688	73	7D	32000	125	B1	45312	177	E5	58624	229
16	05632	22	4A	18944	74	7E	32256	126	B2	45568	178	E6	58880	230
17	05888	23	4B	19200	75	7F	32512	127	B3	45824	179	E7	59136	231
18	06144	24	4C	19456	76	80	32768	128	B4	46080	180	E8	59392	232
19	06400	25	4D	19712	77	81	33024	129	B5	46336	181	E9	59648	233
1A	06656	26	4E	19968	78	82	33280	130	B6	46592	182	EA	59904	234
1B	06912	27	4F	20224	79	83	33536	131	B7	46848	183	EB	60160	235
1C	07168	28	50	20480	80	84	33792	132	B8	47104	184	EC	60416	236
1D	07424	29	51	20736	81	85	34048	133	B9	47360	185	ED	60672	237
1E	07680	30	52	20992	82	86	34304	134	BA	47616	186	EE	60928	238
1F	07936	31	53	21248	83	87	34560	135	BB	47872	187	EF	61184	239
20	08192	32	54	21504	84	88	34816	136	BC	48128	188	F0	61440	240
21	08448	33	55	21760	85	89	35072	137	BD	48384	189	F1	61696	241
22	08704	34	56	22016	86	8A	35328	138	BE	48640	190	F2	61952	242
23	08960	35	57	22272	87	8B	35584	139	BF	48896	191	F3	62208	243
24	09216	36	58	22528	88	8C	35840	140	C0	49152	192	F4	62464	244
25	09472	37	59	22784	89	8D	36096	141	C1	49408	193	F5	62720	245
26	09728	38	5A	23040	90	8E	36352	142	C2	49664	194	F6	62976	246
27	09984	39	5B	23296	91	8F	36608	143	C3	49920	195	F7	63232	247
28	10240	40	5C	23552	92	90	36864	144	C4	50176	196	F8	63488	248
29	10496	41	5D	23808	93	91	37120	145	C5	50432	197	F9	63744	249
2A	10752	42	5E	24064	94	92	37376	146	C6	50688	198	FA	64000	250
2B	11008	43	5F	24320	95	93	37632	147	C7	50944	199	FB	64256	251
2C	11264	44	60	24576	96	94	37888	148	C8	51200	200	FC	64512	252
2D	11520	45	61	24832	97	95	38144	149	C9	51456	201	FD	64768	253
2E	11776	46	62	25088	98	96	38400	150	CA	51712	202	FE	65024	254
2F	12032	47	63	25344	99	97	38656	151	CB	51968	203	FF	65280	255
30	12288	48	64	25600	100	98	38912	152	CC	52224	204			
31	12544	49	65	25856	101	99	39168	153	CD	52480	205			
32	12800	50	66	26112	102	9A	39424	154	CE	52736	206			
33	13056	51	67	26368	103	9B	39680	155	CF	52992	207			

In each row the first column is the Hex code.
The second row is the Decimal equivalent multiplied by 256 for calculating the M.S.B.
The third row is for use with the L.S.B.