
SHARPSOFT
6

We thank John Trippick from Strathclyde for all the work he has put into the marvellous illustrations, which we feel help to make the User Notes come alive.

C O N T E N T S

Be sure not to miss the last page for your 1983 Subscription Form

SHARPSOFT USER NOTES

Issue No. 6

Once again, we reach the end of another year. Judging by your letters these User Notes are as popular as ever. When we started publishing them two years ago, there was little public domain information freely available for SHARP computer users. In the last two years tne SHARP range of computers has grown from one model (the MZ-80K) to four models (the MZ-80K, A, B and PC-3201). Hence, if we are to provide in these User Notes information on each machine our workload obviously increases by a factor of at least three. Our plans for the future are to include in these User Notes articles specific to each machine. For example, this issue features notes for MZ-80B owners. Next year, hopefully, we will be expanding our set articles to include material on each of the SHARP computer models. DO SEND us your contributions for publication.

More general articles will also be included. This issue presents the first in a regular series of CP/M articles, which we hope will interest all SHARP CP/M users. Our series on assembly code programming is also continued with more basic information for those of you who are just starting Z80 machine code.

Our Fig-FORTH package has generated a lot of interest. Your letters are featured, only a selection however, because the FORTH post has been enormous, with another article outlining bug fixes and FORTH programming tips.

At Christmas new games for your computer can add to the seasons fun. If you have not tried the SHARPSOFT Adventure games then you are "missing out" on a real treat. Haunted House, The Mexican Adventure and Dark Star are now available for the MZ-80K, MZ-80A and MZ-80B computers. For Chess fans we are also distributing Apollo Chess V2, which is available for the MZ-80A and MZ-80B. Write to us if you would like more details.

Progress on the tape based PASCAL compiler has been slower than we anticipated and the package will NOT be ready for release for at least a few months. We will keep you informed on our progress through later editions of these User Notes.

With Issue 6 you will find a subscription form for the 1983 User Notes. It helps us if you could return the form fully completed with your subscription as soon as possible.

Issue 7 of the User Notes will be published towards the end of February or early March 1983.

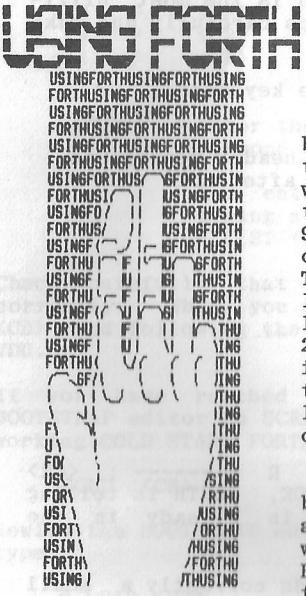
Best wishes to all our readers
for Christmas and the New Year

MIKE BRINSON

EDITOR

FORTH - Bugs, comments and programs

by Mike Brinson



A minor, but significant, bug has been found in the original FORTH EDITOR tape. Our thanks to all those readers who have written to us on the subject of the EDITOR tapes. The bug is on SCREEN 9, see page 37 Issue No. 4, Lines 12-14 of SCREEN 9 are blank and 15 contains --> The word --> should be on line 12 with lines 13 to 15 blank. When you try to load the EDITOR from Screen 7, see page 25 of Issue 4 for the definition of LOAD, interpretation of SCREEN 9 terminates at the blank line 12, hence SCREEN 10 is NOT loaded and the EDITOR does not function correctly.

To overcome this apparent insoluble bug is rather like trying to find the answer to the "chicken and egg paradox - which comes first" for without a working EDITOR how do we edit SCREEN 9 to correct the bug! The way to correct this bug is not as difficult as it seems. FORTH is an extensible language so we can use this idea to BOOTSTRAP a small subset editor into the computer, then use this editor to correct the bug on SCREEN 9. Incidentally, this was the technique I used originally to generate both the EDITOR and ASSEMBLER tapes.

This is how it's done:-

1. LOAD from tape FORTH into your computer.

2. Remove your FORTH tape from the computer and replace it with a work tape which is long enough to store at least 7 screens - a C15 or C30 tape should be OK.

   Then type in the following FORTH words

   DECIMAL <CR>

   : FORMAT 7 0 DO I P-HEADER LOOP ; <CR>

   Rewind your tape and enter

   FORMAT <CR>

   Pressing the "RECORD/PLAY" keys as requested, formats the tape (Screens 0 to 7).

   Rewind the tape and set the tape counter to zero. Fast forward the tape to betwen 60 and 62 on the counter setting. This tape is now positioned ready to record our BOOTSTRAP EDITOR on SCREEN 6.

Next we generate the BOOTSTRAP editor - which is the most difficult part of the operation. Follow the steps carefully and make sure you type each line correctly:-

1. Enter the following FORTH words from the keyboard.

```
COLD <CR>
                              -        NOTE, the header message is
DECIMAL <CR>                           displayed after COLD <CR>

HEX <CR>
: TEXT HERE C/L 1+ BLANKS WORD <CR>
  HERE PAD C/L 1+ CMOVE ; <CR>
: LINE DUP FFF0 and 17 ?ERROR SCR <CR>
    (LINE) DROP ; <CR>
: -MOVE LINE C/L CMOVE UPDATE ; <CR>
: R PAD 1+ SWAP -MOVE ; <CR>
: P 1 TEXT R ; <CR>
DECIMAL <CR>
```

NOTE, after entering the line containing : R -------- ; <CR> error message 4 will be displayed - this is OK, FORTH is telling us that the word R is NOT unique and is already in the dictionary.

If you have entered the above FORTH words correctly a small subset editor is now in memory. However, please note that these words are now part of the FORTH vocabulary and are NOT in a separate EDITOR vocabulary. The word we need is P.

2. The next step is essentially a repeat of the above sequence.

Type the follwing FORTH words.

```
6 LIST <CR>
```

Then press the "PLAY" key when requested.

SCREEN 6 will be displayed on the VDU - followed by the usual OK. The screen will of course be empty.

Next rewind your tape to between 60 and 62 on the tape counter.

Next using the word P type in the following FORTH words:-

```
0   P    (  BOOTSTRAP EDITOR) <CR>
1   P    HEX <CR>
2   P    : TEXT HERE C/L 1+ BLANKS WORD <CR>
3   P           HERE PAD C/L 1+ CMOVE ; <CR>
4   P    : LINE DUP FFF0 AND 17 ?ERROR SCR <CR>
5   P              (LINE) DROP ; <CR>
6   P    : -MOVE LINE C/L CMOVE UPDATE ; <CR>
7   P    : R PAD 1+ SWAP -MOVE ; <CR>
8   P    : P 1 TEXT R ; <CR>
9   P    DECIMAL <CR>
```

```
10   P   <CR>
11   P   <CR>
12   P   <CR>
13   P   <CR>
14   P   <CR>
15   P   <CR>
```

NOTES: After the <CR> at the end of each line FORTH will respond with OK.

The entered information can be displayed after typing a complete line by entering the FORTH words 6 LIST <CR>

Check carefully that you have entered all the FORTH words correctly. When you are sure that everything is OK type FLUSH <CR> and following the PLAY/RECORD instructions displayed on the VDU.

If you have reached this stage your tape now contains the BOOTSTRAP editor on SCREEN 6. To check that this small editor is working COLD START FORTH by entering:

```
COLD  <CR>
```

Rewind the BOOTSTRAP editor tape to roughly setting 60 to 62 and type:

```
6 LOAD <CR>
```

If everything is OK the BOOTSTRAP editor should be operational and you can use the word P. If it does not work go through the sequence again and regenerate the BOOTSTRAP tape.

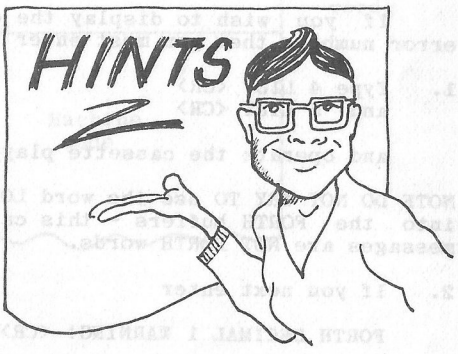
The final stage is to correct the bug on SCREEN 9 of the original EDITOR tape. The steps are as follows:

1. Put some sticky-tape on the left-hand Write Protect hole on the EDITOR tape cassette.

2. LOAD the BOOTSTRAP EDITOR (using 6 LOAD <CR>)

3. Place the EDITOR tape in the cassette player, rewind and fast forward to SCREEN 9 (roughly 90-92 on the tape counter).

4. Type DECIMAL 9 LIST <CR>. Screen 9 should be displayed on the VDU after pressing the "PLAY" key.



5. Rewind the tape top settings 90-92.

6. Type 12 P --> <CR>
   and 15 P <CR>

7. Type 9 LIST <CR>
   - the corrected Screen 9 should then be displayed on the VDU.

8. If SCREEN 9 looks OK - type FLUSH <CR> to re-record Screen 9 on the EDITOR tape.

9. Remove the sticky-tape from the EDITOR CASSETTE.

To test the editor tape - cold start FORTH, LOAD the EDITOR starting at SCREEN 7. After loading is complete, typing EDITOR <CR> should result in OK and a working live editor. You can then use this to enter the more complete editor listing given in ISSUE 5.

The FORTH cold start header

A number of readers have written to us asking if their FORTH tapes were OK because they output a different cold start header for that described in ISSUE 4. The answer is yes, they are OK. We changed the header to identify our tape version of fig-FORTH and unforunately omitted to make the change in the User instructions. Out apologies if this has caused confusion.

The FORTH ERROR messages

FORTH uses a system of error numbers or error messages to identify user errors. For example, error number 4 corresponds to error message "ISN'T UNIQUE", which is the message denoting that the last word compiled already existed in the dictionary. These error messages are recorded on SCREENS 4 and 5 of the EDITOR tape.

If you wish to display the error messages rather than the error numbers then you must enter the following sequence.

1. Type 4 LIST <CR>
   and 5 LIST <CR>

   and operate the cassette player as requested.

NOTE DO NOT TRY TO use the word LOAD to enter the error messages into the FORTH buffers - this causes errors because the error messages are NOT FORTH words.

2. If you next enter

   FORTH DECIMAL 1 WARNING! <CR>

you should then find the system displays the FORTH error messages rather than the error messages.

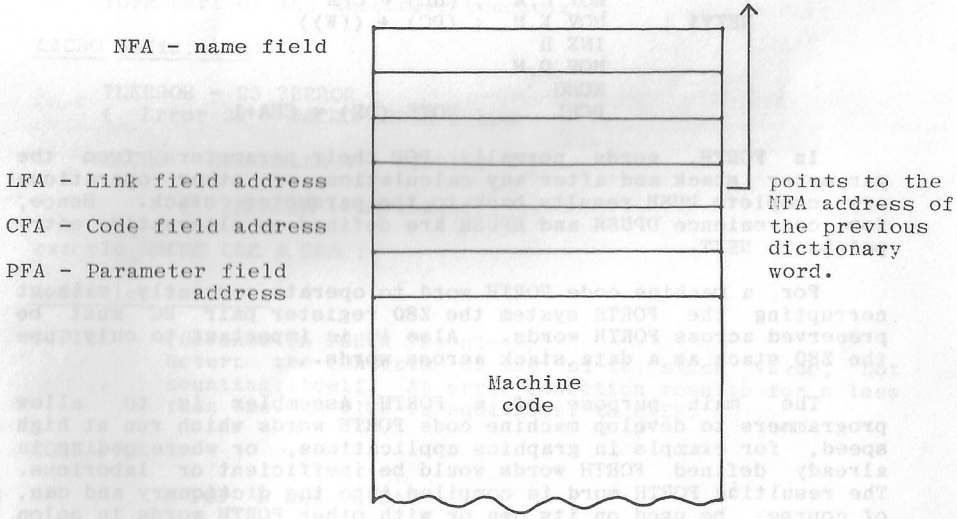
An introduction to the FORTH 8080 Assembler

In some FORTH real time applications the full speed of machine code is required. Normally FORTH systems include an assembler which allows FORTH words to be defined where the body of the word is in machine code rather than high-level FORTH definitions. Often, in the literature these machine code FORTH words are called primatives.

The standard 8080 fig-FORTH Assember will, of course, generate code to run on a Z80 micorprocessor. However, when using this assembler only words written in the 8080/Z80 overlapping instructions can be handled by the assembler.

To understand the operation of a FORTH assembler requires a basic knowledge of the inner workings of FORTH. One of the best books which covers in depth the structure of fig-FORTH is Dr. Ting's book given in the reference section of Issue 4. However, to write FORTH words using the 8080 assembler is straightforward once readers realise the structure of the primative FORTH words. Fig. 1 shows the structure of a FORTH primative which can be stored anywhere in the dictionary.

FIG. 1



In a machine code primative the code field address points to the start of the machine code which defines the operation of the word. Fig. 2 shows the code for the FORTH word +

FIG. 2

```
NFA -          DB 81H        ; +
               DB '+' 80H
LFA -          DW ZLESS-5
CFA -   PLUS:  DW $ + 2      ; (S1) ← (S1) + (S2)
PFA -          POP D
               POP H
               DAD D
               JMP HPUSH
```

When the FORTH word + is executed control passes to the PFA - the machine code following is then executed. NOTE - on completion of the FORTH word control MUST be returned to the FORTH inner interpreter. In 8080 fig-FORTH there are three entry points to the inner interpreter - these are shown in Fig 3.

FIG. 3

```
DPUSH :    PUSH D
HPUSH :    PUSH H
NEXT  :    LDAX B     ; (W) ← ((IP))
           INX  B     ; (IP) ← (IP) + 2
           MOV L,A
           LDAX B
           INX  B
           MOV H,A    ; (HL) ← CFA
NEXT1 :    MOV E,M    ; (PC) ← ((W))
           INX H
           MOV D,M
           XCHG
           PCHL       ; NOTE (DE) = CFA+1
```

In FORTH, words normally POP their parameters from the parameter stack and after any calculations or other operations are complete PUSH results back to the parameter stack. Hence, for convenience DPUSH and HPUSH are defined as alternative entry points to NEXT.

For a machine code FORTH word to operate correctly without corrupting the FORTH system the Z80 register pair BC must be preserved across FORTH words. Also it is important to only use the Z80 stack as a data stack across words.

The main purpose of a FORTH Assembler is to allow programmers to develop machine code FORTH words which run at high speed, for example in graphics applications, or where coding in already defined FORTH words would be inefficient or laborious. The resulting FORTH word is compiled into the dictionary and can, of course, be used on its own or with other FORTH words in colon ":" definitions. This statement implies that a FORTH assembler not only includes a method for representing the 8080 instructions but also includes a compiler which compiles the machine code word into the FORTH dictionary. The FORTH words CODE and C; are corresponding words the the high-level : and ; compiler words. Consider the very simple example on the next page:-

EXAMPLE:

```
CODE    MCODE.TEST

        NEXT JMP

C;
```

This sequence will form a dictionary entry called MCODE.TEST which does nothing except jump to the inner interpreter. Also notice that the assembly code is written in Reverse Polish notation.

The FORTH Assembler consists of three distinct parts:-

1. Defining words - such as CODE and C;
   These words are normally compiled into the FORTH Vocabulary from the FORTH Assembler Screen as it is loaded from tape or disc.

2. Instruction defining words - such as POP PUSH ADD AND
   The <BUILDS....DOES> construction and high-level definitions are used to form the instruction codes in an asssembler Vocabulary.

3. Structured and MACRO assembly words -
   These constructions are defined by colon definitions are form part of the Assembler vocabulary.

MACRO Definition

```
:   ?MERROR - 25 ?ERROR ;
    (  Error 25 = MACRO ERROR number)

:   MACRO ' ASSEMBLER 4 + CURRENT

        @ ?MERROR [COMPILE] : ;
```

example MACRO CLC A ORA ;

Assembler examples

1.PICK (defined in FORTH 79) n1---n2
Return the contents of the n1-th stack value, not counting itself. An error condition results for n less than one. 2 PICK is equivalent to OVER.

```
CODE PICK

        H POP   H DCX   H DAD

        SP DAD  ME MOV  H INX

        M D MOV   D PUSH

        NEXT JMP

C;
```

2. MINUS

```
CODE MINUS
        H POP   L A MOV   CMA
        A L MOV   H A MOV   CMA
        A H MOV   H INX
        HPUSH   JMP
C;
```

NOTE, HPUSH must be defined in the assembler vocabulary. -add

```
1244 CONSTANT HPUSH
```

to screen 8 line 6.

Zero level assembly in FORTH

It is possible to add machine code words to FORTH WITHOUT using the assmbler - the technique is to enter the machine code, usually in hex, directly into the dictionary - consider the following.

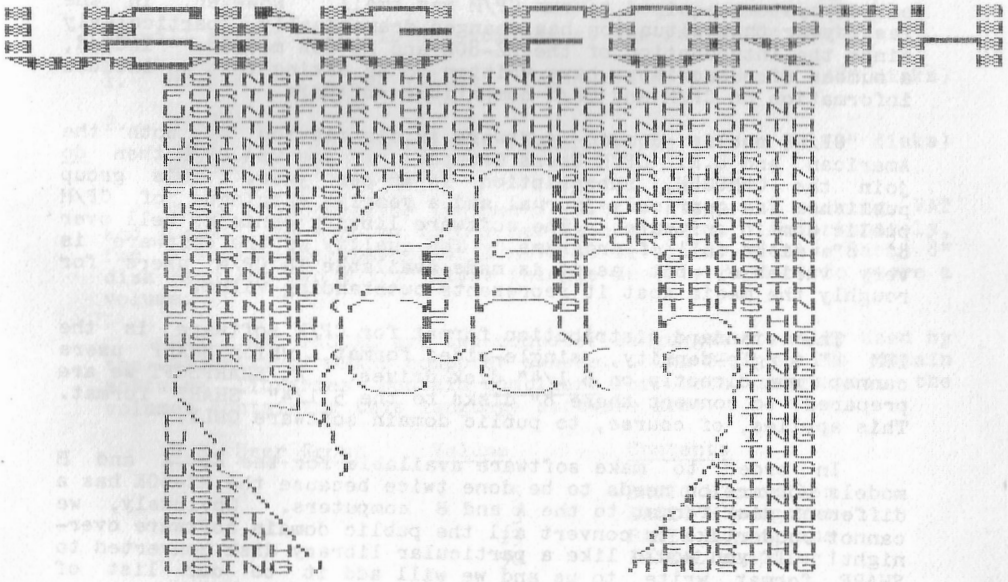
```
FORTH DEFINITIONS HEX
(Example of zero level assembly)
CREATE
SAVE/T    D9 C, CD C, 21 C, 00 C, <CR>
          CD C, 24 C, 00 C, D9 C, <CR>
          C3 C, 45 C, 12 C, SMUDGE <CR>
```

This sequence creates a word called SAVE/T which will generate a COPY of your FORTH tape. However, if you load or list any screens before using SAVE/T the cassette buffers, and parameters, used by the SHARP monitor will be destroyed and a garbage tape will result. The current state of the cassette buffers and values for read-write parameters can be displayed on the VDU using the following FORTH utility.

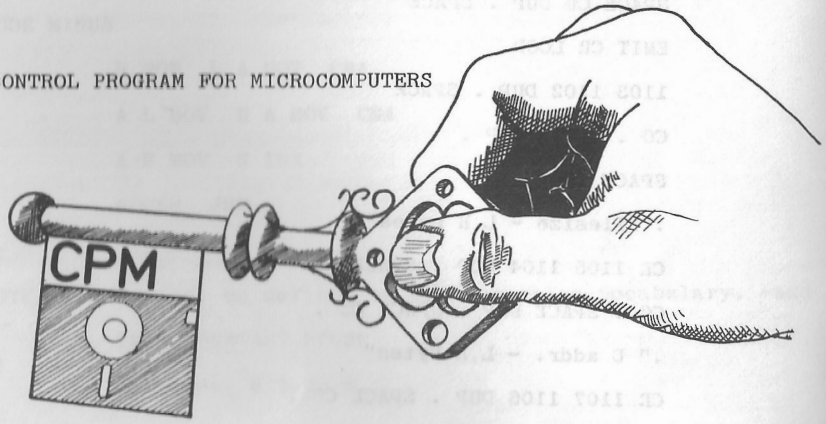
```
: HEADER/T  HEX CR
        10F0 DUP . SPACE C@ .
        ." FILE code" CR
        ." Add Byte ASCII" CR
        1101 10F1 DO I DUP
```

```
        SPACE C@ DUP . SPACE
        EMIT CR LOOP
        1103 1102 DUP . SPACE
        C@ . SPACE DUP .
        SPACE C@ .
        ."Filesize - L,H bytes"
        CR 1105 1104 DUP . SPACE
        C@ . SPACE DUP . SPACE C@ .
        ." D addr. - L,H bytes"
        CR 1107 1106 DUP . SPACE C@ .
        ." E addr - L,H bytes"
        CR
;
```

This routine is for the MZ-80K and MZ-80A. More FORTH next issue.



CONTROL PROGRAM FOR MICROCOMPUTERS



The CP/M disk operating system is available for all the current Sharp computers sold in this country. This column will be published on a regular basis, reviewing in each issue a selection of the CP/M public domain and commercial software which can be used on SHARP microcomputers.

In the past we decided not to include in the SHARPSOFT USER NOTES contributions covering CP/M because the number of disk based systems equiped to run CP/M was small. However, in the last year this situation has changed dramatically, particularly since the introduction of the MZ-80B and MZ-80A models. Indeed, a number of subscribers have written to us asking that we include information on CP/M and CP/M compatible software.

CP/M public domain software is distributed by both the American and U.K. CP/M Users Groups. If you use CP/M then do join the CPMUGUK, subscription is £6 per year. This group publishes a quarterly journal and a yearly catalogue of CP/M public domain software. The software library contains well over 80 8" disks full of programs. The quality of the software is very variable, but as it is made available to CP/M users for roughly the media cost it represents outstanding value.

The standard distribution format for CP/M software is the IBM 8" single-density, single-sided format, which SHARP users cannot run directly on 5 1/4" disk drives. At SHARPSOFT we are prepared to convert these 8" disks to the 5 1/4" SHARP format. This applies, of course, to public domain software ONLY.

In order to make software available for the K, A and B models conversion needs to be done twice because the MZ-80K has a different disk format to the A and B computers. Obviously, we cannot undertake to convert all the public domain software overnight! If you would like a particular library disk converted to SHARP format write to us and we will add it to our list of software for conversion.

Currently we can offer the following disks for SHARP CP/M users:-

1. MZ-80K

U.S. User Group	Volumes 1 to 42
	Volume 49*
	Volume 50
	Volume 51*
SIG/M	Volume 10

2. MZ-80A and MZ-80B

U.S. User Group	Volumes 3, 5, 13
	Volume 49*
	Volume 50
	Volume 51*
SIG/M	Volume 10

* These volumes should be available by the time these notes reach you - contact us for a current list of the disks we can supply in SHARP format.

A complete catalogue for the CP/M software library can be obtained from:

CP/M Users Group (UK),
11 Sun Street,
Finsbury Square,
London, EC2M 2PS
(01-247-0691)

We will supply CP/M library volumes at the following costs:-

1. MZ-80K £12 per volume (each volume is two 5 1/4" disks)

2. MZ-80A and MZ080B £6 per volume (each volume is one 5 1/4" disks)

These charges cover the media cost, postage and packing, VAT and a small copying fee to pay our labour costs. Unfortunately, for MZ-80K users charges will have to be higher because each 8" disk requires two 5 1/4" disks (single density format) to store a volume.

BASIC and PASCAL are two computer languages normally used by the majority of SHARP computer owners. The CP/M public domain software libraries contain compilers for both languages, the volumes containing this language software are:

U.S. User Group Volume	Contents
5	BASIC E compiler
3	BASIC E programs
13	BASIC E programs
50	PASCAL compiler

Catalogue descriptions for Volumes 3, 5, 13 and 50

VOLUME 3

VARIOUS BASIC E GAMES AND PROGRAMS

NUMBER	SIZE	NAME	COMMENTS
		CATALOG.3	CONTENTS OF CP/M VOLUME 3
		VOLUME3A.DOC	COMMENTS ON SOME PROGRAMS
		VOLUME3B.DOC	COMMENTS ON OTHER PROGRAMS
3.1	2K	ACE.BAS	BASIC-E PROGRAM. SEE DOC'S
3.2	4K	AMAZE.BAS	BASIC-E PROGRAM. SEE DOC'S
3.3	2K	ANIMAL.BAS	BASIC-E PROGRAM. SEE DOC'S
3.4	2K	BAGELS.BAS	BASIC-E PROGRAM. SEE DOC'S
3.5	6K	BAGELS2.BAS	BASIC-E PROGRAM. SEE DOC'S
3.6	7K	BIOPRINT.BAS	BASIC-E PROGRAM. SEE DOC'S
3.7	11K	BLKFRI.BAS	BASIC-E PROGRAM. SEE DOC'S
3.8	10K	BLKFRI2.BAS	BASIC-E PROGRAM. SEE DOC'S
3.9	6K	CANNONS.BAS	BASIC-E PROGRAM. SEE DOC'S
3.10	3K	CHASE.BAS	BASIC-E PROGRAM. SEE DOC'S
3.11	2K	CHOMP.BAS	BASIC-E PROGRAM. SEE DOC'S
3.12	1K	COMBINE.BAS	BASIC-E PROGRAM. SEE DOC'S
3.13	1K	CORE.BAS	BASIC-E PROGRAM. SEE DOC'S
3.14	1K	CORETEST.BAS	BASIC-E PROGRAM. SEE DOC'S
3.15	4K	CRAPS.BAS	BASIC-E PROGRAM. SEE DOC'S
3.16	1K	EUCLID.BAS	BASIC-E PROGRAM. SEE DOC'S
3.17	1K	FIB.BAS	BASIC-E PROGRAM. SEE DOC'S
3.18	1K	FIT.BAS	BASIC-E PROGRAM. SEE DOC'S
3.19	4K	FORMAT.BAS	BASIC-E PROGRAM. SEE DOC'S
3.20	1K	FORMAT.FMI	INSTRUCTIONS FOR FORMAT.BAS IN FORMAT CODE
3.21	5K	HANG.BAS	BASIC-E PROGRAM. SEE DOC'S
3.22	5K	HELLO.BAS	BASIC-E PROGRAM. SEE DOC'S
3.23	7K	KENO.BAS	BASIC-E PROGRAM. SEE DOC'S
3.24	5K	LANDER.BAS	BASIC-E PROGRAM. SEE DOC'S
3.25	8K	LANES.BAS	BASIC-E PROGRAM. SEE DOC'S
3.26	7K	LEM.BAS	BASIC-E PROGRAM. SEE DOC'S
3.27	3K	LOAN.BAS	BASIC-E PROGRAM. SEE DOC'S
3.28	2K	LOVE.BAS	BASIC-E GRAPHIC
3.29	2K	PLOT2.BAS	BASIC-E PROGRAM. SEE DOC'S
3.30	4K	POET.BAS	BASIC-E PROGRAM. SEE DOC'S
3.31	6K	README.FMI	ANOTHER FORMAT SOURCE WITH NOTES ON THE AUTHOR, ON ML80 AND ON SPAT
3.32	5K	S/TREK.BAS	BASIC-E PROGRAM. SEE DOC'S
3.33	4K	STARS.BAS	BASIC-E PROGRAM. SEE DOC'S
3.34	27K	STARTREK.BAS	BASIC-E PROGRAM. SEE DOC'S
3.35	7K	STORY.BAS	BASIC-E PROGRAM. SEE DOC'S
3.36	4K	STRIKE9.BAS	BASIC-E PROGRAM. SEE DOC'S
3.37	7K	TREKINST	INSTRUCTIONS FOR STARTREK.BAS
3.38	7K	TTT.BAS	BASIC-E PROGRAM. SEE DOC'S
3.39	9K	WUMPUS.BAS	BASIC-E PROGRAM. SEE DOC'S

VOLUME 5

BASIC-E COMPILERS AND INTERPRETERS
BASIC-E PROGRAMS, CONTINUED FROM VOLUME 3
MICROSOFT BASIC PROGRAMS

NUMBER	SIZE	NAME	COMMENTS
		CATALOG.5	CONTENTS OF CP/M GROUP VOL 5
		VOLUME5.DOC	COMMENTS
5.1	8K	21.ASC	MICROSOFT BASIC PROGRAM
5.2	12K	BAS2-0.COM	BASIC-E COMPILER
5.3	12K	BAS2-1.COM	CASIC-E COMPILER
5.4			DELETED
5.5	5K	BIO-FF.ASC	MICROSOFT BASIC PROGRAM
5.6	4K	BIORYTH.ASC	MICROSOFT BASIC PROGRAM
5.7	10K	BLKFRI2.ASC	MICROSOFT BASIC PROGRAM
5.8			DELETED
5.9	2K	DECISION.ASC	MICROSOFT BASIC PROGRAM
5.10			DELETED
5.11	6K	EDTEXT.ASC	MICROSOFT BASIC PROGRAM
5.12	2K	FORMAT.ASC	MICROSOFT BASIC PROGRAM
5.13			DELETED
5.14	16K	OTHELLO.BAS	BASIC-E PROGRAM
5.15	5K	OTHELLO.DOC	INSTRUCTIONS FOR OTHELLO.BAS
5.16	2K	RADIX.ASC	MICROSOFT BASIC PROGRAM
5.17	1K	RECOVERY.ASC	MICROSOFT BASIC PROGRAM
5.18	12K	RUN2-2.COM	BASIC-E INTERPRETER
5.19	12K	RUN2-3.COM	BASIC-E INTERPRETER
5.20	12K	RUNK2-0.COM	BASIC-E INTERPRETER
5.21	4K	SLOT.ASC	MICROSOFT BASIC PROGRAM
5.22	2K	SORT.ASC	MICROSOFT BASIC PROGRAM
5.23	7K	STARTREK.ASC	MICROSOFT BASIC PROGRAM
5.24	14K	SUPTRK3.ASC	MICROSOFT BASIC PROGRAM
5.25			DELETED

VOLUME 13

BASIC-E/CBASIC AND MICROSOFT BASIC PROGRAMS

A VERY MIXED BUNCH. ZOSO LAMENTS ABOUT THE QUALITY IN HIS CHRISTMAS PLAY, COMPLETE WITH GREEK CHORUS. THIS DISKETTE IS A MUST FOR ANYONE WHO MUST HAVE A COPY OF EVERYTHING. UNFORTUNATELY A FEW JEWELS HERE MAY BE MISSED BECAUSE OF THE COMPANY IT KEEPS IN THIS VOLUME

THE *.ASC ARE MICROSOFT, THE *.BAS WILL (SOMETIMES) COMPILE ON BASIC-E/CBASIC. MANY ARE COMPATIBLE. ALL CAN BE SWITCHED WITH SYNTACTICAL SYNCHRONIZATION (WHO SAID THAT??) (MY LIPS DIDN'T MOVE)

NUMBER	SIZE	NAME	COMMENTS
		CATALOG.13	CONTENTS OF CP/M VOLUME 13
13.1	7K	15/PUZ.ASC	PROGRAM IN MICROSOFT BASIC
13.2	7K	1500.ASC	PROGRAM IN MICROSOFT BASIC
13.3	2K	23MATCH.BAS	PROGRAM IN BASIC-E/CBASIC
13.4	3K	BAGELS.BAS	PROGRAM IN BASIC-E/CBASIC
13.5	4K	BIORYME.ASC	PROGRAM IN MICROSOFT BASIC
13.6	6K	BLACKJAC.BAS	PROGRAM IN BASIC-E/CBASIC
13.7	1K	BULLSEYE.BAS	PROGRAM IN BASIC-E/CBASIC
13.8	6K	CHECKERS.BAS	PROGRAM IN BASIC-E/CBASIC
13.9	2K	CHIEF.BAS	PROGRAM IN BASIC-E/CBASIC
13.10	2K	CONVERT.BAS	PROGRAM IN BASIC-E/CBASIC
13.11	7K	DICE.BAS	PROGRAM IN BASIC-E/CBASIC
13.12	7K	KINGDOM.BAS	PROGRAM IN BASIC-E/CBASIC
13.13			DELETED
13.14	15K	NFL.BAS	PROGRAM IN BASIC-E/CBASIC
13.15	4K	ROCKET.BAS	PROGRAM IN BASIC-E/CBASIC
13.16	1K	RUSSIAN.BAS	PROGRAM IN BASIC-E/CBASIC
13.17	16K	SWARMS.BAS	PROGRAM IN BASIC-E/CBASIC
13.18	16K	SWARMS2.ASC	PROGRAM IN MICROSOFT BASIC
13.19	2K	TRAP.BAS	PROGRAM IN BASIC-E/CBASIC
13.20	6K	WUMPAS.BAS	PROGRAM IN BASIC-E/CBASIC
13.21	23K	ZOSO.2	WHEN THE REVIEW IS ABOUT TWICE AS LONG AS ANY OF THE PROGRAMS, I GUESS THE REVIEWER HAS SOMETHING ON HER/HIS MIND

CP/MUG: VOLUME 50

DESCRIPTION: Bob Van Valzah's "Pascal Pascal Compiler" & a few miscellaneous programs for printing via UNIX.

NUMBER	SIZE	NAME	COMMENTS
		-CATALOG.050	CONTENTS OF CP/M VOL. 50
		UGFORM.LIB	SUBMITTAL FORM
50.1	1K	A.OCO	Sample program, eight queens problem, ready to be linked to RTP.COM (OCO=obj code)
50.2	1K	A.PCO	Sample program P-code, ready for PFET.COM
50.3	5K	ABSTRACT.050	Abstracts for this volume.
50.4	2K	COMPARE.COM	Vital program for compiler writers.
50.5	2K	CPMDIR.C	Prints CP/M DIR on UNIX.
50.6	2K	CRCK.COM	CRC program for validating files on this disk.
50.7	2K	CRCKLIST.CRC	CRC list of files on this disk.
50.8	3K	DISK.DOC	Bob's comments.
50.9	3K	EQ.COM	Prints all solutions to the eight queens problem.
50.10	2K	EQ.PAS	
50.11	5K	FROMCPM.C	Print CP/M file via UNIX stdout.
50.12	1K	FWD.PAS	Forward Procedure Declarations.
50.13	6K	HW5.COM	Builds optimal binary search tree & decodes a message.
50.14	15K	HW5.PAS	
50.15	1K	HW5DATA.	Sample data for above program.
50.16	6K	PASYNTAX.DOC	Syntax graphs for PPC compiler.
50.17	1K	PC.SUB	Submit file to compile a PPC program.

50.18	7K	PFET.COM	Object code of Pcode to 8080 translator.
50.19	11K	PFET.PAS	Source of above.
50.20	1K	PHONE.C	C program to print words you can spell with your phone #.
50.21	1K	PLAYDATA.	Sample data for above program.
50.22	13K	PLAYKAL.PAS	PPC program to determine best moves in game of Kalah.
50.23	2K	POPS.DOC	DOC for Pcodes used by PPC.
50.24	1K	POWTWO.PAS	PPC Program to print negative powers of two.
50.25	16K	PPC.COM	Pascal Pascal Compiler.
50.26	10K	PPC.DOC	DOC for above file.
50.27	26K	PPC.PAS	Pascal source for above file.
50.28	2K	PSTACK.DOC	DOC on run-time P-machine stack.
50.29	3K	REGEN.DOC	Notes on how to modify & compile PPC.PAS.
50.30	11K	RTP.ASM	Source for run-time package.
50.31	2K	RTP.COM	Object of above.
50.32	1K	STIRLING.PAS	Prints a table of Stirling Numbers.
50.33	4K	TESTER.PAS	Tests functionality of PPC.
50.34	1K	VALIDATE.SUB	Submit file to make sure you have a "fertile" computer.

BASIC-E was developed by Gordon Eubanks while he was a postgraduate student at the Naval Postgraduate School, Monterey, California, U.S.A. This BASIC was designed to be compatible with, whenever possible, the proposed ANSI standard BASIC and to run under CP/M. In order to allow BASIC-E to operate in a restricted memory size the implementation of the language was designed in two parts, a compiler that generates an intermediate code for a stack machine, and a "run-time" monitor (interpreter) which runs the intermediate code. However, BASIC-E is not a true compiler in that it does not generate 8080 or Z80 machine code. Its popularity among CP/M users has grown because its free!, and is robust and very accurate. BASIC-E's high numerical accuracy causes programs to run slowly - but if you require high accuracy on an 8-bit micro without floating point hardware this will always occur.

Other features include multidimensional arrays for numbers and strings, logical operators, several string handling routines and both sequential and random access for disk files.

BASIC-E has the following commands and operations:-

ABS ASC ATN CHR$ CLOSE COS COSH DATA DIM END FILE FOR
PRE FN GOSUB GOTO IFEND IF INPUT INT LEFT$ LEN LET LOG
MID$ ON NEXT OUT PRINT POST RANDOMISE READ REM RESTORE
RETURN RIGHT$ RND SGN SIN SINH STR$ SQR TAB STOP TAN
VAL,

and the following expressions:-

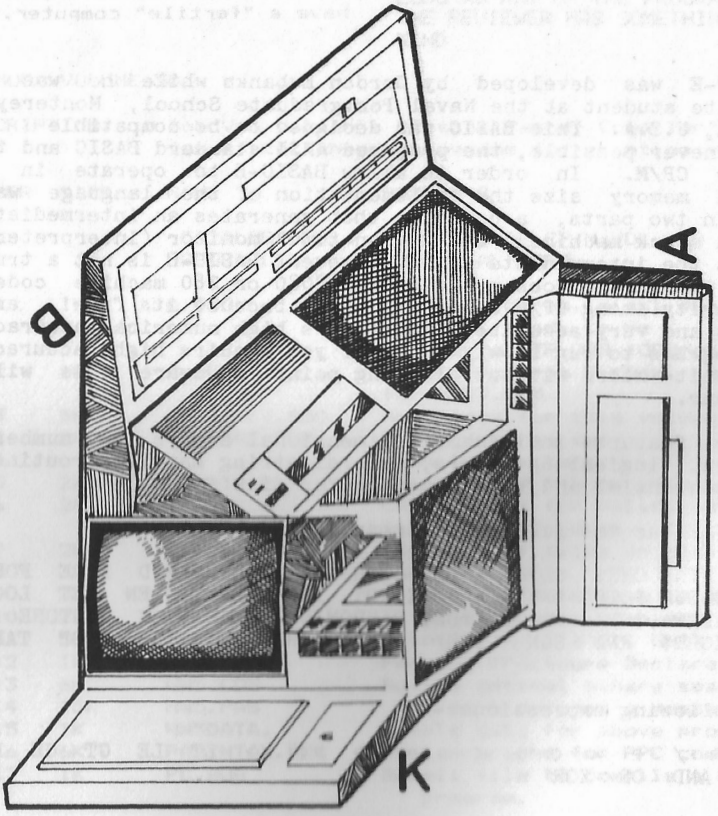
( ) ^ * / + - < > <= >= => =< <> LT LE GT GE EQ
NE NOT AND OR XOR

BASIC-E also allows you to miss line numbers out if the line is unreferenced from elsewhere. Also line numbers can be in any order you like and can be real rather than integer. Variables can be long names such as THIS.IS.A.VARIABLE which makes programs much more readable. In general BASIC-E is a fairly standard BASIC allowing most BASIC programs from other interpreters to run under BASIC-E without drastic alteration.

Following the success of BASIC-E Gordon Eubanks formed Compiler Systems to market an updated version of BASIC-E which developed into the ever popular CBASIC.

CBASIC is a commercial software product widely used in business programming or in other applications which require high precision arithmetic (real numbers range from 1.0E-64 to 9.99999999999999E62). The current cost of CBASIC for the MZ-80K, A and B and PC 3201 is roughly £65.

In the late summer of 1981 Compiler Systems (now part of Digital Research Ltd) marketed for the first time a true compiler version of CBASIC which they call CB80.



CB80 contains a number of important enhancements over CBASIC, for example, it has the ability to accept alphanumeric labels as well as the usual numeric labels and can CALL statements that pass parameters to multiple line functions. Also multiple line functions can contain local variables. As CB80 programs are compiled to 8080 machine code rather than interpreted, the code runs much faster than that generated by CBASIC or BASIC-E. For example, strings are eight times faster, integer arithmetic and looping is roughly a hundred times faster, and real arithmetic is twice as fast. Overall, most programs will run between five and ten times faster. CB80 represents the latest state-or-the-art development in software technology for CP/M based computers. This product is available for the MZ-80B and PC 3201 and costs £298 for a single user licence.

If you are a "software hack" who enjoys attacking the "inner workings" of programs then the PASCAL compiler in Volume 50 of the U.S. CP/M user group will interest you. This Volume contains, Bob Van Valzah's 'PASCAL PASCAL compiler' which is a true machine code compiler for a subset of PASCAL. Written by a group of American students, as a class exercise!, under the supervision of Bob Van Valzah. This Volume is particularly interesting because it contains the source code for the compiler with instructions for its extension to include additional PASCAL constructions. A number of simple PASCAL programs are also provided and full documentation is also given on the disk. The following listings show two of the example PASCAL programs.

```
(* decimal representaton of negative powers of 2 *)

const n=10;

type digit  = 0..9;
     digtyp = array[1..n] of digit;

var i,k,r: integer;
    d : digtyp;

begin for k:=1 to n do
   begin put#0('.'); r:=0;
      for i:=1 to k-1 do
      begin r:=10*r+d[i]; d[i]:=r/2;
            r:=r-2*d[i]; put#0(d[i]+'0')
      end;
      d[k]:=5; put#0('5',13,10)
   end
end.
```

```
(*
        Prints solutions to the problem of placing eight queens on
        a chess board in such a way that no queen checks against
        any other queen.  See "Algorithms+Data Structures = Programs",
        Niklaus Wirth.
*)

type
   boolean = (false, true);
   aryi    = array[0.. 8] of integer;
   aryb    = array[0..16] of boolean;

var
   i : integer;
   a, b, c: aryb;
   x       : aryi;

procedure print;

   var
      k : integer;

   begin
   for k:=1 to 8 do put#0(x[k]#,' ')"
   put#0(13,10)
   end; (* procedure print *)

procedure try(i : integer);

   var
      j : integer;

   begin
   for j:=1 to 8 do
      if (a[j]=true) and (b[i+j]=true) and (c[i-j+7]=true) then
         begin
         x[i]:=j;
         a[j]:=false; b[i+j]:=false; c[i-j+7]:=false;
         if i<8 then try(i+1) else print;
         a[j]:=true; b[i+j]:=true; c[i-j+7]:=true
         end
   end; (* procedure try *)

begin (* main line *)
for i:= 1   to 8  do a[i]   :=true;
for i:= 2   to 16 do b[i]   :=true;
for i:= 0-7 to 7  do c[i+7] :=true;

try(1)
end.
```

The develpment of CP/M for the MZ-80K was an innovative step undertaken by Crystal Electronics Ltd. A full account of its development was published in the December 1980 Liverpool Software Gazette. Mr. A.J. Cornish of Crystal Electronics has given us permission to reprint his article. We hope our readers find this article as instructive as we did when first reading it.

In our next issue, notes on the CP/M user library ALGOL/M compiler and the STAGE2 macro processor.

* * * * * * *

An implementation of CP/M on the SHARP MZ-80K.

by A.J. Cornish BSc.
(Crystal Electronics Ltd.)

The SHARP MZ-80K is arguably one of the most versatile micro-computers on the market at the present time, being both simple and inexpensive, while at the same time incorporating a number of useful hardware features. For those who are unfamiliar, these include the following:

Z80	CPU
8255	I/O chip which handles cassette, keyboard, VDU/software synchronisation and upper/lower case indicator.
8253	Timer chip for real-time clock and musical tones.
2716	EPROM containing character set for VDU, including pixel graphics and lower-case letters.
4K Monitor	ROM.

Scanned keyboard with all keys software-definable. VDU 40x25 characters, memory-mapped. Cassette - Normal audio type, with numeric indicator, speed approx. 1200 baud, software controlled.

External to the main machine, we have: Floppy discs (two, expandable to four), double-sided, single-density, 140K per disc, 70 tracks, 16 sectors per track, 128 bytes per sector, 5 1/4" disks.

Floppy disc controller - TOSHIBA T3444M.
Printer - 80 column EPSON type, tractor-fed, parallel interface. The floppy discs and printer interface to the main machine via the MZ80I/O interface box.

Having already developed our Xtal BASIC on the MZ-80K (see LSG No.5), we decided to attempt an implementation of the CP/M Disc Operating Syste, in order to allow the MZ-80K to use the wide variety of languages and software available to other machines already running under CP/M. Apart from that very important advantage, CP/M also offers a system which is itself totally independent of any one high-level language, whereas the operating system normally offered with the MZ-80K is almost entirely centred around SHARP DISC BASIC.

We decided to set ourselves the following goals in our implementation:

i) The implementation should be as near to standard as possible, while hopefully including some of the features peculiar to the MZ-80K. At the same time, it must be possible to run all the existing facilities and software of the MZ-80K.

ii) The disk operations should be speeded up if at all possible. It had already been noted that, although the SHARP DISC BASIC booted impressively quickly, file accesses for loading of programs and data were exceedingly slow.

iii) All of the SHARP MZ-80 peripherals should be supported (cassette recorder, printer, etc.)

iv) It should be reasonably easy to transfer software from another machine already running CP/M onto the MZ-80K. This is important for software houses, dealers and programmers with other computers running software which they might consider offering on the MZ-80K.

HARDWARE MODIFICATION

Figure 1 shows the memory map of the MZ-80K as it is in normal use, and, next to it, we see the map of the system required while running CP/M. The main problem is immediately apparent: How do we put 4K of RAM in the area from 0000-0FFFH to replace the ROM that site there? It was clear to us that a hardware modification would be necessary, but it soon became clear that there were many different solutions to the problem - all expensive!

Just as despair was beginning to set in, my colleague, Trevor Brownen (sometimes referred to in our establishment as "The Boss"!), had an inspiration, and the problem was solved within a few days. Briefly, the solution adopted makes use of the very flexible address decoding of the MZ-80K, so that the 4K blocks of memory 0000-0FFH and C000-CFFFH (the top 4K of RAM space) could be interchanged by addressing one of the unused memory-mapped ports. The Monitor ROM and the RAM can be restored to their norma positions by addressing another port. Hence the mahcine can be operated with all SHARP BASIC software etc., as if no modification had been made. Of course, all of this means that the CBIOS (Customised Basic I/I System, the part of the CP/M system which must be written by the implementor) must also contain the necessary monitor routines for scanning the keyboard, driving the VDU, etc. This means that we actually have a 46K CP/M system, even though we have 48K RAM.

The hardware modification involves the removal of an IC on the main board of the MZ-80K replacing it with a socket, and then plugging in a PCB holding three IC'S (size about 3 inches by 2). Finally, two wires must be soldered to complete the task. (SHARP (UK) Ltd. have kindly allowed this modification to be done without invalidating the guarantee, so long as it is performed by a SHARP dealer (but any iuser who is in doubt should contact his or her dealer).

KEYBOARD AND CHARACTER SET

Another problem is in the character set employed on the MZ-80K. Although this uses standard ASCII codes for the numerals and upper-case letters, it jumbles up the codes for the lower-case letters. As a matter off fact, the character generator chips does not use ASCII at all, but a special set of 'display codes', which are generated by means of a look-up table in the Monitor. This problem was solved by writing a different look-up table in the CBIOS (thus removing the need to re-program the character generator!). Since the MZ-80 P3 printer also uses the incorrect ASCII codes, we have also made use of a table within the Monitor ROM, which converts 'display codes' into the "SHARP ASCII" codes. Hence, by applying these two look-up tables one after the other, we convert the standard ASCII code to drive the printer. This again removes the need to reprogram the character generator.

While doing all of this, we decided to take the opportunity to redefine the keyboard and VDU fucntions. In particular, the MX-80K does not have a control key, whereas nearly all terminals do, and CP/M requires it. After a little thought, we redefined the "BREAK" key as "CTRL". The "SML/CAP" key has been changed to allow a "toggle" effect (i.e., Hit once to go to lower case, and again to go back to upper case). Normally, we have to hit "shift-SML/CAP" to go to lower case, but this combination is now redefined as an "ESCAPE" (another key used frequently in CP/M software). Apart from these changes, the keyboard behaves exactly as shown on the keytops, and the graphics characters may be used in programs, if desired.

A full list of control-characters is shown at Table 1, together with their effects if printed from within programs. These are in addition to those used in the CP/M line editor (e.g. CTRL-P turns the printer on and off when typed at the keyboard). Note the convention of using "^" an abbreviation for "CTRL".

DISC ACCESS SPEED

A study of the disc operating routines in the SHARP DISC BASIC revealed that the software was much more geared to loading or saving many sectors at once, rather than with one sector at a time, as is required under CP/M. The operating system boots very quickly, since that part of the disc is never changed and can be loaded as one large block. With programs, however, a sector does not necessarily set next to the previous one read from or written to. Hence each sector must be read in separately and, since the gap between sectors is small compared with the time required to set up the floppy disc controller for the next sector, the disc has to spin for a complete revolution before the next sector is loaded - a 16 to 1 reduction in speed!

An appreciable increase in speed is achieved by means of INTERLACING, so that, for example, sector 2 of a track actually appears 5 sectors after sector 1. Although this is still not as fast as it COULD be, it is three times faster than it was, and this is roughly the improvement offered by Xtal CP/M over SHARP DISC BASIC.

Since each sector has an indentification mark for the controller, it is a simple matter to achieve interlacing by means of the FORMAT utility program. This program writes the sector identifications as the first stage of initialisation and we can define the order in which the sectors appear on the disc by means of a table within the program. In fact, we format the discs with 1 to 1 interlacing for the first five tracks (those containing the operating system), and 5 to 1 interlacing for the directory track and file tracks, thus getting both fast booting and reasonably fast file transfer.

PERIPHERAL DEVICES

In order to get the most flexibility in Input/Output facilities, the IOBYTE feature of CP/M has been implemented in full, and software included drive the MZ-80 P3 printer as the LPT: device (the default LST: or listing device), and up to two serial interface cards (one of which is the TTY: device). The Xtal Serial Card has been develped primarily for us with CP/M and is, as far as we know, the first bi-directional serial interface to be made available for the MZ-80K which may be plugged into the MZ-80I/O Interface Box. It offers link-selectable baud rates from 150-2400 baud (adjustable from 110-9600 baud) and is an RS232-compatible interface, meaning that it will interface nearly all standard terminals and serial printers. In particular, it is an absolute necessity for the transfer of software between the MZ-80K and another machine, and thus fulfills the fourth of our goals.

The cassette recorder may be operated by means of a program on the CP/M disc called TAPE.COM. This allows us to load or save programs betwen disc and tape.

PROGRAMS ON DISC

All of the usual CP/M utilities are supplied on the Master discette (and it IS possible to copy it!!), except for MOVCPM.COM, the program used for relocating CP/M systems for different RAM sizes. It was decided to work with just 48K, since memory prices are not much of a problem these days, and it also simplifies the effort of determining whether or not a particular item of software will work with this or that memory size!

In addition, the following programs are supplied on the disc:

EXIT.COM — exits from CP/M right back to the SHARP Monitor.

MUSIC.COM — allows the user to type tunes into a file under ED, or one of the many CP/M text editors now available. By typing MUSIC <filename>, this text file is loaded and played as a tune, the format being very similar to that used in SHARP BASIC. An example tune is supplied on the disc.

TIME.COM — reads the real-time clock and displays the time. If a time is typed in as part of the command, that time is set as the current time.

These two programs are included to illustrate the use of the extra BIOS routines for driving the clock and tone GENERATOR.

AUTOGEN.COM allows the user to generate a "turn-key" disc, i.e. one which runs straight into a particular program that he/she may wish to run whenever the sytem boots. Although this can be done without AUTOGEN, it would require a knowledge of machine code and ASCII codes, so this program is a very useful one.

The standard CP/M utilities supplied on disc are:-

ED.COM	Context editor.
DDT.COM	Dynamic Debugging Tool.
STAT.COM	Status of files/devices.
PIP.COM	Transfer files, etc.
SUBMIT.COM	Batch processing command.
FORMAT.COM	Initialise discs.
ASM.COM	8080 Assembler.
DUMP.COM	Display file.
LOAD.COM	Load.HEX file.
SYSGEN.COM	Generate system.
XSUB.COM	Used with SUBMIT.
COPY.COM	Copy discs.

SOFTWARE DEVELOPMENT

Finally, having described some of the "workings" of the system, we come to that most important of subjects - what can we do with it all? The answer should be quite straightforward - all of the software that runs on a CP/M system of 46K or smaller will run very happily on the MZ-80K, and will only require modification to take account of the VDU size and printer width (the MZ-80K VDU has 40 columns instead of the more usual 64 or 80). Even the Printer presents no great problem, since the Xtal Serial Card allows the use of many types of printer (you COULD also use a terminal to replace the VDU for some software!)

Software transfer is best effected by means of the Serial Card since it is unlikely that the discs will load or dump discettes from a mchine other than another MZ-80K. A few software houses in this country have already shown interest in moving their programs onto the MZ-80K, and we should welcome enquiries from anyone who has developed CP/M software.

For our own part, we shall soon be producing a CP/M version of the now highly popular Xtal BASIC 2.2 - primarily for the MZ-80K, but also for Z80 CP/M systems in general. We already have the well-known CBASIC compiler adapted for the MZ-80K, and are obtaining licences for many other languages, including ALGOL, FORTRAN, PASCAL, C, and COBOL.

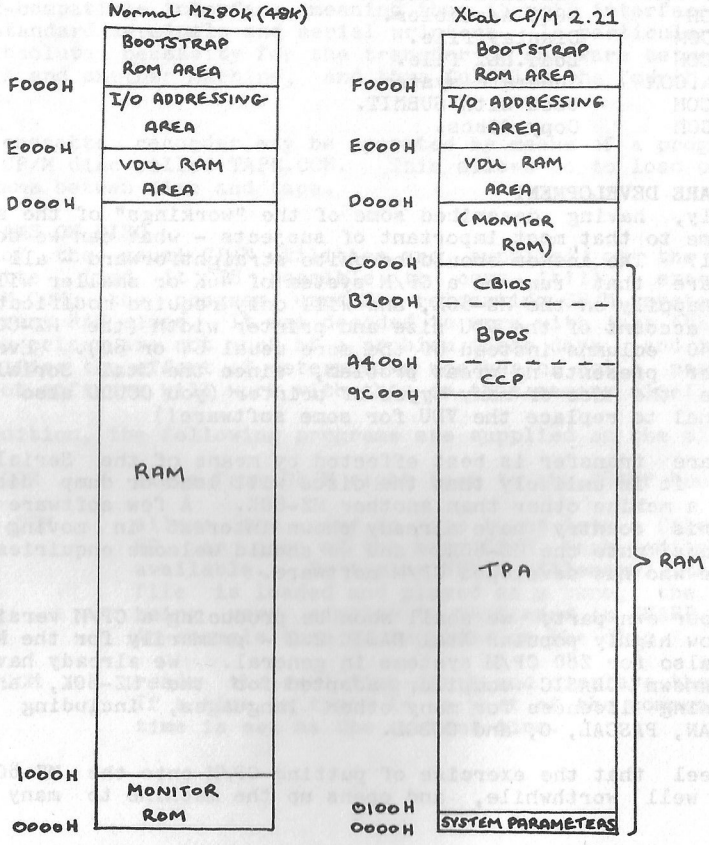
We feel that the exercise of putting CP/M onto the MZ-80K has been well worthwhile, and opens up the machine to many other uses.

TABLE 1

List of control codes available under CP/M on MZ-80K

→ or	^F(06H) - Cursor right.
	G(07H) - Ring bell.
← or	^H(08H) - Cursor left.
↓ or	^J(0AH) - Cursor down.
↑ or	^K(0BH) - Cursor up.
CLR or	^L(0CH) - Clear screen, or Form fee on printer.
CR or	^M(0DH) - Carriage Return without Line Feed.
	N(0EH) - Clear to end of line.
	^O(0FH) - Clear to end of screen.
	^Q(11H) - Line-to-line space reduction on printer.
	^R(12H) - Double-size characters on printer.
	^X(18H) - Cancel special modes on printer.
HOME	(1EH) - Home cursor on screen, or Form-feed on printer.
INST	(1FH) - Insert character into 40 character line.
DEL	(7FH) - Delete character from 40 characters line.

Figure 1 -- Memory maps



Z80 Assembly Code Programming using the SHARP "systems" and FDOS programs

Recently we received from SHARP UK our first copies of the new tape based "systems" program for the MZ-80B. Although this assembly programming package is more expensive than the MZ-80K or A versions (roughly £75 against £35) it still represents very good value for money. Four tapes and two large manuals comprise the package. A number of important enhancements have been included by SHARP software developers, for example a tape based PROM formatter program, similar to that in the disk FDOS package, is included. However, for the home computer user the inclusion of conditional assembly directives and assembly MACROS is probably the more important enhancement. The standard IF....ENDIF and MACRO....ENDM formats have been adopted by SHARP. MACROS may also have a maximum of seven parameters.

The overall package is similar to the earlier versions for the MZ-80K and A. Hence, anyone who has use the systems programs on the MZ-80K should have no difficulty in using the MZ-80B version.

The MZ-80B "systems" program manual is a significant improvement over the original MZ-80K manual. It includes a complete listing of the Monitor used by the MZ-80B Systems programs and a detailed specification of the most important monitor subroutines.

From our point-of-view at SHARPSOFT we now have a serious problem because all the SHARP computers have different monitors. There are significant differences between the original MZ-80K monitor and the new monitor in the MZ-80A. The majority of the subroutines are common to both these computers but often their memory addresses in ROM are different. Also each of the SHARP MZ-80B programs (ie BASIC, PASCAL and the systems programs) has its own monitor. In general, the important subroutines, for example console input/output routines, are contained within the monitors but their start addresses are often different. However, the names of the majority of the monitor routines are the same. If users adopt the principle of using the machine code pseudo directive EQU at the top of their assembly language programs, then all that needs to be changed for a machine code program to run on say the K, A or B is the start address, in Monitor, of each named subroutine.

Beginners Z80 Assembly language programming notes

Data transfer is a fundamental operation on any microprocessor. The Z80 uses the instruction LD, short for load, to denote a data move. Data can be transferred in units of 8 bits or 16 bits. A LD instruction must include two operands. The first operand represents the location where the data is to be stored, a register or memory location. This is often called the destination operand. The second operand represents the original data location, again either a register of memory location. This operand is called the source operand.

For instance:

```
LD    A,B
```

indicates that the data stored in register B (the source) will be transferred to register A (the destination). An extension of this notation is used to represent indirect addressing, for example consider the following

```
LD    A,(HL)
```

The brackets, ( ), are used to signify that the CONTENTS of the memory location pointed to by the address held in the HL register pair is to be moved, in this case to the A register.

Also

```
LD    (HL),A
```

denotes that the contents of the A register are to be moved to the memory address pointed to by HL. Other typical move instructions are:

```
LD      A,B        ;   8-bit     register - register
LD      HL,BC      ;  16-bit
LD      A,(HL)     ;   8-bit     register - memory
POP     AF         ;  16-bit
LD      A,25H      ;   8-bit     register - immediate data
LD    HL,0FFFH     ;  16-bit
LD    (HL),A       ;   8-bit     memory - register
PUSH    BC         ;  16-bit
LDD                ;             memory - memory
LDIR
LD    (HL),55H     ;             memory - immediate data
```

The following short assembly code fragment demonstrates the use of data move instructions to set the microprocessor registers, loading A with 0, B with 1, C with 2 etc.

```
LD    A, 0H
LD    B, 1H
LD    C, 2H
LD    D, 3H
LD    E, 4H
```

The Z80 instruction set includes 8-bit arithmetic and logical operations. These instructions are performed in register A (the accumulator). Reegisters A, B, C, D, E, H and L can also be used as an operand. Here are some examples, with their meaning:

```
ADD A       ;   the data in A is added to itself, doubling the
                data held in A. NOTE this is equivalent to shift-
                ing the contents of A left by one bit.

ADC B       ;   the data in register B is added to register A with
                the carry flag also added.

SUB C       ;   subtract the data in register C from register A.

SBC (HL)    ;   subtract the data in the memory location pointed
                to by HL from register A (with carry).

AND D       ;   logical AND of register D and register A

OR 0FH      ;   logical OR of the data 0FH and register A.

XOR A       ;   exclusive OR of register A and itself - this sets
                A=0, often used to zero the A register.

INC H       ;   increment the H register by 1, H  - H + 1.

INC (IX)    ;   increment the contents of the memory location
                pointed to by the index register IX, by 1

DEC C       ;   decrement the contents of register C by 1,
                C ← C - 1.

DEC (IY+3);     decrement the memory location pointed to by IY+3,
                by 1.
```

Following an arithmetic or logical operation, the result is normally stored in the A register and the microprocessor STATUS flags (F register) are changed to reflect the result.

Carry flag

This flag stores the carry from the highest order bit of the accumulator register A. It is set to 1 after an unsigned addition when the result is larger than an 8-bit number. It may also be set if a borrow is generated during a subtraction instruction. The carry flag is often used as a condition for a jump, call a return instruction.

Overflow/parity flag

This flag is set if overflow occurs during a signed two's compliment arithmetic operation, for example if the resulting number in register A is greater than +127 or less than -128. During a Z80 logical operation this flag is set if the parity of the 8-bit result in the accumulator is even.

Zero flag

If register A is zero following a logical or arithmetic operation the zero flag is set to 1. This flag is normally used in a conditional test for branch instructions and looping.

Sign flag

If the leftmost, bit 7, of register A is 1 after a logical or arithmetic operation, the number held in the A register is interpreted by the Z80 to be a negative number and the sign flag is set to 1.

The remaining flags are normally used in binary coded decimal arithmetic (BCD).

Unfortunately, the Z80 instruction set does not include any multiply or divide instructions. The following longer Z80 assembly code subroutines perform examples of these operations. NOTE, when writing assembly code it is important to document your program. Good documentation allows your program to be easily understood by other Z80 programmers and yourself at a later date!

If you have written any useful Z80 routines send us a copy, with documentation please, and we will publish them in a future issue.

```
; **** BINARY MULTIPLY SUBROUTINE ****
; ENTRY
;       MULTIPLYER    IN E
;       MULTIPLICAND IN A
;
; EXIT
;       PRODUCT       IN HL
; REGISTERS USED .. B, D, HL
;
;
MULT1: LD B,8 ; SET BYTE COUNTER TO 8
LD D,0
LD H,D
LD L,D ; CLEAR D, HL REGISTERS
LOOP:ADD HL, HL ; SHIFT HL LEFT
RLCA;ROTATE BIT 7 OF A INTO CARRY FLAG
JR NC,NADD ; TEST CARRY FLAG
ADD HL,DE ; ADD DE TO HL
NADD:DJNZ LOOP ; END?
RET

; **** 16 BIT DIVISION ROUTINE ****
; ENTRY
;    DIVIDEND IN DE
;    DIVISOR  IN BC
; EXIT
;    RESULT IN HL
;    REEMAINDER IN DE
: REGISTERS USED...AF, DE, HL

DIV16:XOR A ; CLEAR CARRY FLAG
```

```
LD H,A
LD L,A ; HL=0
LD A,16; A=16, LOOP COUNTER
DV0:RL E ; SHIFT LEFT, STORE PARTIAL RESULT
;          IN BIT 0
RLD
ADC HL,HL ; ROTATE HL LEFT
; IF HL > BC ; HL = HL - BC
JR NC,DV1
ADD HL,BC ; IF NEGATIVE, RESTORE HL
DV1:CCF ; PARTIAL RESULT IN CARRY FLAG
DEC A
JR NZ,DV0
EX DE,HL
ADC HL,HL ; STORE LAST BIT OF RESULT
RET
```

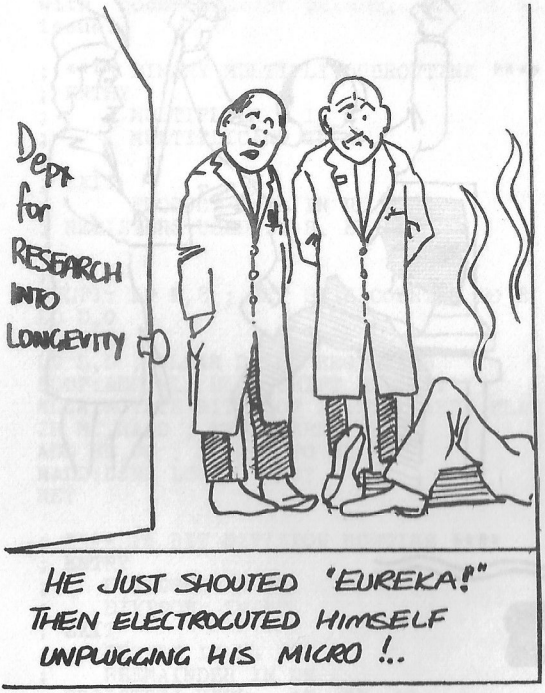
MZ-80B NOTES

The MZ-80B has quickly established itself on the UK market as an easy to use computer with plenty of advanced features. We have received many requests from subscribers for information on the operation of the MZ-80B. In the future we will publish a regular MZ-80B column provided Users send us contributions for publication. Please write to us if you have any tips or hints etc. on the operation of the MZ-80B. Programs including high-resolution graphics routines will be especially interesting to all MZ-80B Users.

MZ-80K and MZ-80B Monitor Routines

The table below details the monitor routines common to both the MZ-80K and MZ-80B Computers.

Subroutine	MZ-80K address	MZ-80B address
LETNL	0006H	08B0H
PRNTS	000CH	08B9H
PRNT	0012H	0916H
MSG	0015H	08DBH
MELDY	0030H	0EE9H
XTEMP	0041H	0DF8H
TIMST	0033H	0E06H
TIMRD	003BH	0E51H
BRKEY	001EH	0562H
GETL	0003H	06A4H
GETKY	001BH	0871H
ASC	03DAH	05F3H
HEX	03F9H	05FDH
HLHEX	0410H	0614H
2HEX	041FH	00623H
?DPCY	0DDCH	0A6EH
?PONT	0FB1H	0C53H



MZ-80B Manual Amendments

Two amendments relating to software/manual updates for the MZ-80B:

I. With respect to the monitor program for MZ-80B, please revise the text concerning the Save command (S command) to read as follows:

Affected: Monitors, SB-1510 and SB-1511

Revised Text: To save a memory block onto the cassette tape using the monitor command "S", the address value to be keyed in as the end address (E-ADR.) should be **"Final Address Number of Memory Block to be Saved plus One (1)".**

For example, to save the program contained in a memory block from $6000 to $60A2 using the S command and with a file name "ABRACADABRA" assigned, enter:

S-ADR . $6000

as the star address followed by the <CR> key, then enter:

E-ADR . $60A3 .....($60A2+1)

as the end address followed by the <CR> key.

You will have a display as follows.

II. PASCAL Manual (MZ-8BT02) Page 77 Graphic Display program line 4 should read:-

```
LINE (160, 0, TRUNC (COS (FLOAT (A) * 3. 1415927/150.0) *150.0) +
160, 100, 160, 199)
```

MZ-80B Tape BASIC

Although the SHARP MZ-80B tape BASIC includes both POKE and PEEK statements the internal operation of the computer electronics does NOT allow direct POKE or PEEK operations to the VDU Screen RAM. For those who are technically minded a form of bank switching has been used by SHARP to increase the MZ-80B ROM and RAM beyond the normal Z80A 64K. The details of the electronics are given in the MZ-80B Owners Manual.

The following MZ-80B BASIC subroutines allow PEEK and POKE operations to the VDU screen. These routines have NOT been optimized for speed but are presented in a simple form to illustrate the memory switching technique. A simple POKE demonstration routine is also included.

```
10 REM
20 REM
30 REM PEEK FROM VDU ROUTINE
40 REM FOR THE MZ80B.
50 REM
60 REM
70 REM ADD = ADDRESS IN DECIMAL
80 REM CHR = CHARACTER CODE IN
90 REM       DECIMAL
100 REM
61000 INP@232,H
61100 H = INT(H+128)
61200 OUT@232,H
61300 CHR = PEEK(ADD)
61400 H = INT(H-128)
61500 OUT@232,H
61600 RETURN
```

```
10 REM
20 REM
30 REM POKE TO VDU ROUTINE
40 REM FOR THE MZ-80B.
50 REM
60 REM
70 REM ADD = ADDRESS IN DECIMAL
80 REM CHR = CHARACTER CODE IN
90 REM       DECIMAL
100 REM
60000 INP@232,H
60100 H = INT(H+128)
60200 OUT@232,H
60300 POKE ADD,CHR
60400 H = INT(H-128)
60500 OUT@232,H
60600 RETURN
```

```
10 REM
20 REM
30 REM DEMONSTRATION POKE TO SCREEN
40 REM FOR THE MZ-80B.
50 REM
60 REM
70 REM ADD = ADDRESS IN DECIMAL
80 REM CHR = CHARACTER CODE IN
90 REM       DECIMAL
100 REM
500 REM CLEAR SCREEN
600 PRINT CHR$(6)
700 REM
900 REM MAIN TEST LOOP
1000 FOR I=1 TO 100
1100 CHR = INT(RND(1)*255)
1200 IF CHR < 30 THEN 1100
1300 ADD = INT(RND(1)*1000) + 53248
1400 GOSUB 60000 : REM POKE CHR
1500 NEXT I
1600 END
60000 INP@232,H
60100 H= INT(H+128)
60200 OUT@232,H
60300 POKE ADD,CHR
60400 H = INT(H-128)
60500 OUT@232,H
60600 RETURN
```

MEMBERS' LETTERS

Dear Sir,

Thank you for Issue No. 4 and back copies of your other issues. As a Sharp MZ-80B owner, I was very interested in the section on the MZ-80B by Mr. R.A. Wymark of Birmingham, but I seem to have a problem, I have attempted to run his "BASIC Copy Routine" and his "Utilities" program on my machine but they will not run, I have double checked the program listings on my machine, with the listings published in your book and I can't find any mistakes. The specific problems seem to be that when I make a copy of my BASIC tape according to Mr. Wymark's instructions I cannot then reload the copy of BASIC into my machine, and as far as the Utilities program is concerned, it will not run, and I suspect as far as the Renumber program is concerned, I am Poking into the program itself and it crashes. I have had the MZ-80B for some months now and can program in BASIC fairly proficiently, having created some 40K programs for work, but I don't profess to understand the intricacies of machine language and Peeks and Pokes. I would therefore appreciate some advise as to where I am going wrong. The BASIC language that I use is the 1.1 version which replaced the one that came with the machine.

M.J. STANLEY
NOTTS

****

Dear Sharpsoft,

Thank you for your swift reply to my query about Fig-FORTH.

I've been working through the User Notes in order to get the feel of the language and I've found a few errors.

1. BYE exits to monitor (not MON).
2. Screen clear 3 should be:
   :SCREEN-CLEAR3 DECIMAL 22 EMIT;
3. In the IF...THEN TUTORIAL 8, the final THEN DROP should be left out.

4. Prime numbers - TUT. 12 has a DUP missing between :PPRIME and 2 DO.

I am enjoying learning FORTH and have made the following CURSOR CONTROL Program, which may be of interest.

```
 0;  (CURSOR CONTROLS)
 1)  DECIMAL
 2)  : CU-R DUP 0 DO 19 EMIT LOOP;
 3)  : CU-L DUP 0 DO 20 EMIT LOOP;
 4)  : CU-U DUP 0 DO 18 EMIT LOOK;
 5)  : CU-D DUP 0 DO 17 EMIT LOOP;
 6)  : SCR-CL 22 EMIT;
 7)  : HOME 21 EMIT;
 8)  : CU-PL DUP 4466 1 ROT FILL
 9)    DROP DUP 4465 1 ROT FILL;
10)  ( CU = CURSOR, R = RIGHT, L = LEFT, U = UP)
11)  ( D = DOWN, PL = PLACE)
12)  ( EXAMPLES)
13)  : CENTRE-PRINT 20 112 CU-PL ."";
14)  : CIRCLE SCR-CL 19 CU-R 12 CU-D 131 EMIT 136 EMIT
15)    2 CU-L 1 CU-D 132 EMIT 137 EMIT 6 CU-D;S
```

MS. J. BENNETT
CORNWALL

Gentlemen:

Many thanks for the latest Sharpsoft User Notes and the FORTH tapes. The Notes are excellent as usual.

Comments:

Personally I prefer the current A5 size as it is handy to use, easy to file and all 4 have been that size - Much easier to keep on the shelf for use than an A4 size. But I have been an editor/publisher of several limited interest publications and know these are problems - so do what is necessary but keep up the quality and coverage.

Next - how about program conversion information, PEEK & POKE comparisons, Memory locations, etc. for those who want to use MZ-80K programs on the new MZ-80A??? I understand there are some differences. O.K.??

M.G. WALLIS
SUSSEX

SHARP provide a conversion program for MZ-80K-to-MZ-80A BASIC conversion. (Note it does NOT convert POKE and USR addresses.) As the differences between the K and A BASIC's and Monitor routines become known we will publish them in these User Notes.

EDITOR

****

Dear Sir,

Thank you for your Sharpsoft User Notes which are a welcome addition to the home computing scene. If I may I would like to offer some pointers on FORTH which I received from you some while ago.

The Fig-FORTH standard editor (which you have copied from the installation manual) is not adequate for the serious programmer who after being weaned on BASIC would surely miss the screen editing facilities offered by the monitor. It is possible with a little thought to re-organise the internals of FORTH to accept lines from the screen into PAD directly by substituting the organised address of PAD into Register DE and then calling the monitor routine at 0003H. Some juggling of pointers may be required but it would make the whole system more interactive, which was one of the main design features of FORTH!

There is a bug in the editor as used in your version from the installation manual. The word 'T' leaves a number on the stack! To correct this change the definition of 'T' by editing out the second 'DUP'.

The most serious deficiency in the present version of your implementation is the screen storage on cassette. While I appreciate that the system owes a lot to CP/M for your virtual storage system it is really not necessary on a tape-based system. I have implemented the 8080 version of fig-FORTH from the listing and deleted all references to CP/M and inserted the monitor calls as appropriate.

The way around tape storage is simple. The installation manual gives a good routine for testing 'R/W' using memory as virtual storage. I used this routine, modified the addresses, and simply call the monitor routines to save a complete block of memory with a suitable header onto tape. The monitor does all the checking necessary to make sure the screen names is loaded. I made two new FORTH words to help the implementation: F-TAPE and T-TAPE, which load from tape into memory named 1K screens and from memory to tape. Tape usage is much improved and faster and gives the additional bonus that screens may be overlayed while a program is running to give self modifying progams! Initialisation of the tape is of course not necessary. I will supply a listing of the words and modified R/W if the interest is high enough.

I have just read with disappointment your plans to produce a compiled PASCAL. I am at present using the excellent Hisoft Pascal 3 for the Sharp and I doubt whether you could improve on this present version. Would it not be better to release this version more widely under licence, say from Hisoft with a few additions of your own say. I feel it would be a far more efficient way to use the programming effort than to develop your own. I have a few suggestions for improvements which I would gladly send to you or implement them.

Please carry on the good work despite my criticism!

G. FEARNHEAD
ESSEX

****

Sharpsoft,

Re: Letter Mr. D.H. Jackson, Yorkshire
(User Notes 5, page 42)

Cassette-speed 4 MHz: two warnings:

If an Epson MX 80 FT printer with MZ-80K Module is used, BASIC SP 5025 needs minor alteration (easily carried out), however, some systems-programs then cannot use/have not access to printer.

FORTH works well with this combination !

Only use quality tapes (brand names) - otherwise one will encounter mechanical trouble - no prediction possible. Only use Cassettes C60 or C45 or C10.

Otherwise, 4 MHz is great - 2 x computer power at virtually zero cost.

M. HERMANN Dipl.Phys.
WEST GERMANY

See my notes on how to copy the FORTH tapes in this issue's FORTH column.

EDITOR

****

Dear Sirs,

With great regret I have to report serious problems with the Fig-FORTH tapes and User Notes No.4 received a month or so ago but which I have only just got round to trying. I list the problems somewhat tersely below.

1. Why does my MZ-80K not display the same messages and output format as the Notes give?

2. Why do the Editor and Assembler have so very long a run-in? (to tape counter 043, apparently). At first I believed this tape to be blank! Also, who do illegible messages appear on loading? - see attached sheets.

3. Why do some of the "tutorial" examples not work?? Specifically, in tutorial 5 versions 1 and 3 do not clear the screen (and version 2 reaches only 400 hex counter, not 800 hex): tutorial 8 entered in any way crashes back to the SP-1002 monitor: tutorials 10 and 11 produce absolute rubbish, with output starting 1 20300...., followed later by a long series of zeros, then ones, ending with 11512 11512 1 7185 and total lockup.

4. Why is there no explanation of the mechanism of writing and recording FORTH programs, either in listed or compiled forms?

R.F. FEAR
BERKSHIRE

This issue's FORTH notes will help clear some of these questions.

********** EDITOR

Dear Sirs,

Please find enclosed Circuit of an Interface Card, I have made for a Sharp MZ-80K. I use the Computer in my Dark Room, and the card is designed to switch the Enlarger and other devices on and off, up to eight in all.

IC's LS10 and LS27 connect to the Address Bus along with WR signal. When WR goes low and the Address Bus has 50,000 on it. Output A goes low, e.g. Poke 50,000,

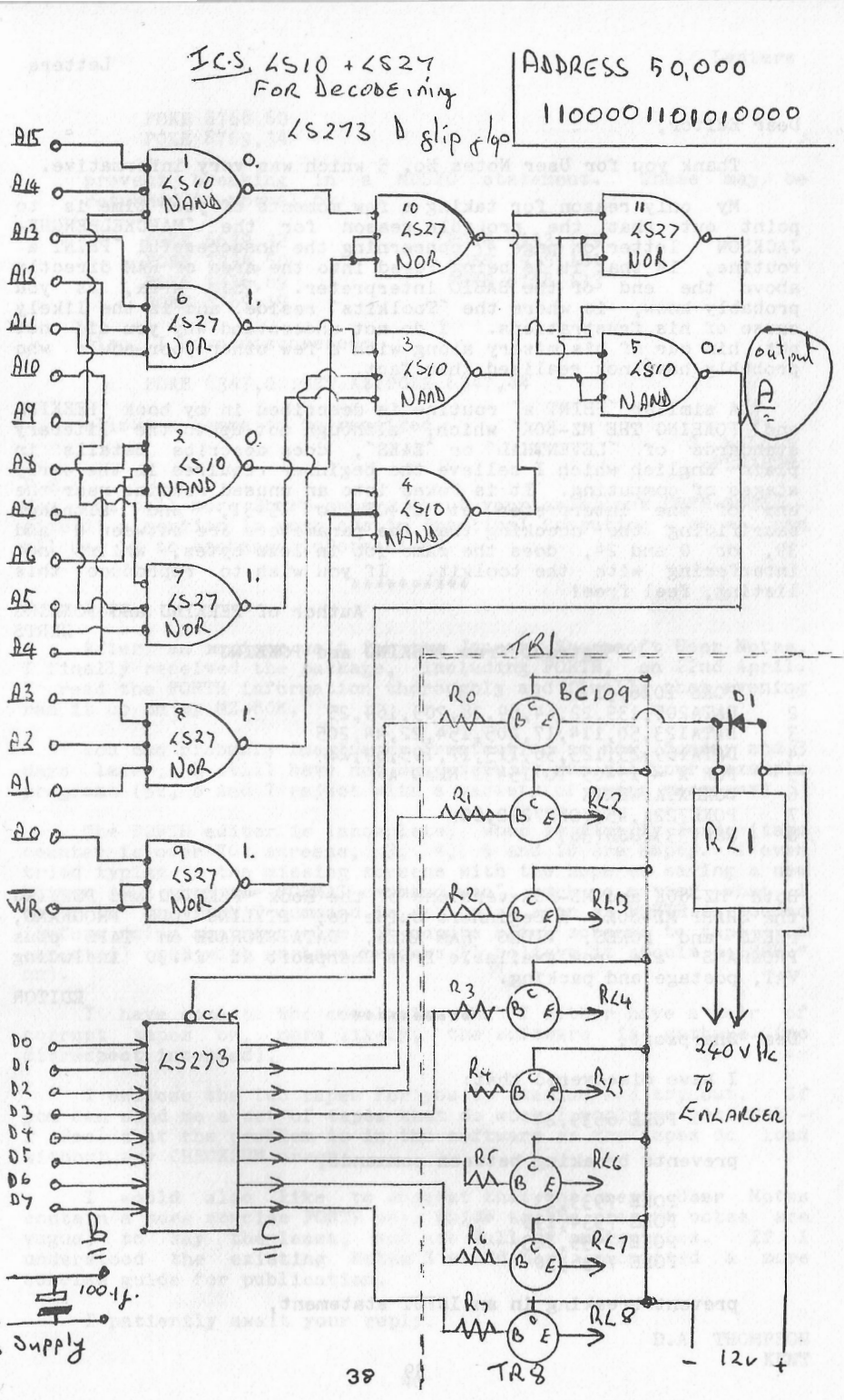
Output A is connected to the Clock of IC - LS278. which is a D-Type Flip Flop. When the Clock goes low DATA DO- D7 is transferred from B to C turning TR1 to TR8 on or off according to Data.

The Transistors in turn switch on a 12 volt supply which turns on the Relay.

I have up to now had this board working switching various devices, but quite often the Computer goes into a Loop and I have to switch off the Computer and start again.

I would be very grateful if anyone could solve this problem.

N.D. OSMOND
GLOS.


IC.S LS10 + LS27
FOR DECODEING
LS273 D slip flop
ADDRESS 50,000
1100001101010000
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
WR
1 LS10 NAND
2 LS10 NAND
3 LS10 NAND
4 LS10 NAND
5 LS10 NAND
6 LS27 NOR
7 LS27 NOR
8 LS27 NOR
9 LS27 NOR
10 LS27 NOR
11 LS27 NOR
output A
TR1
R0
BC109
D1
R1
RL2
R2
RL3
R3
RL4
R4
RL5
R5
RL6
R6
RL7
R7
RL8
RL1
240v Ac
TO ENLARGER
CLK
LS273
D0
D1
D2
D3
D4
D5
D6
D7
5v
100 µf.
chip supply
- 12v +
TR8
38

Dear Editor,

Thank you for User Notes No. 5 which was very informative.

My only reason for taking a few moments of your time is to point out that the probable reason for the ´MAECKELBERGHE-JACKSON´ letter on page 47 concerning the unsuccessful ´PRINT a´ routine, is that it is being Poked into the area of RAM directly above the end of the BASIC interpreter. This area, as you probably know, is where the ´Toolkits´ reside, and is the likely cause of his frustrations. I do not understand why you did not put him our of his misery along with a few other poor souls who probably have not realised this fact.

A similar ´PRINT a´ routine is described in my book ´PEEKING and POKEING THE MZ-80K´ which, although not up to the literary standards of ´LEVENTHAL´ or ´ZAKS´, does describe details in plain English which I believe the beginner requires in the early stages of computing. It is Poked into an unused routine near the end of the interpreter at 15405 to 15431, and although sacrificing the checking that the parameters are between 0 and 39, or 0 and 24, does the same job in less bytes, whilst not interfering with the toolkit. If you wish to reproduce this listing, feel free!

Author of PEEKING and POKEING
HERTS

"Extract from PEEKING and POKEING"

```
1    POKE 10167,1
2    DATA205,139,22,64,69,28,205,169,25
3    DATA123,50,114,17,205,154,22,44,205
4    DATA169,25,123,50,113,17,195,69,28
5    FOR X = 15405 TO 15431:READ A
6    POKEX,A:NEXTX
7    POKE7221,45:POKE7222,60
8    USR(33):USR(36)
```

Both MZ-80K and MZ-80A versions of the Book "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" are now available from Sharpsoft at £4.30 including VAT, postage and packing.

EDITOR

****

Dear Sharpsoft,

I have discovered that:

```
POKE 6639,24
```

prevents breaking between commands,

```
POKE 7933,13
POKE 7934,195
POKE 7935,183
POKE 7936,30
```

prevent breaking in an INPUT statement,

```
POKE 8768,60
POKE 8769,34
```

prevent breaking in a MUSIC statement. These may be returned to normal by:

```
POKE 6639,32          POKE 7933,129
POKE 7934,50          POKE 7935,101
POKE 7936,69          POKE 8768,133
POKE 8769,19
```

Also, I have discovered:

```
POKE 6347,0 :GET A$:POKE 6347,44
```

allows a comma to be received.

S. WALLIS
PENARTH

P.S.
Why not tell more Sharp-owners about your excellent USER NOTES - give it a mention in your ads in Practical Computing - so we can all use it to exchange information?

****

Dear Sirs,

After an anxious wait for the January Sharpsoft User Notes, I finally received the package, including FORTH, on 22nd April. I read the FORTH information thoroughly and finally that evening ran it up on my MZ-80K.

You can probably imagine my frustration as now, Sunday and 3 days later, I still have not successfully run all your example programs (5c, 6 and 7 reject with a variety of error messages).

The FORTH editor is incomplete, when it finally loads (tape counter is over 70) screens, 3, 4, 5 and 10 are empty. I even tried typing in the missing screens with the hope of saving a new Editor but then the FORMAT command won't work no matter what I do. If the FORMAT command is set up as soon as FORTH is loaded (before doing anything else) it ouputs blank screens to tape (as expected) but with garbage headers (I believe it should be SCR " nn).

I have come to the conclusion that I either have a pair of corrupt tapes or, more likely, the software is garbage (no disrespect intended).

I enclose the two tapes for you to examine and try out. If you can send me a set of tapes that do work I would be grateful - I feel that the problem is in the software as the tapes do load without any CHECKSUM error.

I would also like to suggest that the next User Notes contain a more concise FORTH user guide as the present notes are vague, to say the least, and are full of ambiguities. If I understood the existing Notes I would have submitted a more concise guide for publication.

I patiently await your reply.

D.A. THOMPSON
KENT

I hope Mr. Thompson has managed to load his replacement tapes! If you have difficulty loading any of our tapes let us know and we will do our best to sort out the problem as quickly as we can. Sending tapes by post does cause problems sometimes.

EDITOR

****

Dear Sir,

Thank you for Issues 3 and 4 of the User Notes. I was amazed at the quality, much better than the ones of another User-Group!

Having worked my way through the Tutorial Examples for Beginners, there is still about 99.99% recurring of the fig-FORTH Glossary I don't understand! I have also wondered why the language crashed so easily. By SHIFT & INSERT, I got the message PRINTER OFF as expected but thereafter pressing any other gives a PRINTER OFF message as well! Therefore I've had to switch off and start all over again!

Although I've been having problems, I enjoyed using FORTH (more so than BASIC), partly because of its remarkable speed and especially its unusuality! I would like to extend my knowledge of FORTH (in other words learn to program in it) so could you recommend a book for beginners (like me) on fig-FORTH and using the Text Editor and Assembler as I've had little success with my local bookshops.

Also, some readers will probably find the series "GOING FORTH" by D.S. Peckett in magazine COMPUTING TODAY in the January 1982 to April 1982 editions very useful as I have found. The article "Not fifth but FORTH" by M.E. BRINSON (the Editor!) in the October 1981 edition of Computing Today is no longer available as back issues but photocopies of the article are available from them at £1.25, the same price as back issues themselves! The form for the photocopies and back issues is found in the magazine.

C. CHOI
NORTH HUMBERSIDE

GULP! - a red face on my part - thanks for your kind letter. Two other books on FORTH are "DISCOVER FORTH" by Osborne at £10.95 (highly recommended for beginners) and "Intro. to FORTH" at £6.95 both available from SHARPSOFT.

EDITOR

****

Dear Sharpsoft,

I couldn't get FORTH Tutorials 10 and 11 to work. I've altered them as follows:

```
Tut. 10   :KOUNT 0 BEGIN 1 + DUP DUP . CR
          100 = UNTIL ;

Tut. 11   :SQUARE 1 BEGIN DUP DUP *
          DUP 1500 " WHILE
          SWAP DUP . SWAP . 1+ REPEAT;
```

It was good training to FIGure them out! FORTH Cold START ADDRESS 1200H, warm START 1202H.

P. BENNETT
CORNWALL

Dear Sirs,

On attempting to run some benchwork tests to discover comparisons with equivalent BASIC programs, I discovered that the following non-standard PASCAL program, operating under Sharp SP-4015 will fail at the line indicated:-

```
                                                    or
      VAR       A: REAL     ;          )
                K:  INTEGER ;          )          A,K:INTEGER;
                BEGIN
                  WRITELN ('S') ;
                  K:=0 ;
                  WHILE (K<1000)  DO                    REPEAT
                  BEGIN                                 BEGIN
                   K:=K+1;
  FAILURE -        A:=K/K*K+K-K;                        END;
                  END:                                  UNTIL -
                END
```

A second point, resetting the clock via TI$="000000" in Speed Basic corrupts the interpreter.

I would be grateful for your comments on these points.

DR. G. HAYHURST
LANCASTER

Firstly, any comments from other PASCAL users? The Speed Basic problem was a 'bug' found in the first versions sold and has since been corrected. Any User with this fault on their tape should return the original to us for replacement with a corrected version.

EDITOR

****

Dear Sir,

I am including a couple of tips which you may like to pass on to other Users.

Regarding the Article in User Notes 3 "Boolean Algebra" on the MZ-80K by P.L. Birch. There seems to be an error in his and the MZ-80K logic. In the section AND/OR, page 50 paragraph 2, he states that if 2 time statements are used in an 'and' logic statement then the result is time i.e. -1 x -1 = 1 non zero therefore time. Then in the 'or' statement a truth table is printed. But if a statement is (TRUE AND TRUE) OR TRUE then the logic gives a False output:

i.e. (-1 x -1) + -1 = 0 i.e. (+1) + -1 = 0.

To make the logical Time +1 Poke 8861,20 Poke 13541,20 and Poke 13561,20. To restore Time to -1 Poke 8861,30 Poke 13541,30 and Poke 13561,30.

Use of GET statement. In User Notes 3 several people suggested ways of getting a continuous output from the 'get' statement.

I have another way of doing this, such that the output can be continuous or single shot (as in normal GET) depending on the input.

This can be used in a shooting game i.e. continuous movement for the gun but only one shot at the time for the gun

```
10   GETA$:IFA$=" " THEN 10
20   IFA$=L THEN POKE 17828,0:GOTO
     (LEFT MOVEMENT ROUTINE)
30   IFA$=R THEN POKE 17828,0:GOTO
     (RIGHT MOVEMENT ROUTINE)
40   IFA$=S:GOTO (SHOOT ROUTINE)
```

Note No Poke in Line 40

J.H. BELL
CHESHIRE

****

Dear Sir,

I am thinking of trying to upgrade my SHARP MZ-80K to 64K (or more) here at Marlborough. I am writing to you to ask you if the machine is capable of addressing 64K or more of RAM. I am probably capable of working all the information out, but I would be grateful if you could advise me on what IC's to use, and anything else that I might need to know to undertake this project.

P. HENDRIE
MARLBOROUGH

The MZ-80K would need considerable modification (probably a complete rebuild) to extend the RAM to 64K or more. Above 64K some form of bank switching would be required as the Z80 can only address 64K i.e. A0 - A15 address lines only. To our knowledge no commercial attempts have been made to increase the RAM size above 48K. However, a 64K RAM conversion is now available for the MZ-80A.

EDITOR

****

Dear Sir,

Thank you for sending the replacement Sharpsoft fig-FORTH tape. This seems to load correctly, though it is not possible for me to be sure until I have found out more about the fig-FORTH system. (I am a bit worried by the message 'MSG No. 4' which I get when loading the Editor). I hope you will consider giving more information about fig-FORTH in future issues of the User Notes. Topics I would particularly like to see discussed are:

The Error(?) Messages "MSG No. ...";
Screen storage and the Editor; and
The 8080 Assembler.

P.G. DIXON
SHEFFIELD

See FORTH column in this Issue for Error Messages.

EDITOR

Dear Sirs,

Thank you for publishing my last letter, the following might also be of interest to your readers.

I. Bit 2 of location E002H changes the colour of the LED, green is set (or 1) red is reset (or 0), more useful than loading in a number as this will alter other things as well.

II. Bit 0 of location 1170H changes between upper and lower case, reset and set respectively.

III. Call 000FH will tab the ´cursor´ across the screen in steps of 10, this is therefore similar to the BASIC "," in the PRINT instruction.

IV. A complete list of monitor subroutines is enclosed, can anyone explain what CALL 0018H does as it jumps to a location a little way after the start of MSG and appears to do the same job.

V. Since my last letter I still haven´t managed to understand the printer subroutines but if they are used with the enclosed 3 subroutines, these will directly replace LETNL PRNT and MSG.

VI. Point S of my last letter is incorrect, it is possible to relocate a file in memory by changing the start address pointer after CALL 0027 but before CALL 002A. This will cause problems unless the file is either a data file or totally relocatable (very unlikely). Auto start doesn´t appear to be changeable, another problem.

VII. On the subject of totally relocatable programs, the following subroutine simulates a JR M or JR P by shifting the F register are placed to the right, a JR Z or JR NZ will then perform the require relative jump.

```
PUSH DE   D5
PUSH AF   F5
POP  DE   D1
RR   E    CB 1B
PUSH DE   P5
POP  AF   F1
POP  DE   D1
RET       C9
```

replacing the RRE with two RRE´s it is possible to jump relative on PARITY/OVERFLOW, in this case a JR C should be used instead of the JR Z.

VIII. For confused users of PASCAL SP-4015 the insert command is - (shifted E) not the chequered square.

Also there is a command T (line No) which will print out that line, is there any special reason for including this command as it is covered in the P command.

Finally to echo P Braithwaite's point about the "clean" computer design, I now have 8 different language tapes (and numerous machine code programs about a total of 80K!!

I hope these points will be of interest.

W. HOWARD
HERTS

** Z80 ASSEMBLER SP-2102 PAGE 01 ** MONITOR ROUTINES **

```
0000                    ;
0000                    ;       A COMPLETE LIST OF MONITOR SUBROUTINES.
0000                    ;       ======================================
0000                    ;
0000 C30000      E              JP      TRUEST          ;JUMP OVER SUBROUTINES
0003 C30000      E              JP      GETL            ;GET LINE OF TEXT
0006 C30000      E              JP      LETNL           ;PRINT A <CR> <LF>
0009 C30000      E              JP      NL              ;OUTPUT A <CR> <LF> ONLY
000C                    ;                               IF NOT AT TOP OF SCREEN
000C C30000      E              JP      PRNTS           ;PRINT A SPACE
000F C30000      E              JP      TAB             ;SAME AS BASIC ,
0012 C30000      E              JP      PRNT            ;PRINT ACCUMALATOR
0015 C30000      E              JP      MSG             ;PRINT MESSAGE
0018 C30000      E              JP      MSG2            ;SIMILAR TO MSG.WHY?
001B C30000      E              JP      GETKY           ;INPUT ONE CHARACTER
001E C30000      E              JP      BRKEY           ;CHECK FOR BREAK
0021 C30000      E              JP      HEAD.O          ;OUTPUT TAPE HEADER
0024 C30000      E              JP      DATA.O          ;OUTPUT TAPE DATA
0027 C30000      E              JP      HEAD.I          ;INPUT TAPE HEADER
002A C30000      E              JP      DATA.I          ;INPUT TAPE DATA
002D C30000      E              JP      VERIFY          ;VERIFY TAPE
0030 C30000      E              JP      MELDY           ;PLAY MUSIC
0033 C30000      E              JP      TIMST           ;SET CLOCK
0036 00                         NOP
0037 00                         NOP
0038 C33810                     JP      1038H           ;RESTART 38H
003B C30000      E              JP      TIMRD           ;READ CLOCK
003E C30000      E              JP      BELL            ;RING BELL
0041 C30000      E              JP      XTEMP           ;CHANGE TEMPO
0044 C30000      E              JP      MSTA            ;START SOUND
0047 C30000      E              JP      MSTP            ;STOP SOUND
004A                    NOP;MONITOR START
004A                    ;
004A                    ;SEE ISSUE 1 PAGE 36 OF USER NOTES
004A                    ;FOR A DESCRIBTION OF THESE
004A                    ;
004A C30000      E              JP      ASC             ;03DAH
004D C30000      E              JP      HEX             ;03F9H
0050 C30000      E              JP      HLHEX           ;0410H
0053 C30000      E              JP      2HEX            ;041FH
0056 C30000      E              JP      ??KEY           ;09B3H
0059 C30000      E              JP      ?ADCN           ;0BB9H
005C C30000      E              JP      ?DACN           ;0BCEH
005F C30000      E              JP      ?BLNK           ;0DA6H;WAIT FOR BLANKING
0062 C30000      E              JP      ?DPCT           ;0DDCH
0065 C30000      E              JP      ?PONT           ;0FB1H
0068                            END
```

```
** Z80 ASSEMBLER SP-2102 PAGE 01 ** PRINTER SUBROUTINES **


01 0000                 ;
02 0000                 ;-----PRINTER SUBROUTINES----
03 0000                 ;     ====================
04 0000                 ; Output will only be passed
05 0000                 ;to the printer if Bit 0 of
06 0000                 ;location PFLAG is Set.
07 0000                 ;
08 0000                 ;-----------------------------
09 0000                 ;
10 0000                 ;     LINE FEED (LETNL)
11 0000                 ;
12 0000                 PLETNL: ENT
13 0000 F5                      PUSH  AF
14 0001 E5                      PUSH  HL
15 0002 CD0600                  CALL  LETNL             ;OUTPUT (CRLF) TO SCREEN
16 0005 210000      E           LD    HL,PFLAG
17 0008 CB46                    BIT   0,(HL)            ;PRINTER OUTPUT WANTED ?
18 000A 2805                    JR    Z,NO
19 000C 3E0D                    LD    A,0DH             ;(CRLF) CODE
20 000E CD0000      E           CALL  PRINT
21 0011 E1              NO:     POP   HL
22 0012 F1                      POP   AF
23 0013 C9                      RET
24 0014                 ;
25 0014                 ;   CHARACTER PRINT (PRNT)
26 0014                 ;
27 0014                 PPRNT:  ENT
28 0014 F5                      PUSH  AF
29 0015 E5                      PUSH  HL
30 0016 CD1200                  CALL  PRNT              ;OUTPUT CHAR TO SCREEN
31 0019 210000      E           LD    HL,PFLAG
32 001C CB46                    BIT   0,(HL)            ;PRINTER OUTPUT WANTED ?
33 001E 28F1                    JR    Z,NO
34 0020 CD0000      E           CALL  PRINT
35 0023 18EC                    JR    NO
36 0025                 ;
37 0025                 ;     MESSAGE PRINT (MSG)
38 0025                 ;
39 0025                 PMSG:   ENT
40 0025 F5                      PUSH  AF
41 0026 D5                      PUSH  DE
42 0027 E5                      PUSH  HL
43 0028 CD1500                  CALL  MSG               ;OUTPUT MESSAGE
44 002B 210000      E           LD    HL,PFLAG
45 002E CB46                    BIT   0,(HL)            ;PRINTER OUTPUT WANTED ?
46 0030 280E                    JR    Z,NOT
47 0032 1A              PMSG1:  LD    A,(DE)            ;GET NEXT CHARACTER
48 0033 FE0D                    CP    0DH               ;CHECK FOR END
49 0035 2806                    JR    Z,PMSG2
50 0037 CD0000      E           CALL  ?PRNT             ;OUTPUT CHARACTER
```

```
** Z80 ASSEMBLER SP-2102 PAGE 02 ** PRINTER SUBROUTINES **

01 003A 13                          INC   DE                ;INCREMENT PTR
02 003B 18F5                        JR    PMSG1             ;LOOP BACK UNTIL END
03 003D CD0000      E     PMSG2:    CALL  STCK              ;CHECK STATUS
04 0040 E1                  NOT:    POP   HL
05 0041 D1                          POP   DE
06 0042 F1                          POP   AF
07 0043 C9                          RET
08 0044                     ;
09 0044                     ;       CONSTANTS
10 0044                     ;
11 0044  P                  LETNL:  EQU   0006H
12 0044  P                  PRNT:   EQU   0012H
13 0044  P                  MSG:    EQU   0015H
14 0044                     ;PFLAG Any convient memory location.
15 0044                     ;PRINT Printer control subroutine.
16 0044                     ;?PRNT      "         "         "
17 0044                     ;STCK       "         "         "
18 0044                           END
```

Dear Sir,

I have bought a Sharp MZ-80K (48K) from a gentleman in Welwyn Garden City about five months ago. I would like to take the opportunity to seek your advice regarding the problems I have encountered while doing some programs using the Sharp Basic SP-5025. Here is an example program:

```
10 read N$,A,C
20 PRINT A
Rest of program
100 RESTORE 540
Rest of program
530 DATA"PAUL",4,9
540 DATA"TIM",12,7
550 DATA"HARRY",3,9
```

This does not restore the data from the number 540 but from the beginning of the data. I would like you to advise me whether this type of difficulty is due to me overlooking some vital programming technique or could it be due to my Sharp Basic SP-5025 cassette being defective or could it be a fault in the computer itself?

I am looking forward to your reply on your earliest conveniency.

S. SUNDERANI
ESSEX

BASIC SP-5025 does not include the statement RESTORE "Line number", only RESTORE.

EDITOR

****

Dear Sirs,

An initial run through the FORTH Tutorial has shown some oddities: At page 51 only Screen-Clear2 seems to work. Trying to run the other two just results in "OK".

At page 52 the character display is quite a revelation in speed, and more than anything else spurs one on to master FORTH.

Page 53. I cannot get this Hex dump to work; it only produces a long column of integers mostly "5" towards the end, and finishes with a star, and I am back in Monitor!

At page 54 in the DIVN program there are two "THENs" before "DROP LOOP"

At page 57: I have very carefully repeated the "SQUARE" program but get the following:-

```
SQUARE    1       20819
-7029    16725
-13843    17746
SQUARE  ? MSG No. 1
```

I am baffled. A listing of the MSG. Nos. would be useful. Thank you for a very interesting issue.

G.O. HAYWARD
DYFED

I am baffled also!

EDITOR

****

Dear Sirs,

The package has been well worth the wait. Frustratingly, however, the interpreter does not load quite correctly, only first couple of tutorial programs.

Principally, loading is completed with "An implementation of Fig-Forth by Dr. M. Brinson and A. Grey MSC for Sharpsoft Ltd." and flashing cursor. (There is no OK before cursor?)

VLIST execution gives all the existing interpreter words spelt with the last letter replaced by a Sharp graphics symbol.

Definitions split over several lines cause various random error messages, sometimes the first word in a new line is "not recognised", sometimes the delimiting right parenthesis of a comment is "not recognised" as a legal word. Only your tutorial programs. Only your tutorial programs have been tried carefully.

The Editor finishes loading with MSG No. 4 - MSG No. 4 OK (Both with spurious graphics characters intermingled therein) Screens 3, 4, 5, +10 list as "empty" screens. Screens 7 and 8 can also be listed by 17 LIST (CR) PLAY BREAK and 18 LIST etc. respectively as well as normal 7 and 8 LIST.

Lower case characters as per tutorial programs are not available, SHIFT/SML does not operate, no L.E.D. change green-red.

The assembler tape loads with several lines of mixed graphics and ? MSG No. 4's. All assembler screens appear to list normally.

Please will you test both tapes and exchange if necessary.

Please also, if there is a simple way to security copy the tapes, and please, is it possible to issue a list of ? MSG No. messages.

M.M. SHAW
MID-GLAMORGAN

Our FORTH column should answer most of your queries.

EDITOR

****

Dear Sir,

Firstly, thank you for the User Notes, as an MZ-80K (48K) user, I have found them very useful.

I understand that there is now a compiler available for Sharp BASIC, but that it can only be used with the FDOS Operating System. Can compiled programs be run on an MZ-80K machine without the FDOS Operating System? If so, do Sharpsoft offer a program compilation service for those of us who do not have disk drives? I would be grateful for any advice you can give me on this matter.

T. JONES
SURREY

Sorry we do not run a compilation service. Please note, as far as we have been able to find out, the Sharp BASIC Compiler will not produce stand alone cassette machine code programs. If anyone has been able to generate a machine code tape from the BASIC Compiler we would be very interested in publishing how it was done.

EDITOR

****

Dear Sharpsoft,

Two points which could be used in future User Notes:

1. For anyone using Graphic Chips and found that 'LIST/P' causes a form feed before listing starts, POKE 15478,0:POKE 15478,13 will cause a line feed. To return to normal POKE 15478,15.

2. For HIGH RES. GRAPHICS using Graphic Chips (and BASIC Modifier you sent me) a 16 bit Hex. No. is required. Up to now I have only been acquainted with 8 bit Hex. No. Could you explain how to work out a 16 bit Hex. No.

R.A. BAXTER
ESSEX

The answer is really quite simple! You are aware that eight bits are represented as two hexadecimal characters, the first character being the value of the left most four bits and the right character representing the value of the right four bits. To convert the hexadecimal value to decimal we multiply the value of the right hand character by 16 ^0, multiply the value of the left character by 16 ^1 and add the results together.

Example:

Nos. in Hexadecimal (base 16) from 0-15 are represented as follows:

```
Decimal       0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Hexadecimal   0 1 2 3 4 5 6 7 8 9  A  B  C  D  E  F
```

```
Binary              11100110
Hexadecimal           E   6

Convert to decimal,
     Left hand character,      E = 14d * 16^1 = 224
     Right character           6 =  6d * 16^0 =   6
                    Total                     = 230
```

To handle 16 bits we simply put two extra hexidecimal characters to the left of the first two characters. When converting this value to decimal we multiply value of the third character from the right by 16 ^2 , the value of the leftmost character by 16 ^3, carry out the calculations detailed above and add the whole lot together.

Example:

```
Binary              1011100111100110
Hexadecimal           B   9   E   6

Convert to decimal,
     Leftmost character       B = 11d * 16^3 = 45056
     Next character in        9 =  9d * 16^2 =  2304
     Next character in        E = 14d * 16^1 =   224
     Last character           6 =  6d * 16^0 =     6
                    Total                      47590
```

In the above text the character ^ signifies "to the power of".

Having grasped this you will be able to deal with any number of bytes so long as your maths, or calculator, is up to it. If you are still having problems I would suggest that you buy a Sharp 506H scientific calculator, at the special SHARPSOFT discount price of £15 inc. VAT. which is capable of working in hexadecimal.

EDITOR

****

Dear Sir,

I write out of frustration if not disgust.

Issue 2 of SUN suggested the production of a list of MZ-80K owners willing to exchange ideas, letters etc. Since then I have seen no mention of this excellent idea, and simply cannot believe the response did not warrant pursuance.

The second point concerns the FORTH language tapes which I purchased from you. To put it bluntly, I am of the opinion that they, in conjunction with your explanatory notes etc., are a joke. Am I right in my assumption that the FORTH GLOSSARY was written by a demented martian? Did you really, honestly believe that pages 2-66 of SUN Issue 4, would be of some use to more than a very small minority of your subscribers? Or is the whole venture just a commercial "come on"? Incidentally some of your TUTORIAL EXAMPLES don´t function.

In conclusion may I say, that, in my view your SUNs are becoming of less and less interest to me with each issue. Please, please, remember the bulk of your subscribers are not Chief Design Engineers at Ferrantis Ltd.

I enclose a S.A.E. should you desire to comment on the above constructive comments.

Yours, More in Hope than.....

W.A. Campbell
Tytherington, Macclesfield, Cheshire.

Dear Hopeful,

I am sorry to hear that you are becoming disillusioned with S.U.N. fortunately you are very much in the minority amongst nearly 1000 subscribers throughout the world!

Let me confirm that the answer to the MZ-80K user club was not well received and the cost of producing such a register in view of the negative response would have been exorbitant.

The FORTH Language tapes were given to each subscriber with the 1982 User Notes, and were produced by two very dedicated computer enthusiasts. Their comments with the notes were that these tapes should be used by the serious FORTH user together with publications such as "Invitation to FORTH", Katzen H. and "Using FORTH" FORTH, Inc. It surely was obvious that our brief introduction to the language in Issue No. 4 of SUN, would not replace a dedicated book on the subject.

Whilst I would agree that occasionally we feel like a bunch of demented Martians I do assure you that we do our utmost to ensure that SUN´s are produced with all levels in mind, right from the interested 10 year old - Mr. Zatman´s letter page 44-45 and R. Young Page 49-50 (Issue 4)right up to a number of ´young´ O.A.P.´s.

We have looked carefully through our correspondence and have found no previous contact from you, why not help us to improve SUN by sending your **constructive** contributions.

SHARPSOFT

Madrid, 9-10-82

Queridos amigos:

Muchas gracias for sus cassettes.

Leegaron en unos 20 dias!
perde mi orden..
Increible!

Mas rapido impossible.

Recuerdos,

Dear Friends:

Many thanks for your cassettes.
They arrived just 20 days from my order.
Incredible!

Faster - impossible.

Regards,

JULIAN COLINA

This appreciative letter arrived together with a photograph of this Member's MZ-80K, taking pride of place in his home in Spain. Yet another one of our satisfied International Members taking time out to thank us - Muchas Gracias Julian Colina.

THE SHARPSOFT TEAM

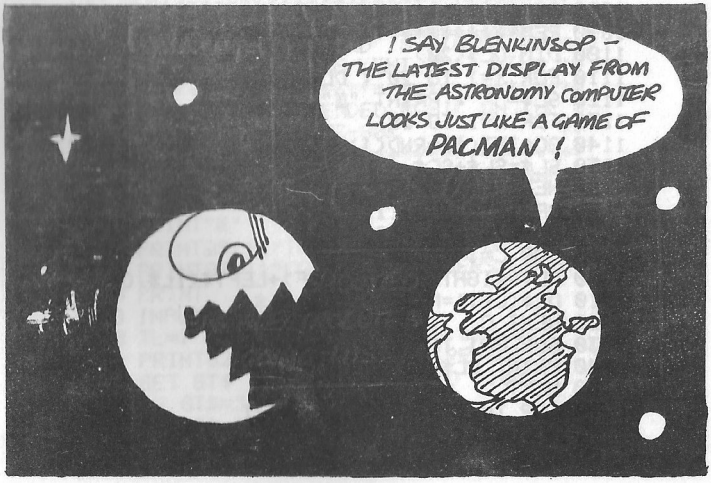
****

Dear Sir,

I write enclosing an order form for the Cassettes, and also to inform you that I have scored 67,780 on Sharpsoft's Asteroids program, beating the 50,490 score mentioned in your catalogue. I also congratulate you on your excellent User Notes, which have provded an invaluable aid to my programming.

B.J. HALLETT
SOMERSET

****



Two Programs from **Mr. Frank Woodward** written in Extended BASIC.

1. PATTERNS

A DIVERTISEMENT FOR THE SHARP MZ-80K

This program although very short and simple results in intriguing results on the screen, and as it is set in a loop gives a continually changing series of patterned areas on the screen.

The program operation is simplicity itself. First a string of randomly generated graphics characters is generated, and printed on the screen. The string is then offset by a randon number of characters, and "wrapped around", and this is repeated until the dipslay is build up.

Although as shown here the program only gives a changing series of displays, it could be used to generate background displays for other pgorams, and if the output is directed to the printer, and the number of lines of the display extended with perhaps a periodic reversal of the offset some fascinating print-outs can be obtained which might well interest designers or artists.

I hope readers will find this short program of interest and I would be pleased to hear from anyone who finds a use for it, and any extended variations on the theme.

FRANK WOODWARD
KENT

```
:000 REM**********************************
1010 REM**        PATTERNS                **
1020 REM** A 'DIVERTESEMENT' FOR THE MZ-80K **
1030 REM**                                **
1040 REM**                                **
1050 REM** BY-F.E.WOODWARD                **
1060 REM**       8 VIOLET AV.             **
1070 REM**         RAMSGATE               **
1080 REM**            KENT                **
1090 REM**********************************
1100 DIM DL$(20)
1110 FOR X=1 TO 20 : DL$(X)="" : NEXT X
1120 SL$="" : PRINT "@"
1130 FOR L=1 TO 38
1140 GC=INT(13*RND(1)+1)+204 : GC$=CHR$(GC)
1150 SL$=SL$+GC$
1160 NEXT L
1170 OF=INT(5*RND(1)+1)
1180 DL$(I)=SL$ : IL$=SL$
1190 FOR C=1 TO 19
1200 NL$=RIGHT$(IL$,38-OF)+LEFT$(IL$,OF)
1210 DL$(C+1)=NL$  : IL$=NL$
1220 NEXT C
1230 FOR P=1 TO 20
1240 PRINT DL$(P)
1250 NEXT
1260 FOR T=1 TO 3000 : NEXT T
1270 GOTO 1110
```

2. BIG CHARACTERS

A SUBROUTINE IN EXTENDED BASIC FOR THE SHARP MZ-80K

This subroutine enables characters, or text of upto five characters to be displayed on the screen in an enlarged format. The format is based on a five by seven matrix made up of the normal sized characters.

When used in a program the subroutine must be initialised by activation of a preliminary subroutine to fill the array BC$(37,7) from the DATA statements. This is initiated by placing a GOSUB 50000 statement early in the main body of the program using the subroutine.

The main subroutine is activated by first equating TX$ to the character, or text to be displayed, and TL to the top line at which the top of the enlarged characters are to appear in the main body of the program, and then a GOSUB 52000 statement.

As it stands the program will work on all the CAPITAL letters, NUMBERS and the HYPHEN. Further characters can easily be accomodated by re-dimensioning the array BC$, and adding the new characters to the end of the string B$, and adding the new characters to the end of the string B$. New DATA statements also need to be added in the correct order at the end of the present series of DATA statements.

Use of the program is demonstrated by the short program starting at LINE 1000 in the listing.

The subroutine was originally devised as part of a teaching program for young children, and can readily be combined with graphics, and sound effects if desired. There are probably many other uses for such a subroutine, and I would be interested to hear from any readers who find it of use.

```
100 REM**********************************
110 REM** BIG CHARACTERS - WRITTEN BY *
120 REM**  F.E.WOODWARD  8 VIOLET AV **
130 REM**    RAMSGATE KENT CT12 6TH  **
140 REM**********************************
150 REM** USING EXTENDED BASIC ON THE *
160 REM**     SHARP MZ-80K            **
170 REM**********************************
1000 REM** DEMO PROG.USING LARGE CHAR.**
1010 GOSUB 50020
1020 PRINT"E"
1030 PRINT@5,1;"INPUT ANY TEXT UP TO 5 CHARS."
1040 PRINT@7,1;"CAPITALS,NUMBERS,OR HYPHEN."
1050 PRINT
1060 INPUT" > ";TX$
1070 TL=7 : PRINT"E" : GOSUB 52020
1090 PRINT@20,1;"KEY 'R' TO REPEAT, 'E' TO EXIT"
1100 GET GT$ : IF GT$="" GOTO 1100
1110 IF GT$="R" GOTO 1020
1120 IF GT$="E" GOTO 1140
1130 GOTO 1100
1140 END
```

```
50000 REM** INITIALISATION OF LARGE  **
50010 REM**   CHARACTER SUBROUTINE   **
50020 DIM BC$(37,7)
50030 FOR A=1 TO 37
50040 FOR B=1 TO 7
50050 READ BC$(A,B)
50060 NEXT B
50070 NEXT A
50080 DATA " AAA ","A   A","A   A"
50090 DATA "AAAAA","A   A","A   A"
50100 DATA "A   A"
50110 DATA "BBBB ","B   B","B   B"
50120 DATA "BBBB ","B   B","B   B"
50130 DATA "BBBB "
50140 DATA " CCCC","C    ","C    "
50150 DATA "C    ","C    ","C    "
50160 DATA " CCCC"
50170 DATA "DDDD ","D   D","D   D"
50180 DATA "D   D","D   D","D   D"
50190 DATA "DDDD "
50200 DATA "EEEEE","E    ","E    "
50210 DATA "EEEEE","E    ","E    "
50220 DATA "EEEEE"
50230 DATA "FFFFF","F    ","F    "
50240 DATA "FFFF ","F    ","F    "
50250 DATA "F    "
50260 DATA " GGGG","G    ","G    "
50270 DATA "G    ","G GGG","G   G"
50280 DATA " GGG "
50290 DATA "H   H","H   H","H   H"
50300 DATA "HHHHH","H   H","H   H"
50310 DATA "H   H"
50320 DATA " III ","  I  ","  I  "
50330 DATA "  I  ","  I  ","  I  "
50340 DATA " III "
50350 DATA "    J","    J","    J"
50360 DATA "    J","J   J","J   J"
50370 DATA " JJJ "
50380 DATA "K   K","K   K","K  K "
50390 DATA "KKK  ","K  K ","K   K"
50400 DATA "K   K"
50410 DATA "L    ","L    ","L    "
50420 DATA "L    ","L    ","L    "
50430 DATA "LLLLL"
50440 DATA "M   M","MM MM","M M M"
50450 DATA "M   M","M   M","M   M"
50460 DATA "M   M"
50470 DATA "N   N","N   N","NN  N"
50480 DATA "N N N","N  NN","N   N"
50490 DATA "N   N"
50500 DATA " OOO ","O   O","O   O"
50510 DATA "O   O","O   O","O   O"
50520 DATA " OOO "
50530 DATA "PPPP ","P   P","P   P"
50540 DATA "PPPP ","P    ","P    "
50550 DATA "P    "
50560 DATA " QQQ ","Q   Q","Q   Q"
50570 DATA "Q   Q","Q Q Q","Q  Q "
50580 DATA " QQ Q"
```

```
50590 DATA "RRRR ","R   R","R   R"
50600 DATA "RRRR ","R R  ","R  R "
50610 DATA "R   R"
50620 DATA " SSSS","S    ","S    "
50630 DATA " SSS ","    S","    S"
50640 DATA "SSSS "
50650 DATA "TTTTT","  T  ","  T  "
50660 DATA "  T  ","  T  ","  T  "
50670 DATA "  T  "
50680 DATA "U   U","U   U","U   U"
50690 DATA "U   U","U   U","U   U"
50700 DATA " UUU "
50710 DATA "V   V","V   V","V   V"
50720 DATA "V   V","V   V"," V V "
50730 DATA "  V  "
50740 DATA "W   W","W   W","W   W"
50750 DATA "W   W","W W W","WW WW"
50760 DATA "W   W"
50770 DATA "X   X","X   X"," X X "
50780 DATA "  X  "," X X ","X   X"
50790 DATA "X   X"
50800 DATA "Y   Y","Y   Y"," Y Y "
50810 DATA "  Y  ","  Y  ","  Y  "
50820 DATA "  Y  "
50830 DATA "ZZZZZ","    Z","   Z "
50840 DATA "  Z  "," Z   ","Z    "
50850 DATA "ZZZZZ"
50860 DATA "  1  "," 11  ","  1  "
50870 DATA "  1  ","  1  ","  1  "
50880 DATA " 111 "
50890 DATA " 222 ","2   2","    2"
50900 DATA "   2 ","  2  "," 2   "
50910 DATA "22222"
50920 DATA " 333 ","3   3","    3"
50930 DATA "   3 ","    3","3   3"
50940 DATA " 333 "
50950 DATA "4  4 ","4  4 ","4  4 "
50960 DATA "44444","   4 ","   4 "
50970 DATA "   4 "
50980 DATA "55555","5    ","555  "
50990 DATA "    5 ","    5","5   5"
51000 DATA " 555 "
51010 DATA " 666 ","6    ","6    "
51020 DATA "6666 ","6   6","6   6"
51030 DATA " 666 "
51040 DATA "7777 ","   7 ","   7 "
51050 DATA "   7 ","  7  ","  7  "
51060 DATA "7    "
51070 DATA " 888 ","8   8","8   8"
51080 DATA " 888 ","8   8","8   8"
51090 DATA " 888 "
51100 DATA " 999 ","9   9","9   9"
51110 DATA " 9999","    9","    9"
51120 DATA " 999 "
51130 DATA " 000 ","0   0","0   0"
51140 DATA "0   0","0   0","0   0"
51150 DATA " 000 "
51160 DATA "     ","     ","     "
51170 DATA " ■■■ ","     ","     "
51180 DATA "     "
```

```
51190 REM** END OF DATA STATEMENTS   **
51200 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
51210 B$="1234567890-"
51220 M$=A$+B$ : LM=LEN(M$)
51230 RETURN
52000 REM*********************************"
52010 REM** LARGE CHARACTER MAIN SUB.*"
52020 REM** REQ.-- TX$=TEXT AND      **
52030 REM**      TL=TOP LINE OF CHAR.**
52040 REM** SET IN MAIN PROGRAM      **
52050 LT=LEN(TX$) : Y=(40-((5*LT)+2))/2
52060 FOR C=1 TO LT
52070 S$=MID$(TX$,C,1) : P=0 : QP=0
52080 FOR QP=1 TO LM
52090 IF MID$(M$,QP,1)=S$ GOTO 52110
52100 NEXT QP
52110 P=QP
52120 FOR Z=1 TO 7
52130 X=TL+Z
52140 PRINT@X,Y;BC$(P,Z)
52150 NEXT Z
52160 Y=Y+6
52170 NEXT C
52180 RETURN
```

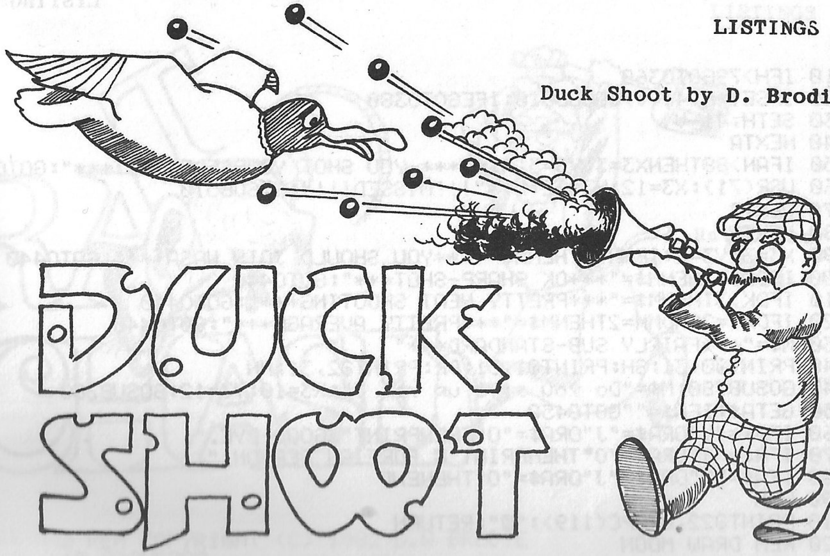
"STORM" by **Peter Chase** (Try this for size!!!)

```
10 PRINT"■                Storm■■■■■■■■■■"
20 PRINT"■■■■ Catch the spaceships and avoid the hail"
30 PRINT "■■■■ A  -  Left";TAB(20);"D  -  Right"
40 PRINT@12,22;"Press a key":GETZ$:IFZ$=""THEN40
100 TEMPO7:LIMIT53236:FORI=53236TO53247:READP:POKEI,P:NEXT:POKE10407,0
110 DATA1,32,3,17,071,211,33,031,211,237,184,201
120 S=0:PRINT"■":P=54067:PRINT@5,22;"Score":PRINT@25,22;"High";H
130 FORI=0TO39:PRINT@I,21;"─":NEXT
140 POKE53288+RND(1)*40,88-111*(RND(1)<.1)
150 USR(53236):IFPEEK(P)=88THEN180
160 IFPEEK(P)=199THENS=S+10:PRINT@5,22;"Score":S:MUSIC"─C0"
170 GETZ$:P=P+((Z$="A")AND(P>54048))-((Z$="D")AND(P<54087)):POKEP,202:GOTO14
180 MUSIC"▁C5":PRINT"■":PRINT@12,12;"GAME OVER":IFH<STHENH=S
190 PRINT@12,22;"Hit SPACE":GETZ$:IFZ$=" "THEN120
200 GOTO190
```

TENTHS OF SECONDS TIMER PROGRAM from **Mr. D. Wiley**

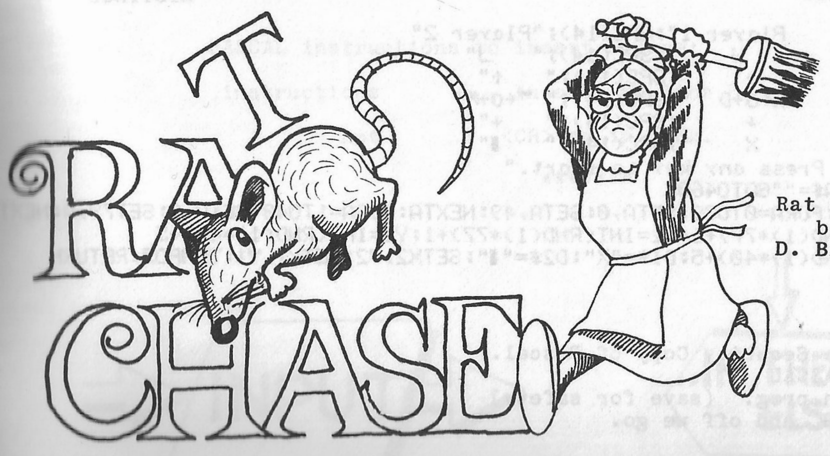
```
10 REM TIME DISPLAY WITH TENTHS OF SECONDS ADDED----D.WILEY NORWICH
15 PRINT"■ H M S T"
20 B=VAL(TI$)*10
30 FORI=1TO10
40 B=B+1
50 PRINTB
60 PRINT"■■"
65 FORA=1TO55:NEXTA
70 NEXTI
80 GOTO10
```

Duck Shoot by D. Brodie



```
0 REM COPYRIGHT (C) 1982 D.W BRODIE
10 DATA53,116,62,58,78,191,67,77,66,86
20 PRINT"▣ ** SHARP DUCK-SHOOT **":TEMPO4:MUSIC"‾C_CC"
30 PRINT"▩▩ You get five rounds of three shots."
40 PRINT"▩▩ Angles must be in the range 17°-90°."
50 PRINT"▩▩ Watch out for the moon!"
60 PRINT"▩▩▩";SPC(7);"** PRESS ANY KEY **"
70 GETA$:IFA$=""GOTO70
80 FORR=1TO5:PRINT"▣":PRINT@21,0;"*** ROUND";R;" ***":FO=INT(RND(1)*50)+30
90 REM DRAW DUCK
100 MUSIC"‾C_CC":X=INT(RND(1)*13)+13
110 PRINT@17,X;"▁▄◢▀▩◣"
120 PRINT@18,X;"─‾▀▩▩▀◢▩▩◣◢◣"
130 PRINT@19,X;"   ▀▩▩▩▩▀"
140 PRINT@20,X;"    ▀▩▩▀"
150 X1=X-INT(RND(1)*10):Y1=INT(RND(1)*6)+1:GOSUB520
160 REM THE GAME
170 FORS=1TO3:PRINT@21,17;"SHOT";S;" ***":GOSUB510
180 PRINT@0,24;"SHOTS =";SH:SH=SH+1
190 PRINT@1,24;"DUCKS =";DK
200 PRINT@2,24;"MOONS =";MN
210 PRINT@22,1;"Angle of elevation";:INPUTAN:GOSUB510
215 FORA=0TO10:POKE4514,A:USR(68):NEXTA:USR(71)
220 IFAN=INT(ABS(AN))ANDAN<91ANDAN>16GOTO240
230 X3=12:Y3=22:M$="!!!MISFIRE!!!":GOSUB780:GOSUB510:GOTO370
240 FORA=1TOFO:V=INT(A*COS((90-AN)/180*π)):H=INT(A*COS(AN/180*π))
250 IF41-V<0GOTO280
260 IFSET(H,41-V)GOSUB610:IFEGOTO380
270 SETH,41-V
280 NEXTA:H1=H:FORA=FOTO1STEP-1:V=INT(A*COS((90-AN)/180*π))
290 H=2*H1-INT(A*COS(AN/180*π))
300 IF41-V<0GOTO340
```

```
310 IFH>79GOTO360
320 IFSET(H,41-V)GOSUB610:IFEGOTO380
330 SETH,41-V
340 NEXTA
350 IFAN>88THENX3=3:Y3=5:M$="*** YOU SHOT YOURSELF TWIT***":GOTO440
360 USR(71):X3=12:Y3=22:M$="!!!MISSED!!!":GOSUB510
370 NEXTS
380 NEXTR
390 X3=3:Y3=5:IFMN>2THENM$="***YOU SHOULD JOIN NASA***":GOTO440
400 IFDK=5THENM$="***OK SHARP-SHOT***":GOTO440
410 IFDK>3THENM$="***PRETTY NEAT SHOOTING***":GOTO440
420 IFDK>=2ANDMN=2THENM$="***PRETTY AVERAGE***":GOTO440
430 M$="***FAIRLY SUB-STANDARD***"
440 PRINT@0,31;SH:PRINT@1,31;DK:PRINT@2,31;MN
445 GOSUB780:M$="Do you give up yet ?":X3=10:Y3=12:GOSUB780
450 GETA$:IFA$=""GOTO450
460 IFA$="Y"ORA$="J"ORA$="O"THENPRINT"GOOD-BYE.";
470 IFA$="J"ORA$="O"THENPRINT" FOREIGN PERSON."
480 IFA$="Y"ORA$="J"ORA$="O"THENEND
490 RUN
510 PRINT@22,0;SPC(119);"":RETURN
520 REM DRAW MOON
530 PRINT@Y1,X1;"     "
540 PRINT@Y1+1,X1;"   "
550 PRINT@Y1+2,X1;"    "
560 PRINT@Y1+3,X1;"    "
570 PRINT@Y1+4,X1;"    "
580 PRINT@Y1+5,X1;"   "
590 PRINT@Y1+6,X1;"   "
600 PRINT@Y1+7,X1;"   ":RETURN
610 REM WHAT 'AVE U 'IT ?
620 E=0:X2=INT(H/2):Y2=INT((41-V)/2)
630 CH=PEEK(53248+X2+40*Y2)
640 IFCH>240THENRETURN
650 RESTORE:FORF=1TO10:READC1:IFCH=C1GOTO670
660 NEXTF:RETURN
670 IFX2>X+10THENM$="RUDE!":X3=X+3:Y3=16:GOSUB780:RETURN
680 E=9999:IFCH<>67GOTO720
690 IFY2>Y1+7GOTO720
700 FORY3=Y1TO13:Y1=Y3:PRINT@Y3-1,X1;"    ":GOSUB520:NEXTY3
710 X3=11:Y3=10:M$="YOU SHOT THE MOON!":MN=MN+1:GOTO780
720 DK=DK+1:Y3=16:ONINT(RND(1)*5)+1GOTO730,740,750,760,770
730 M$="*DEAD*":X3=X+3:GOTO780
740 M$="*R.I.P*":X3=X+2:GOTO780
750 M$="*QUACK*":X3=X+2:GOTO780
760 M$="*OUCH*":X3=X+3:GOTO780
770 M$="*DEAD-DUCK*":X3=X
780 REM DISPLAY MESSAGES
790 FORO=0TO9:IFO/2=INT(O/2)THENPRINT@Y3,X3;SPC(LEN(M$)):GOTO810
800 USR(62):PRINT@Y3,X3;M$
810 FORP=0TO99:NEXTP,O:FORP=0TO599:NEXTP:RETURN
```



Rat Chase

by
D. Brodie

```
0 REM COPYRIGHT (C) 1982 D.W BRODIE
10 GOSUB290
20 POKE17828,0:GETA$
30 IFA$="X"THENY1=Y1+1:GOTO80
40 IFA$="W"THENY1=Y1-1:GOTO80
50 IFA$="A"THENX1=X1-1:GOTO80
60 IFA$="D"THENX1=X1+1:GOTO80
70 A$=D1$:GOTO30
80 D1$=A$:IFSET(X1,Y1)THENPL=2:GOTO170
90 SETX1,Y1:MUSIC"_A0":POKE17828,0:GETA$
100 IFA$="┘"THENY2=Y2-1:GOTO150
110 IFA$="▐"THENY2=Y2+1:GOTO150
120 IFA$="‾"THENX2=X2-1:GOTO150
130 IFA$="⁄"THENX2=X2+1:GOTO150
140 A$=D2$:GOTO100
150 D2$=A$:IFSET(X2,Y2)THENPL=1:GOTO170
160 SETX2,Y2:MUSIC"A0":GOTO20
170 ONPLGOTO180,190
180 TEMPO1:MUSIC"_A0RA0F0D0B0":GOTO200
190 TEMPO1:MUSIC"A0R_A0C0E0G0"
200 PRINT@10,12;"▓▒▒▒▒▒▒▒▒▒▒▒▒▓":PRINT@14,12;"▓▒▒▒▒▒▒▒▒▒▒▒▒▓"
210 PRINT@11,12;"▒            ▒":PRINT@13,12;"▒            ▒"
220 PRINT@12,12;"▓            ▓":FORA=0TO3
230 IFA/2=INT(A/2)THENA$="WINNER IS":GOTO250
240 A$="PLAYER  "+STR$(PL)
250 FORB=1TOLEN(A$):PRINT@12,13+B;MID$(A$,B,1):NEXTB,A
260 FORA=0TO1999:NEXTA:GETA$:IFPEEK(17828)=0THENRUN
270 GOSUB470:RUN30
280 END
290 REM INSTRUCTIONS
300 PRINT"▓ Rat Chase.▓"
310 PRINT"  This is a game for two players. Each"
320 PRINT" player is in control of a rat and the"
330 PRINT" object of the game is to trap your"
340 PRINT" opponents rat."
350 PRINT"▓  If your rat runs over either its own"
360 PRINT" path or that of your opponents; or if"
370 PRINT" it runs into the wall then you lose!"
380 PRINT"▓▓ Controls are as follow:"
```

```
390 PRINT"■      Player 1";SPC(14);"Player 2"
400 PRINT"■        W     ";SPC(14);"    _|"
410 PRINT"         ↑     ";SPC(14);"    ↑"
420 PRINT"       A←O→D   ";SPC(14);" ~←O→/"
430 PRINT"         ↓     ";SPC(14);"    ↓"
440 PRINT"         X     ";SPC(14);"    ▮"
450 PRINT"■  Press any key to start."
460 GETA$:IFA$=""GOTO460
470 PRINT"■":FORA=0TO79:SETA,0:SETA,49:NEXTA:FORA=1TO48:SET0,A:SET79,A:NEXTA
480 X1=INT(RND(1)*77)+1:X2=INT(RND(1)*77)+1:Y1=INT(RND(1)*40)+5
490 Y2=INT(RND(1)*40)+5:D1$="X":D2$="▮":SETX2,Y2:SETX1,Y1:TEMPO5:RETURN
```

To make Security Copy of Pascal.
Load PASCAL
Type in prog. (save for safety)
Then 'G' and off we go.

```
%.........PASCAL COPY.........%
VAR J,L,I,A,Z:INTEGER;
   B:CHAR;
PROCEDURE ROUTINE;
  BEGIN
     I:=L DIV 256;
     J:=L-(I*256);
  END;
BEGIN
    A:=1;
    POKE(CHR(A),4336);
    POKE(CHR(80),4337);
    POKE(CHR(65),4338);
    POKE(CHR(83),4339);
    POKE(CHR(67),4340);
    POKE(CHR(65),4341);
    POKE(CHR(76),4342);                    "PASCAL COPY"
    POKE(CHR(32),4343);
    POKE(CHR(67),4344);
    POKE(CHR(79),4345);
    POKE(CHR(80),4346);
    POKE(CHR(89),4347);
    POKE(CHR(13),4348);
    L:=16384;
   ROUTINE;
    POKE(CHR(J),4354);                      PROG. LENGTH
    POKE(CHR(I),4357);
    L:=4608;
   ROUTINE;
    POKE(CHR(J),4356);                      START ADDRESS
    POKE(CHR(I),4357);
    L:=4608;
   ROUTINE;
    POKE(CHR(J),4358);                      EXECUTE ADDRESS
    POKE(CHR(I),4359);
    WRITELN("POSITION TAPE");
    WRITELN("ANY KEY TO CONT");
    B:=KEY;
    WHILE ORD(B)=0 DO B:=KEY;
    CALL(33)Z,Z;
    CALL(36)Z,Z;
END.
```

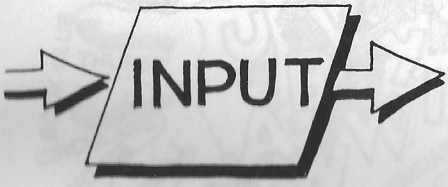
Two PASCAL tips from
D. Willey .

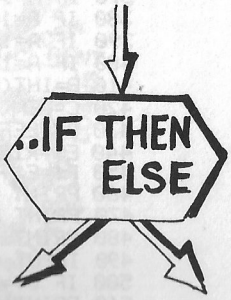
PASCAL instructions to insert a line:

instructions 5 - inserts at line 5

not 5 <CR> as in book.

****



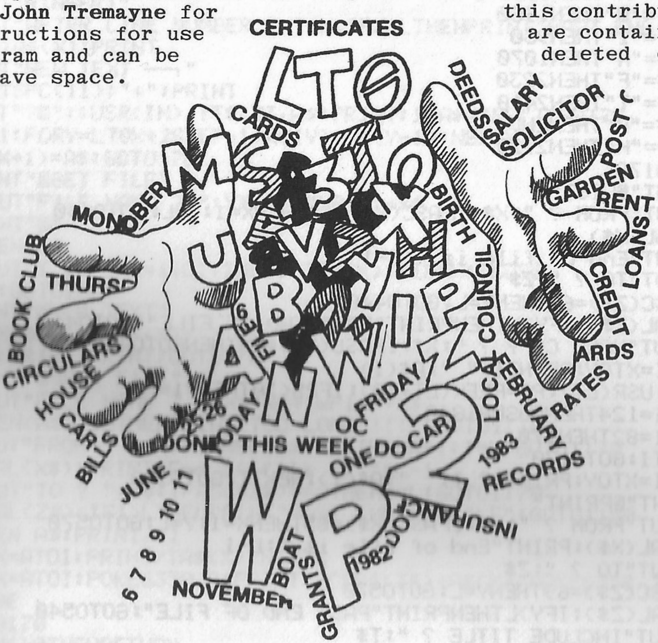



```
10 REMARKABLE PROGRAM BY S.ZATMAN
20 REM Ase 10
30 PRINT "■"
40 PRINT "The object of this game is to"
50 PRINT "guess the word which has it's"
60 PRINT "letters jumbled up.You have 10"
70 PRINT "tries.The score and the number"
80 PRINT "of questions is at the top of"
90 PRINT "the screen.You may get nulls."
100 PRINT "Press any key to start the game.■"
110 GET J$
120 IF J$=""THENGOTO110
130 PRINT "■"
140 RESTORE
150 READ A$
160 READ AA$
170 READ AB$
180 READ AC$
190 READ AD$
200 READ AE$
210 READ AF$
220 READ AG$
230 READ AH$
240 READ AI$
250 READ AJ$
260 READ AK$
270 C=LEN(B$)
280 A=INT(RND(1)*12)+1
290 IF A=1 THEN B$=A$
300 IF A=2 THEN B$=AA$
310 IF A=3 THEN B$=AB$
320 IF A=4 THEN B$=AC$
330 IF A=5 THEN B$=AD$
340 IF A=6 THEN B$=AE$
```

```
350 IF A=7 THEN B$=AF$
360 IF A=8 THEN B$=AG$
370 IF A=9 THEN B$=AH$
380 IF A=10 THEN B$=AI$
390 IF A=11 THEN B$=AJ$
400 IF A=12 THEN B$=AK$
410 B=INT(RND(1)*LEN(B$))+1
420 C=LEN(B$)
430 D$=LEFT$(B$,C/2)
440 E$=RIGHT$(B$,C/2+1)
450 F$=E$+D$
460 PRINT E;"/";F
470 PRINT F$
480 PRINT
490 INPUT C$
500 IF C$=B$ THEN GOTO 560
510 PRINT TAB(15);"Wrong !"
520 F=F+1
530 IF F=10 THEN PRINT E;" Right":GOTO 660
540 PRINT B$
550 MUSIC"_C9R0_C9R0_C9":GOTO 130
560 PRINT TAB(15);"Right !":E=E+1
570 F=F+1
580 IF F=10 THEN PRINT E;" Right":GOTO 660
590 MUSIC"^G0^A0^B0^G0^A0^B0"
600 GOTO 130
610 DATA "CORRECT","THERMONUCLEAR","ATOMIC"
620 DATA "TRIDENT D5","SPASMODIC","IDIOTIC"
630 DATA "FISSION","FUSION","CELSIUS"
640 DATA "KELVING","IGNORAMUS","SUPERB"
650 END
660 G$="You are a "
670 H$="You are an "
680 IF E<5 THEN PRINT H$;"ignoramus.":GOTO 710
690 IF E<9 THEN PRINT "You are O.K.":GOTO 710
700 IF E=9 THEN PRINT "You are clever."
710 IF E=10 THEN PRINT "You are SUPERB."
720 PRINT "Do you want another go ?"
730 GET I$
740 IF I$=""THEN GOTO 730
750 IF I$="Y"THEN RUN
760 PRINT "■"
770 PRINT "■■■■■■■■■■■"
780 PRINT TAB(15);"THANKYOU"
790 GOTO 760
800 END
```

WORD PROCESSOR TWO (WP2)

An Upgraded version of our original WP1 program. Our thanks to Mr. John Tremayne for this contribution. All instructions for use are contained in the program and can be deleted or omitted to save space.




```
10 CLR:PRINT"▣"
20 GOSUB1800
30 TT$="!#$%&'()π+┌┤┬▒ ┬┐ ¯*↑:?@\][←×♠♥♦♣●○□┐▀▄─|∧~▀▄|─×┘ ▙▞▜▓+→◢ ▙|_"
40 TR$="≡||#▧▨◡⌒β ⌒ / /\\ √ üpoiuytrewqasdfghjklöä\ ◡ /\\´~`üÄömnbvcxz"◔╫+¥"
50 TP$="1234567890-┘├▒┴|¯=POIUYTREWQASDFGHJKL;£-|▜▒▙|-/.,MNBVCXZ× ▙ L"
60 TT$=TT$+TR$+TP$+CHR$(34)+" ":GOSUB2470:CH=52856:TT=240
70 PRINT"▣";
80 FORX=1TO40:PRINT"▒";:NEXT
90 PRINT"▒         WORD PROCESSOR TWO           ▒";
100 FORX=1TO40:PRINT"▒";:NEXTX:PRINT
110 DIMA$(255):L=1:LH=0
120 PRINT"▣▣▣▣Do you need instructions ? Y/N"
130 GET I$:IFI$="Y" THEN2720
140 IFI$="N"THEN160
150 GOTO130
160 PRINT"▣The symbol × (bottom right hand key)":GOTO2040
170 PRINT"?▣";:USR(IN),TT$,TT,A$:PRINT:IFLEN(A$)<1THENA$(L)=A$:L=L+1:GOTO170
180 IFASC(A$)=124THEN200
190 A$(L)=A$:L=L+1:GOTO170
200 C$=MID$(A$,2,1)
210 IFC$="L"THEN370
220 IFC$="G"THEN1010
230 IFC$="P"THEN510
240 IFC$="S"THEN1100
250 IFC$="R"THEN790
```

```
260 IFC$="E"THEN2030
270 IFC$="D"THEN860
280 IFC$="A"THEN1280
290 IFC$="W"THEN1960
300 IFC$="I"THEN930
310 IFC$="N"THEN1070
320 IFC$="F"THEN2230
330 IFC$="T"THEN2410
340 IFC$="B"THENEND
350 IFC$="H"THEN2720
360 GOTO170
370 PRINT"■LIST"
380 INPUT"FROM ? ";X$:IFASC(X$)=65THENX=1:Y=L:GOTO440
390 X=VAL(X$)
400 PRINT"End of file is ";L-1
410 INPUT"TO ? ";Z$
420 IFASC(Z$)=69THENY=L:GOTO440
430 Y=VAL(Z$):IFY>LTHENPRINT"PAST END OF FILE":GOTO410
440 INPUT"HARD COPY ? ";X$:IFASC(X$)=89THENGOTO480
450 FORI=XTOY:PRINTI;" ";A$(I)
460 P=0:USR(LT):PS=PEEK(LT+10):IFPS<>0THENP1=PS
470 IFP1=124THENGOSUB1840
480 IFP1=82THEN170
490 NEXTI:GOTO170
500 FORI=XTOY:PRINT#P I;" ";A$(I):NEXTI:GOTO170
510 PRINT"■PRINT"
520 INPUT"FROM ? ";X$:IFASC(X$)=65THENX=1:Y=L:GOTO570
530 X=VAL(X$):PRINT"End of file is ";L-1
540 INPUT"TO ? ";Z$
550 IFASC(Z$)=69THENY=L:GOTO570
560 Y=VAL(Z$):IFY>LTHENPRINT"PAST END OF FILE":GOTO540
570 INPUT"INCLUDE TITLE ? ";T$
580 T=0:IFASC(T$)=89THENT=1
590 INPUT"HARD COPY ? ";X$:IFASC(X$)=89THEN680
600 IFT<>1THEN620
610 GOSUB1220:PRINTI1$:PRINTI2$:PRINTI3$:PRINTI4$
620 PRINT:PRINT:PRINT
630 FORI=XTOY:PRINTA$(I)
640 P1=0:USR(LT):PS=PEEK(LT+10):IFPS<>0THENP1=PS
650 IFP1=124THENGOSUB1840
660 IFP1=82THEN170
670 NEXTI:PRINT:GOTO170
680 INPUT"No. of copies ? ";NC$:IFVAL(NC$)=0THENNC=1:RF=1:JF=0:GOTO
690 NC=VAL(NC$):RF=0
700 FOR P1=1TONC:IFT<>1THEN730
710 PRINT#P:PRINT#P:GOSUB1220
720 PRINT#PI1$:PRINT#PI2$:PRINT#PI3$:PRINT#P:PRINT#PI4$:PRINT#P
730 FOR I=XTOY:IF(RF=1)*(LEFT$(A$(I),1)="*")THENGOSUB1860
740 PRINT#PA$(I):IFRF=2THENA$(I)=A1$:RF=1
750 IFEF=1THENRF=0
760 NEXTI
770 PRINT#P"■":NEXTP1:IFRF<>0THENRF=1:GOTO700
780 PRINT#P"■":GOTO170
790 PRINT"■REPLACE"
800 INPUT"LINE NUMBER ? ";I:IFI>LTHEN170
810 PRINT"REPLACE TEXT ——┐"
820 PRINTSPC(15);"↓":PRINT
830 PRINT@0,5;A$(I):PRINT
840 USR(IN),TT$,TT,A$:PRINT:IFMID$(A$,1,1)="N"THEN170
850 A$(I)=A$:GOTO170
860 PRINT"■DELETE"
870 INPUT"LINE NUMBER ? ";X:IFX>LTHENPRINT"PAST END OF FILE":GOTO87
```

```
800 PRINT"DELETE TEXT ──┐"
890 PRINTSPC(14);"↓":PRINT
900 PRINTA$(X):PRINT
910 INPUT"  OK ? ";A$:IFASC(A$)<>89THEN170
920 L=L-1:FORY=XTOL:A$(Y)=A$(Y+1):NEXTY:GOTO170
930 PRINT"■INSERT"
940 INPUT"AFTER LINE NUMBER ? ";X:IFX>LTHENPRINT"PAST END OF FILE":GOTO940
950 PRINTA$(X):PRINT
960 PRINT"NEW TEXT ──┐"
970 PRINTSPC(11);"↓":PRINT
980 PRINT"?■";:USR(IN),TT$,TT,A$:PRINT:IFA$="NO"THEN170
990 L=L+1:FORY=LTOX+2STEP-1:A$(Y)=A$(Y-1):NEXTY
1000 A$(X+1)=A$:GOTO170
1010 PRINT"■GET FILE"
1020 INPUT"FILE NAME ? ";X$:A$="WP1."+X$
1030 PRINT"■"
1040 ROPEN A$
1050 INPUT/T I:FORX=1TOI:INPUT/TA$(X):NEXTX:CLOSE
1060 L=I:GOTO2030
1070 PRINT"■NEW TEXT"
1080 FORX=0TOL+1:A$(X)="":NEXTX
1090 L=1:PRINT:PRINT:GOTO170
1100 PRINT"■SAVE FILE"
1110 INPUT"FILE NAME ? ";X$:A$="WP1."+X$
1120 IFLEN(A$)>15THENPRINT"TOO LONG !!!":GOTO1110
1130 INPUT"FROM ? ";X$:IFASC(X$)=65THENA=1:I=L:GOTO1170
1140 A=VAL(X$):PRINT"End of file is "L-1
1150 INPUT"TO ? ";Z$:IFASC(Z$)=69THENI=L:GOTO1170
1160 I=VAL(Z$):IFI>LTHENPRINT"PAST END OF FILE":GOTO1150
1170 WOPEN A$:PRINT/TI
1180 FORX=ATOI:PRINT/TA$(X):NEXTX
1190 FORX=ATOI:POKE6350,0:PRINT/TCHR$(34)+A$(X):POKE6350,34:NEXTX
1200 CLOSE
1210 GOTO170
1220 IFLH<>0THENRETURN
1230 I1$="                                                    Mylor, 12 Forge Lane"
1240 I2$="                                                      Bassaleg,Newport,"
1250 I3$="                                                        Gwent. NP1 9NF."
1260 I4$="■■                                                    Tel. Rhiwderin 3371"
1270 RETURN
1280 PRINT"■FORMAT TEXT"
1290 INPUT"CHARS./LINE (MAX.132) ?":CL
1300 IF(CL>132)+(CL<0)THEN1290
1310 INPUT"FROM ?";X$:IFASC(X$)=65THENX=1:Y=L:GOTO1360
1320 X=VAL(X$)
1330 INPUT" TO ?";Z$
1340 IFASC(Z$)=69THENY=L:GOTO1360
1350 Y=VAL(Z$):IFY>LTHENPRINT" PAST END OF FILE":GOTO1330
1360 IFX>YTHEN170
1370 GOSUB1720
1380 IF LEN(A$(X))=CLTHENGOSUB1760:X=X+1:GOTO1360
1390 IF LEN(A$(X))<CLTHEN1540
1400 FORI=LEN(A$(X))TO1STEP-1
1410 IFMID$(A$(X),I,1)="Φ"THENNF=2
1420 IF(MID$(A$(X),I,1)=" ")*(I=<CL)THEN1460
1430 NEXTI:PRINT"LINE ";X;" TOO LONG - NO SPACES."
1440 IFNF=2THENNF=1
1450 GOSUB1760:PRINTA$(X):GOTO170
1460 A$=RIGHT$(A$(X),(LEN(A$(X))-I)):A$(X)=LEFT$(A$(X),I)
1470 IF(LEFT$(A$(X+1),2)="  ")+(X=Y)THENGOSUB1510:GOTO1500
1480 A$(X+1)=A$+""+A$(X+1):GOSUB1760
1490 X=X+1:GOTO1360
1500 A$(X+1)=A$:GOSUB1760:X=X+1:GOTO1360
1510 L=L+1:Y=Y+1
1520 FORI=LTOX+2 STEP-1:A$(I)=A$(I-1):NEXTI:IFFL=1THENFX=1:FL=0
1530 RETURN
```

```
1540 IF(LEFT$(A$(X+1),2)="  ")+(X=Y)THEN1560
1550 GOTO1570
1560 GOSUB1760:X=X+1:GOTO1360
1570 IFLEN(A$(X))+LEN(A$(X+1))<=CLTHENA$(X)=A$(X)+""+A$(X+1):GOTO1680
1580 FORI=1TOLEN(A$(X+1))
1590 IF(MID$(A$(X+1),I,1)="¤")THENA$(X+1)=RIGHT$(A$(X+1),LEN(A$(X+1))-1):FL=2
1600 IF(MID$(A$(X+1),I,1)<>" ")THENNEXTI
1610 IFLEN(A$(X))+I>CLTHEN1630
1620 GOTO1650
1630 IFFL=2THENFL=1
1640 GOSUB1760:X=X+1:GOTO1360
1650 A$(X)=A$(X)+""+LEFT$(A$(X+1),I)
1660 A$(X+1)=RIGHT$(A$(X+1),(LEN(A$(X+1))-I)):GOSUB1760:GOTO1360
1670 NEXTI:GOSUB1760:X=X+1:GOTO1360
1680 L=L-1:Y=Y-1:IFFL=1THENFL=2
1690 GOSUB1760
1700 FORI=X+1TOL:A$(I)=A$(I+1):NEXTI
1710 GOTO1360
1720 NF=0:FL=0:FX=0
1730 IFLEFT$(A$(X),1)="¤"THENA$(X)=RIGHT$(A$(X),LEN(A$(X))-1):NF=1
1740 IFLEFT$(A$(X+1),1)="¤"THENA$(X+1)=RIGHT$(A$(X+1),LEN(A$(X+1))-1):FL=1
1750 RETURN
1760 IF(NF=1)+(FL=2)THENA$(X)="¤"+""+A$(X)
1770 IF(NF=2)+(FL=1)THENA$(X+1)="¤"+""+A$(X+1)
1780 IF FX=1THENA$(X+2)="¤"+""+A$(X+2)
1790 RETURN
1800 LT=52470:LIMITLT
1810 FORI=0TO7:READMC:POKELT+I,MC:NEXTI
1820 DATA205,27,0,50,0,205,201,0
1830 RETURN
1840 GETPS$:IFPS$=""THEN1840
1850 P1=ASC(PS$):RETURN
1860 EF=0:IFJF=0THENPRINT" INSERT NAMEFILE TAPE.":ROPEN:JF=1
1870 INPUT#TNM$
1880 IFRIGHT$(NM$,1)="¤"THENNM$=LEFT$(NM$,LEN(NM$)-1):EF=1
1890 A1$=A$(I):RF=2
1900 A$(I)=RIGHT$(A$(I),LEN(A$(I))-1)
1910 FORJ=1TOLEN(A$(I))
1920 IFMID$(A$(I),J,1)="¤"THEN1940
1930 NEXTJ:PRINT"¤NOT FOUND IN LINE ";I:RF=0:A$(I)=A1$:CLOSE:RETURN
1940 A$(I)=LEFT$(A$(I),J-1)+NM$+RIGHT$(A$(I),LEN(A$(I))-J)
1950 RETURN
1960 PRINT"INSERT TAPE AND ENTER NAMES AS PROMPTED"
1970 PRINT"TERMINATE LAST NAME WITH '¤'":PRINT
1980 WOPEN "NAMEFILE"
1990 INPUT"ENTER NAME ";NM$
2000 PRINT#TNM$
2010 IFRIGHT$(NM$,1)="¤"THENCLOSE:GOTO170
2020 GOTO1990
2030 PRINT"■The symbol ↑ (bottom right hand key)
2040 PRINT"followed by a letter will select the
2050 PRINT"Edit commands.  They are :-
2060 PRINT"■  E =  List edit commands
2070 PRINT"   L =  List text
2080 PRINT"   P =  Print"
2090 PRINT"   I =  Insert a line"
2100 PRINT"   R =  Replace a line"
2110 PRINT"   S =  Save text on tape
2120 PRINT"   G =  Get text from tape"
2130 PRINT"   D =  Delete a line"
2140 PRINT"   N =  Start a new text"
2150 PRINT"   F =  Find and replace text"
2160 PRINT"   H =  Instructions
2170 PRINT"   T =  Input new title"
2180 PRINT"   A =  Format text
2190 PRINT"   W =  Write name file to tape"
```

```
2200 PRINT"  B =  Exit to basic"
2210 PRINT"@Enter text (or use edit command option) NOW"
2220 GOTO170
2230 PRINT"@FIND"
2240 PRINT"OLD TEXT ? ";:USR(IN),TT$,TT,O$:PRINT:LO=LEN(O$)
2250 PRINT"NEW TEXT ? ";:USR(IN),TT$,TT,N$:PRINT:LN=LEN(N$)
2260 INPUT"FROM ? ";X$:IFASC(X$)=65THENX=1:Y=L:GOTO2310
2270 X=VAL(X$)
2280 INPUT"TO ? ";Z$
2290 IFASC(Z$)=69THENY=L:GOTO2310
2300 Y=VAL(Z$):IFY>LTHENPRINT"PAST END OF FILE":GOTO2280
2310 RL=0:INPUT"REPLACE ? ";X$:IFASC(X$)=89THENRL=1
2320 FORM=XTOY:LS=LEN(A$(M)):LC=0
2330 FORN=1TOLO:P=LC+N:IFMID$(O$,N,1)=MID$(A$(M),P,1)THEN2360
2340 LC=LC+1:IFLC+LO<=LSTHEN2330
2350 NEXTM:GOTO170
2360 NEXTN
2370 A$=MID$(A$(M),1,LC)+N$+MID$(A$(M),LC+LO+1,LS-LC-LO)
2380 IFLEN(A$)>79THENPRINT"LINE ";M;" TOO LONG":GOTO2350
2390 IFRL=1THENA$(M)=A$
2400 PRINTM;" ";A$(M):LC=LC+1:GOTO2330
2410 PRINT"@TITLE":LH=1
2420 PRINT"LINE 1 ? ";:USR(IN),TT$,TT,I1$:PRINT
2430 PRINT"LINE 2 ? ";:USR(IN),TT$,TT,I2$:PRINT
2440 PRINT"LINE 3 ? ";:USR(IN),TT$,TT,I3$:PRINT
2450 PRINT"LINE 4 ? ";:USR(IN),TT$,TT,I4$:PRINT
2460 GOTO170
2470 DATA 229,197,1,255,0,33,160,206,62,32,205,217,15,193,225,17,160,206
2480 DATA 237,83,12,17,205,151,22,44,205,175,34,205,15,24,205,1,24,213
2490 DATA 197,205,154,22,44,205,169,25,205,154,22,44,34,1,72,225,34,10
2500 DATA 17,225,67,74,237,91,12,17,213,205,179,9,205,206,11,254,102,202
2510 DATA 85,206,254,96,40,61,254,99,40,79,254,98,40,84,254,20,40,91
2520 DATA 254,19,40,120,254,18,202,251,205,254,17,202,36,206,197,237,75,10
2530 DATA 17,229,237,177,225,193,32,203,18,19,12,205,18,0,16,195,205,179
2540 DATA 9,205,206,11,254,102,202,85,206,254,96,32,241,13,12,40,176,62
2550 DATA 20,205,18,0,205,12,0,62,20,205,18,0,13,4,27,24,158,62
2560 DATA 4,50,3,224,62,1,24,6,62,5,50,3,224,175,50,112,17,24
2570 DATA 138,213,197,237,75,12,17,235,237,66,235,193,209,218,83,205,202,83
2580 DATA 205,27,4,13,229,33,83,205,227,245,197,213,229,195,174,14,12,5
2590 DATA 40,103,19,229,33,83,205,227,245,197,213,229,195,144,14,229,175,33
2600 DATA 200,206,237,82,48,28,42,113,17,124,183,40,21,197,1,40,0,37
2610 DATA 34,113,17,235,237,66,235,193,62,40,128,71,121,214,40,79,225,195
2620 DATA 83,205,120,254,40,218,83,205,214,40,71,121,198,40,79,197,1,40
2630 DATA 0,235,9,235,193,229,42,113,17,124,254,24,40,8,36,34,113,17
2640 DATA 225,195,83,205,33,83,205,227,245,197,213,229,195,50,14,175,71,197
2650 DATA 42,1,72,205,19,38,205,54,36,205,41,24,34,1,72,205,123,25
2660 DATA 193,225,205,223,24,205,118,25,205,106,27,157,19,201,33,200,208,1
2670 DATA 240,0,17,160,206,213,197,237,176,193,225,126,205,206,11,119,35,16
2680 DATA 248,33,0,5,34,113,17,195,35,205,0-0,0,0,0,0,0,0
2690 IN=52500
2700 FORX=52500TO52895:READZ:POKEX,Z
2710 NEXT:RETURN
```

```
2720 PRINT"■■■■■■■■■■■■*INSTRUCTIONS*
2730 PRINT"■■Most of the Edit Commands are self
2740 PRINT"explanatory, but note the following:-
2750 PRINT"■■■■1.  In answer to the prompt 'FROM?',
2760 PRINT"given by several of the edit commands,
2770 PRINT"the letter A will give the whole file;
2780 PRINT"in answer to the prompt 'TO' the letter
2790 PRINT"E will write to the end of file.
2800 PRINT"■■■■2.  The edit option A (Format Text)
2810 PRINT"allows a format of up to 132 characters
2820 PRINT"/line to the printer.  To preserve
2830 PRINT"paragraphing, type TWO OR MORE spaces
2840 PRINT"start of the first line of paragraph.
2850 PRINT"■■■Hit any key to continue."
2860 GETI$:IFI$=""THEN2860
2870 PRINT"■■■3.  The option W allows the creation
2880 PRINT"of a name file for use when sending the
2890 PRINT"same letter to a number of people. You
2900 PRINT"will be prompted on how to proceed.
2910 PRINT"■■■4.  To prepare for using the name
2920 PRINT"file, enter, when writing the text, the
2930 PRINT"symbol * in the place where you wish
2940 PRINT "the names to appear. The same symbol *
2950 PRINT"must be entered as the first character
2960 PRINT"of any line in which the name is to be
2970 PRINT"printed from the name file.
2980 PRINT"■■5.To use the name file start the Print
2990 PRINT"routine, and, in answer to the prompt
3000 PRINT"  'How many copies ?', enter  F.  Again
3010 PRINT"you will be prompted on how to proceed.
3020 PRINT"■■Hit any key to continue.
3030 GETI$:IFI$=""THEN3030
3040 PRINT"■■■6.  One line of text can contain up
3050 PRINT"to 240 characters before you need to
3060 PRINT"hit CR.
3070 PRINT"■■7.  All Edit Commands may be used, or
3080 PRINT"text entered, whenever the prompt '?'
3090 PRINT"is flashing.
3100 PRINT"■■8.  The symbol * can also be used to
3110 PRINT"interrupt listing or printing.  It can
3120 PRINT"be restarted by either * or CR.
3130 PRINT"■■9.  The option T allows you to enter
3140 PRINT"a postal address as the title. Four
3150 PRINT"lines of text have been allocated.
3160 PRINT"■■Hit any key to continue.
3170 GET I$:IFI$=""THEN3170
3180 PRINT"■":GOTO2030
```

Many thanks to **N. Alam and K. Syed** for their interesting MZ-80K "SOUNDS" program. Keys A-U are used to produce various sounds from a sexy whistle (H) to a telephone bell!!

```
1 PRINT"E":TEMPO5:GETA$:A$=CHR$(PEEK(17828)):IFA$=""THENUSR(71)
2 ONASC(A$)-64GOSUB4,5,6,7,8,9,11,18,10,24,32,33,34,35,38,41,42,43,44,46,47
3 GOTO1
4 FORJ=1TO2:FORI=1TO10:POKE4514,I:POKE4513,10-I:USR(68):NEXTI,J:RETURN
5 FORI=25TO50STEP1:POKE4513,I:POKE4514,51-I:USR(68):NEXTI:RETURN
6 FORI=1TO2:POKE4513,I:FORJ=10TO1STEP-1:POKE4514,J:USR(68):NEXT:NEXT:RETURN
7 FORI=1TO255:POKE4514,I:USR(68):NEXT:RETURN
8 FORI=1TO3:MUSIC"R5^A2":NEXT:MUSIC"R4^A6":RETURN
9 MUSIC"R0DAGFEDC":RETURN
10 FORI=1TO20STEP10:POKE4513,I:FORJ=1TO10:POKE4514,J:USR(68):NEXTJ,I:RETURN
11 REM
12 FORA=10TO1STEP-1
13 POKE4514,A
14 FORB=0TO255STEPA
15 POKE4513,B:USR(68)
16 NEXTB,A:USR(71)
17 RETURN
18 FORA=255TO0STEP-12
19 POKE4514,1:POKE4513,A
20 USR(68):NEXTA:MUSIC"R2"
21 FORA=-255TO255STEP8
22 B=ABS(A):POKE4514,1:POKE4513,B
23 USR(68):NEXTA:USR(71):RETURN
24 FORB=1TO2
25 POKE4514,1:FORC=1TO15
26 POKE4513,150:USR(68)
27 FORD=1TO6:NEXTD
28 POKE4513,255:USR(68)
29 FORE=1TO6:NEXTE
30 NEXTC:MUSIC"R3"
31 NEXTB:MUSIC"R9":RETURN
32 MUSIC"R0CDEFGAB":RETURN
33 FORT=1TO4:FORH=1TO15:POKE4514,H:USR(68):NEXTH,T:RETURN
34 FORI=0TO100:POKE4514,255*RND(1)+1:USR(68):NEXT:RETURN
35 FORD=1TO2:FORI=1TO10:POKE4514,I:USR(68):NEXT
36 FORI=10TO1:POKE4514,I:USR(68):NEXTI,D
37 RETURN
38 IFA>100THENM=-1:A=100
39 IFA=0THENM=1:A=1
40 A=A+M:POKE4514,A:USR(68):RETURN
41 MUSIC""C1A1R4A1#A1^C^AR1^A3^F3R4":RETURN
42 USR(62):MUSIC"R3":RETURN
43 TEMPO3:MUSIC"R0C6R2C4R1C4R1C6R1#D4R1D4R1D4R1C4R1C4_B4R1C6":RETURN
44 TEMPO3: MUSIC"D1#D1E1^C2E1^C2E1^C4^C1^C1^D1^#D1^E1^C1^D1^E1^E1B1^D1^C4"
45 RETURN
46 TEMPO7:MUSIC"R0C6R2C4R1C4R1C6R1#D4R1D4R1D4R1C4R1C4_B4R1C6":RETURN
47 TEMPO2:MUSIC"G1^C1^":RETURN
```

Three BASIC Games from Mr. M. Severin

HIGHER OR LOWER

```
20 REM ** HIGHER OR LOWER **
21 REM ** BY M.SEVERIN 82 **
22 REM
25 GOSUB57
26 TEMPO5:CLR:DIMA(4,13),S$(4),N$(13):DEF FNA(X)=INT(RND(1)*X)+1
27 S$(1)="♥":S$(2)="♣":S$(3)="♦":S$(4)="♠":FORA=1TO13:READN$(A):NEXT
28 DATA1,2,3,4,5,6,7,8,9,T,J,Q,K
29 A=1:N1=0:S1=0:T=3:FORB=1TO6:PRINT"▒▒▒▒▒▒▒▒▒▒";TAB(B*6-4);"▓▓▓▓▒▒▒▒▒▒▓▓▓▓
30 PRINT"▒▒▒▒▒▒▓▓▓▓▒▒▒▒▒▒▓▓▓▓▒▒▒▒▒▒▓▓▓▓▒▒▒▒▒▒▓▓▓▓▒▒▒▒▒▒▓▓▓▓":USR(62):NE
31 S=FNA(4):N=FNA(13):IFA(S,N)THEN31
32 A(S,N)=1
33 A$=TAB(A*6-4)+"▓▓▓▓▒▒▒▒▒▒▓"+S$(S)+"▓"+S$(S)+"▓▒▒▒▒▒▒▓▓▓▓▒▒▒▒▒▒"
34 A$=A$+"▓▓"+N$(N)+"▓▓▒▒▒▒▒▒▓▓▓▓▒▒▒▒▒▒▓"+S$(S)+"▓"+S$(S)+"▓▒▒▒▒▒▒▓▓▓▓"
35 PRINT"▒▒▒▒▒▒▒▒▒▒";A$:USR(62)
36 ONTGOTO37,40,44
37 IFN>N1THEN43
38 IF(N=N1)*(S<S1)THEN43
39 GOTO56
40 IFN<N1THEN43
41 IF(N=N1)*(S>S1)THEN43
42 GOTO56
43 PRINT"▒Well done - you made the right decision":MUSIC"_C4C‾CCRR"
44 N1=N:S1=S
45 A=A+1:IFA=7THEN50
46 PRINT"▒Next card higher or lower ? 'H' or 'L' "
47 GETA$:IFA$="H"THENT=1:GOTO31
48 IFA$="L"THENT=2:GOTO31
49 B=RND(1):GOTO47
50 PRINT"▒You've make the correct decision 6 times":MUSIC"C1D2C1D2C1D2C1D2C4
51 PRINTTAB(13);"You have won !"
52 PRINT"▒▒▒▒▒Another game ? Answer 'Y' or 'N'"
53 GETA$:IFA$="N"THENPRINT"▒";:END
54 IFA$="Y"THENFORA=1TO4:FORB=1TO13:A(A,B)=0:NEXT,:PRINT"▒";:GOTO29
55 B=RND(1):GOTO53
56 PRINT"▒Bad luck old boy you seem to have lost !":GOTO52
57 POKE59555,0:PRINT"▒";
58 PRINTTAB(12);"HIGHER OR LOWER":PRINTTAB(12);"‾‾‾‾‾‾‾‾‾‾‾‾‾‾‾"
59 PRINT"▒You have to predict whether the next"
60 PRINT"▒card will be higher or lower than the"
61 PRINT"▒current one. If the cards are the same"
62 PRINT"▒then it depends on the suits, hearts"
63 PRINT"▒beats clubs, and so on down to spades."
64 PRINT"▒If you predict wrongly, you will lose."
65 PRINT"▒If you predict correctly for 6 turns,"
66 PRINT"▒then you will have won."
67 PRINT"▒▒PRESS 'SPACE' KEY TO START THE GAME.":POKE59555,1:USR(62)
68 GETA$:IFA$=" "THENPRINT"▒":RETURN
69 B=RND(1):GOTO68
```

ALIEN ATTACK

```
1 REM ** ALIEN  ATTACK **
2 REM **    M.SEVERIN    **
3 REM **    MARCH '82    **
4 GOSUB700:POKE10167,1:TEMPO5:GOSUB400
5 DIMA(8),P(8),B(8),M(SK):FORA=1TO8:READA(A),P(A):NEXT
6 DATA80,-40,93,-39,90,1,29,41,88,40,28,39,69,-1,92,-41,121,118,120,119
7 D=1:B=5:P=53748:K=17828:FORA=1TO4:READB(A):B(A+4)=B(A):NEXT
8 PRINT"■":FORA=53248TO53287:POKEA,67:POKEA+960,67:NEXT
9 FORA=53288TO54168STEP40:POKEA,67:POKEA+39,67:NEXT:GOSUB70:TI$="000000"
10 POKEP,A(D)
11 GETA$:X=PEEK(K):POKEK,0
12 IFX=32THEN80
13 IFX=76THEN100
14 X=-(X=68)+(X=65):IFX=0THEN16
15 D=D+X:D=D+8*(D=9)-8*(D=0):POKEP,A(D)
16 FORA=1TOSK:X=M(A):IFX=0THEN20
17 B=P(INT(RND(1)*8)+1):IFPEEK(B+X)=A(D)THEN21
18 IFPEEK(B+X)THEN17
19 POKEX,0:X=X+B:POKEX,206:M(A)=X
20 NEXT:GOTO10
21 POKEX,0:X=B+X:POKEX,107
22 FORA=1TO10:POKE4514,80-A*5:FORB=1TO20:POKE4513,B*12:USR(68):NEXT,:USR(71)
23 POKEX,0:FORA=1TO1000:NEXT:PRINT"■C R A S H !  One ruined laser-ship !"
24 GOTO91
70 FORA=1TOSK:POKE4514,150-(A/2)*10
71 X=INT(RND(1)*1000)+53248
72 IF(PEEK(X))+(X=P)THEN71
73 USR(68):FORB=1TO200:NEXT:USR(71):POKEX,206:M(A)=X:NEXT:RETURN
80 X=P(D):B=B(D):F=P+X:A=40
81 POKE4514,A:USR(68):IFPEEK(F)THEN84
82 POKEF,B:IFF-X<>PTHENPOKEF-X,0
83 F=F+X:A=A-1:GOTO81
84 USR(71):IFPEEK(F)=67THENPOKEF-X,0:GOTO16
85 IFF-X<>PTHENPOKEF-X,0
86 FORA=1TOSK:IFF=M(A)THENM(A)=0:S=S+1
87 NEXT:POKEF,207:FORA=1TO100:POKE4514,150-A:USR(68):NEXT:USR(71):POKEF,0
88 IFS<SKTHEN16
89 X=VAL(RIGHT$(TI$,2))+60*VAL(MID$(TI$,3,2)):FORA=1TO1000:NEXT
90 PRINT"■WELL DONE, YOU DESTROYED ALL THE ALIENS":PRINT"■■IN";X;" SECONDS."
91 PRINT"■■WOULD YOU LIKE TO PLAY AGAIN ?":MUSIC"C4_BC_BD_A_B"
92 PRINT"■■PLEASE PRESS 'Y' OR 'N'"
93 GETA$:IFA$="Y"THENRUN
94 IF(A$="N")=0THENA=RND(1):GOTO93
95 PRINT"■":POKE4466,12:PRINTTAB(17);"BYE !!":POKE4466,23:END
100 POKEP,0:X=P(D):P=P+X:IFPEEK(P)=67THENP=P-X
101 IFPEEK(P)=206THENPOKEP,107:X=P:GOTO22
102 POKEP,A(D):GOTO16
400 PRINT"■":POKE4466,12:PRINT"■■■■Do you want instructions ? (Y/N)":USR(62)
401 GETA$:IFA$="N"THEN600
402 IF(A$="Y")=0THENA=RND(1):GOTO401
500 PRINT"■";TAB(14);"ALIEN ATTACK":PRINTTAB(14);"━━━━━━━━━━■"
501 PRINT"You are a Space Admiral, defending the"
502 PRINT"■planet Trydeferon. You must kill all the"
503 PRINT"aliens with your laser-ship."
504 PRINT"■Key:  Laser-ship=↑ (changes),  Aliens=■"
505 PRINT"■You can move the laser-ship as follows"
```

```
506 PRINT"■'A'  Rotate to the left."
507 PRINT"■'D'  Rotate to the right."
508 PRINT"■'L'  Move forward."
509 PRINT"■Press 'SPACE' to fire laser."
510 PRINT"■Beware for collisions !"
511 PRINT"■■■■** PRESS ANY KEY TO CONTINUE **■■■■":MUSIC"E4F GAB"
512 GETA$:IFA$=""THEN512
600 PRINT"■■■■■■■■SELECT SKILL LEVEL (1-9)"
601 PRINT"■■■■■■■■■    1=EASY, 9=HARD":MUSIC"A4B CFG"
602 GETA$:IFA$=""THEN602
603 SK=ASC(A$)-48:IF(SK>9)+(SK<1)THEN602
604 SK=SK*2:USR(62):RETURN
700 PRINT"■":POKE4466,12:PRINTTAB(12);"■ ALIEN ATTACK ■";:GOSUB705
702 FORA=0TO11:POKE53248+A*41,119:POKE53287+A*39,118
703 POKE54247-A*41,119:POKE54208-A*39,118:POKE4514,26-(A+1)*2:USR(68):NEXT
704 A$="":USR(71):POKE4465,12:FORA=1TO16:A$=A$+CHR$(96):NEXT:PRINTA$:CLR
705 FORA=1TO2000:NEXT:RETURN
```

MOSAIC

```
1 TEMPO7:GOSUB800
2 CLR:DIMA(5,5),P(4,4),A$(5),N(5),C(5):PRINT"■";:H=4465:V=H+1:GOSUB350
3 E$="E0R":FORX=1TO9:E$=E$+"ER":NEXT
4 DATA1,2,3,4,4,5,1,5,1,2,4,2,4,3,5,3
5 FORX=1TO5:READA$(X),N(X),C(X):NEXT
6 DATA■■■■,3,250,■■■■,4,247,■■■■■■,3,200,■■■■,3,35,■■■■,3,246
7 PRINT"┌─┬─┬─┬─┐":B$="│ │ │ │ │"
8 FORX=1TO3:PRINTB$:PRINTB$:PRINTB$:PRINTB$:PRINT"├─┼─┼─┼─┤":NEXT
9 PRINTB$:PRINTB$:PRINTB$:PRINTB$:PRINT"└─┴─┴─┴─┘■":FORX=1TO4
11 GOSUB300:FORX=1TO4:POKE54125+X*5,X:POKE54190-X*200,32+X:NEXT:M=0
12 PRINT"■";TAB(25);"PIECES  LEFT■■■■■■":PRINTTAB(25);"D  H  C  3  /":GOSUB500
13 REM ** MAIN BLOCK **
14 PRINT:POKEV,10:PRINTTAB(25);"POSITION ?  ■■";:USR(62)
15 GOSUB400:IFZ=33THEN100
16 Z=Z-64:IF(Z<1)+(Z>4)THEN15
17 PRINTA$;:X=Z
18 GOSUB400:IFZ=96THENPRINT"■ ■";:GOTO15
19 Z=Z-48:IF(Z<1)+(Z>4)THEN18
20 Y=Z:PRINTA$;
21 GOSUB400:IFZ=96THENPRINT"■ ■";:GOTO18
22 IFZ<>102THEN21
23 IFP(X,Y)THENMUSIC$:GOTO14
24 PRINT:POKEV,13:PRINTTAB(25);"PIECE ?  ■";:USR(62)
25 GOSUB400:N=0:FORO=1TO5:IFZ=C(O)THENN=O
26 NEXT:IFN=0THENMUSIC$:GOTO25
27 IF(A(X,Y)=N)+(N(N)=0)THEN80
28 IF(A(X,Y+1)=N)+(A(X+1,Y+1)=N)THEN80
29 IF(A(X+1,Y)=N)+(A(X+1,Y-1)=N)THEN80
30 IF(A(X,Y-1)=N)+(A(X-1,Y-1)=N)THEN80
31 IF(A(X-1,Y)=N)+(A(X-1,Y+1)=N)THEN80
32 PRINTA$:P(X,Y)=1:N(N)=N(N)-1:A(X,Y)=N:GOSUB36:GOSUB500:M=M+1:IFM<16THEN14
33 MUSIC"E7DCR_G9_A9R3C7DER7_G9_A"
35 PRINT:POKEV,16:PRINTTAB(25);"WELL DONE !":GOTO105
36 POKEV,21-Y*5:POKEH,X*5-4:PRINT"┌■■■■■┐";A$(N);"■│■■■■■│■■│■■■■■└─┘"
37 RETURN
```

```
80 MUSICE$:GOTO14
100 PRINT:POKEV,16:PRINTTAB(25);"HARD LUCK !"
105 PRINT"▣▣";TAB(25);"AGAIN (Y/N) ? ";
106 GOSUB400:IFZ=89THEN2
107 IFZ=78THENPRINT"▣";:END
108 GOTO106
300 FORX=1TO4:FORY=1TO4:POKEV,21-Y*5:POKEH,X*5-4
301 POKE4514,80-X*10-Y*4:USR(68):PRINT"     ▣▣▣▣▣ ";A$(A(X,Y));
302 PRINT"▣ ▣▣▣▣▣ ▣▣ ▣▣▣▣▣  ";:NEXT,:USR(71):RETURN
350 RESTORE:FORX=1TO4:FORY=1TO4:READA(X,Y):NEXT,:RETURN
400 C=0
401 GETA$:IF(A$="")=0THENZ=ASC(A$):PRINT" ▣";:RETURN
402 T=T+1:IFT>7THENT=0:C=1-C:PRINTCHR$(207-79*C);"▣";
403 GOTO401
500 FORX=1TO5:PRINT"▣▣▣";TAB(21+X*3);
501 IFN(X)THEN503
502 PRINT"  ▣▣▣  ";GOTO504
503 PRINTA$(X)
504 PRINT:PRINTTAB(21+X*3);"  ▣▣";:IFN(X)THENPRINTN(X)
505 NEXT:RETURN
800 PRINT"▣▣▣▣▣▣▣▣▣▣▣▣▣▣Want the instructions ? Press Y or N ";:USR(62)
801 GOSUB400:IFZ=78THENRETURN
802 IFZ<>89THEN801
803 PRINT"▣";TAB(17);"MOSAIC▣▣▣▣▣▣▣‾‾‾‾‾‾‾"
```

```
804 PRINT"This is a puzzle needing great skill and▣also a little luck."
805 PRINT"▣You have to place your pieces on the"
806 PRINT"▣Grid so that they aren't adjacent to one▣similar, even diagonall
807 PRINT"▣After a shape is covered, the one below ▣it can be forgotten."
808 PRINT"▣You enter the position by using A-D as  ▣columns and 1-4 as rows.
809 PRINT"▣You may use DEL to clear mistakes. When ▣finished, press 'CR'"
810 PRINT"▣PRESS SPACE TO CONTINUE ";:USR(62):GOSUB400
820 PRINT"▣You must then enter the piece that you  ▣wish to play."
821 PRINT"▣(These are the SHIFT's of D,H,C,3 and /)"
822 PRINT"A key is provided below the pieces.▣"
823 PRINT"There's no need to press 'CR' this time."
824 PRINT"Any illegal moves or keys are ignored."
825 PRINT"▣To give up push ! when asked 'POSITION'."
826 PRINT"The idea is to fit the 16 pieces in the"
827 PRINT"▣grid, so no similar ones are adjacent."
828 PRINT"▣This might sound easy but it is"
829 PRINT"▣VERY DIFFICULT."
830 PRINT"▣▣PRESS SPACE TO START ";:USR(62):GOTO400
```

Two BASIC Games' Listings from **Mr. D. Adams**

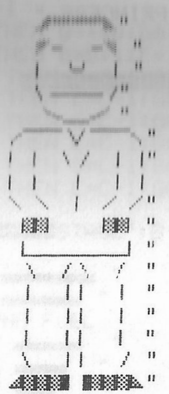
CASTLE

```
1 PRINT"■":TI$="000000":GOTO 9500
2 PRINT"■■■■┌┬┬┬┐          ┌┬┬┬┬┐ "
3 PRINT"■■■■|||||          ||||||"
4 PRINT"■■■■|||=|          ||||=| "
5 PRINT"■■■■ [1]||          | [2]|| "
6 PRINT"■■■■|||||          |||||| "
7 PRINT"■■■■|||=           ||||=| "
8 PRINT"■■■■|||||          ||||||"
9 PRINT"■■■■└┴┴┴┘          └┴┴┴┴┘ "
10 PRINT"■■■PICK ONE OF THE DOORS"
11 R=INT(2*RND(1))+1
12 INPUT A:IFA=R THEN Z=Z+1:GOTO 40
13 IFA>2 THEN PRINT"NUMBERS 1-2 STUPID!!!":GOTO 12
14 PRINT"■■■■■■■         ΦΦΦΦΦ
15 PRINT"■■■■         ~ ,. .\~      ↑ "
16 PRINT"■■■■         Y ┌┬┐ Y     ▲"
17 PRINT"■■■■         Ч └┴┘ Ͱ     ▲"
18 PRINT"■■■■          \___/      ▲"
19 PRINT"■■■■           \VVV/      ■"
20 PRINT"■■■■          <■■■>   /─┘"
21 PRINT"■■■■      /─── \_/ ───\ /  ┘"
22 PRINT"■■■■     /  ▫▫▫▫▫▫▫▫  \■■■"
23 PRINT"■■■■     | |>>>╥<<<| |  ■"
24 PRINT"■■■■     | |\\\ ///| |  ■"
25 PRINT"■■■■     | |*******|   /■"
26 PRINT"■■YOU PICKED THE WRONG DOOR AND YOU COME"
27 PRINT"■FACE TO FACE WITH A MONSTER!!"
28 MUSIC"_A0‾A0_A0‾A0_A0"
29 PRINT"HE'S GRABBED YOUR TORCH SO YOU CAN'T"
30 PRINT"SEE THE ENTRANCE TO THE MAGIC CASTLE"
31 FOR I=1TO3000:NEXT I
40 PRINT"■":Y=INT(120*RND(1))+1
41 PRINT"■■XOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOX"
42 PRINT
43 PRINT
44 PRINT"XOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOXOX"
45 IF Z=1 THEN 210
50 GETA$:IFA$=""THEN 130
60 POKE 53248+40+X,64
70 IFA$="Y"THEN X=X-40
80 IFA$="N"THEN X=X+40
90 IFA$="G"THEN X=X-1
100 IFA$="J"THEN X=X+1
115 IF PEEK(53248+40+X)=67 THEN 250
116 IF PEEK(53248+40+X)=109 THEN 260
120 IF X=Y THEN 250
130 POKE 53248+40+X,202
140 GOTO 50
210 MUSIC"‾A0"
220 POKE 53248+Y+T,67
240 GOTO 50
250 GOTO 280
260 PRINT"YOU'VE BEEN ELECTRICUTED AND  YOU ARE"
270 PRINT"FRYING TONIGHT!!!!!":FOR J=1TO1000:NEXT J:GOTO 6100
280 PRINT"■"
290 PRINT"■#####################################"
291 PRINT"#     *          *       ●#"
310 PRINT"###*########    ###    *######"
320 PRINT"#  #     #  ◢  #    ### ###"
```

```
330 PRINT"#  #     #* #  # *#     #"
340 PRINT"# ###    #   ###  #     #"
350 PRINT"#  ######## #       ###   #"
360 PRINT"#   *      *###       *# ###"
370 PRINT"# ###       # ######      #"
380 PRINT"#*# ####    #     #* ######"
390 PRINT"# #   #   ########## #   #"
400 PRINT"# #######              #######"
410 PRINT"#●     *        *        #"
420 PRINT"##############################"
430 PRINT"■■■■THE MAGIC CASTLE"
440 GETD$:IFD$=""THEN 550
445 POKE 53248+V+123,64
450 IFD$="V"THEN V=V-40
455 IF D$="H"THEN 6000
460 IFD$="N"THEN V=V+40
470 IFD$="G"THEN V=V-1
480 IFD$="J"THEN V=V+1
490 IFD$="T"THEN V=V-41
500 IFD$="U"THEN V=V-39
510 IFD$="B"THEN V=V+39
520 IFD$="M"THENV=V+41
525 IF PEEK(53248+V+123)=71  THEN 1010
530 IF PEEK(53248+V+123)=163 THEN 600
540 IF PEEK(53248+V+123)=107 THEN 700
550 POKE 53248+V+123,202
560 GOTO 440
600 PRINT"YOU'VE BEEN ELECTRICUTED AND YOUR"
610 PRINT"FRYING TONIGHT"
611 MUSIC"_BB_GG_EE_CC^DD^FF^AA"
615 FOR O=1TO1000:NEXT O:GOTO 6100
700 MUSIC"_A0": H=INT(10*RND(1))+1
701 IF H=2 THEN 4000
702 IF H=3 THEN 4000
703 P=P+1:IF P=5 THEN P=0:GOTO 280
710 IF H=5 THEN PRINT"■■A PIECE OF THE LADDER  ":B=B+1:GOTO 1000
720 IF H=7 THEN PRINT"■■SOME MONEY          ":G=G+10:GOTO 900
730 PRINT"■■A WORTHLESS BUNDLE          ":GOTO 440
800 PRINT"AGAIN";
801 GETF$:IFF$=""THEN 801
810 IFF$="Y"THEN 1
820 PRINT"BYEEEEEEE"
900 IF G=30 THEN PRINT"■■YOU CAN AFFORD A LADDER":GOTO 440
1000 IF B=3 THEN PRINT"■■YOU'VE FOUND ALL THE LADDER":GOTO 440
1001 GOTO 440
1010 IF G=30 THEN 1030
1020 IF B=3 THEN 1030
1021 GOTO 440
1030 PRINT"■▲▲▲▲▲▲"
1040 PRINT"■■■■■■■■"
1050 PRINT"■■■■■    "
1060 PRINT"■■■■■ ▌▌▌▌"
1070 PRINT"■■■■■■■■■■"
1080 PRINT"■■■■■o■■"
1090 PRINT"■■■■o■■■"
1100 PRINT"■■■o■■■■"
1110 PRINT"■■■■o■■■"
1120 PRINT"■■o■■■■■"
1130 PRINT"■■o■o■o■"
1140 PRINT"■■■■■o■■"
```

```
1150 PRINT"■■■oo■■■"
1160 PRINT"■■o■■o■■"
1170 PRINT"■■■o■■■■"
1180 PRINT"■■o■oo■■"
1190 PRINT"■■■o■■oo"
1200 PRINT"                  ──────────────────"
1300 PRINT"          THE TOWER"
1301 V=0
1310 GETC$:IFC$=""THEN 1310
1311 POKE 53248+S+660,64
1312 IFC$="Y"THEN S=S-40
1313 IFC$="N"THEN S=S+40
1314 IFC$="G"THEN S=S-1
1315 IFC$="J"THEN S=S+1
1316 IFC$="T"THEN S=S-41
1317 IFC$="U"THEN S=S-39
1318 IFC$="B"THEN S=S+39
1319 IFC$="M"THENS=S+41
1320 IF PEEK(53248+S+660)=67 THEN 2000
1321 IF PEEK(53248+S+660)=74 THEN 7000
1322 POKE 53248+S+660,202
1323 GOTO 1310
2000 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"
2100 PRINT"   ▲■■■"
2200 PRINT"  ■└┘■■"
2300 PRINT"  ■■■■■"
2400 PRINT"   ▄▄▄ "
2500 PRINT" //  \\"
2510 PRINT" └┘  └┘"
2520 PRINT" ■°°°■"
2530 PRINT"  | | | "
2540 PRINT"  | | | "
2550 PRINT"  | | | "
2560 PRINT"  | | | "
2570 PRINT"▄▄▄▄▄▄▄"
2600 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■HELP"
2610 PRINT"■■■■■■■■YOU'VE FALLEN OF THE LADDER IDIOT!!"
3000 PRINT
3100 FOR D=1TO1000:NEXT D:GOTO 6100
4000 L=INT(N*RND(1))+1:IFL<J+1THENL=J
4001 M=50
4002 PRINT"■"
4100 PRINT"YOU'VE BEEN ATTACKED"
4200 PRINT"YOUR STAMINA=";M
4300 PRINT"THE MONSTERS STAMINA=";L
4310 IF M<0 THEN 5700
4320 IF L<0 THEN 5800
4400 PRINT"■■■■■STRIKE NOW";
4500 GETN$:IFN$=""THEN 4500
4600 IF N$="S"THEN 4700
4610 IF N$="P"THEN 4660
4650 GOTO 4500
4660 IF C=2 THEN PRINT"YOU'VE GOT NO POTION":GOTO 5000
4670 IF C=0 THEN PRINT"YOU'VE GOT NO POTION":GOTO 5000
4680 IF C=1 THEN PRINT"SO YOU USED THE POTION":C=0:GOTO 5800
4700 Q=INT(2*RND(1))+1
4800 IF Q=2 THEN PRINT"A MISS":GOTO 5000
4900 PRINT"A HIT":GOTO 5500
5000 J=INT(10*RND(1))+1
5100 PRINT"YOU TAKE ";J;" DAMAGE":M=M-J:GOTO 4002
5500 A=INT(10*RND(1))+1
5600 PRINT"THEN MONSTER TAKES ";A;" DAMAGE":L=L-A:GOTO 4002
```

```
5700 PRINT"THE MONSTER KILLS YOU":FOR P=1TO1000:NEXT P:GOTO 6100
5800 PRINT"YOU KILLED THE MONSTER":K=K+1:W=W+1:FOR T=1TO900:NEXT T
5805 IF W=3 THEN 9000
5810 POKE 53248+V+83,202
5820 GOTO 280
5900 PRINT"▓AGAIN";
5910 GETW$:IFW$=""THEN 5910
5920 IFW$="Y"THEN 1
5930 PRINT"BYEEEEEEEE":END
6000 PRINT"▓"
6010 PRINT"STATUS REPORT"
6020 PRINT"MONSTERS KILLED";K
6030 PRINT"MONEY";G
6035 PRINT"MONSTER SKINS";W
6040 PRINT"PIECES OF LADDER";B
6045 PRINT"POTIONS";C
6050 PRINT"TIME ELAPSED ";TI$
6060 FOR E=1TO1000:NEXT E
6070 GOTO 200
6100 PRINT"▓"
6110 PRINT"
6120 PRINT"
6130 PRINT"
6140 PRINT"
6150 PRINT"
6160 PRINT"
6170 PRINT"
6180 PRINT"
6190 PRINT"
6200 PRINT"
6300 PRINT"
6400 PRINT"
6410 PRINT"
6420 PRINT"
6430 PRINT"
6440 PRINT"
6450 PRINT"
```



```
6456 FOR Y=1TO3
6460 FOR U=1TO200:NEXT U
6465 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓\———/                "
6470 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓/o   o\             "
6471 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓B"
6472 MUSIC"_C0"
6473 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓BA"
6474 MUSIC"_C0"
6475 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓BAD"
6476 MUSIC"_C0"
6477 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓BAD L"
6478 MUSIC"_C0"
6479 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓BAD LU"
6480 MUSIC"_C0"
6481 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓BAD LUC"
6482 MUSIC"_C0"
6483 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓BAD LUCK"
6484 MUSIC"_C0"
6485 FOR O=1TO200:NEXT O
6486 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓/o   o\"
6487 FOR O=1TO200:NEXT O
6490 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓/-   -\      "
6495 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓"
6500 NEXT Y
6555 PRINT"▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓GOODBYE"
```

```
6556 FOR F=1TO5
6560 MUSIC"_CC‾C_DD‾D_EE‾E_FF‾F_GG‾G_AA‾A_BB‾B"
6565 END
7000 PRINT"■■■■■■■■■■■■■■■■■W
7001 MUSIC"_CC"
7002 PRINT"■■■■■■■■■■■■E"
7003 MUSIC"_AA"
7004 PRINT"■■■■■■■■■■■■■L"
7005 MUSIC"_DD"
7006 PRINT"■■■■■■■■■■■■■■L"
7007 MUSIC"_FF"
7008 PRINT"■■■■■■■■■■■■■■■D"
7009 MUSIC"_GG"
7010 PRINT"■■■■■■■■■■■■■■■■O"
7011 MUSIC"_EE"
7012 PRINT"■■■■■■■■■■■■■■■■■N"
7013 MUSIC"_BB"
7014 PRINT"■■■■■■■■■■■■■■■■■■E"
7015 MUSIC"_FF"
7016 PRINT"■■■■■■■■YOU'VE SAVED THE PRINCESS,"
7017 PRINT"■■■■THE KINGDOM IS YOURS."
7018 FOR L=1TO1000:NEXT L
7019 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■        /\    "
7020 PRINT"■■■■■■■■        /\       "
7021 PRINT"■■■■■■■■       /  \      "
7022 PRINT"■■■■■■■■       /  \      "
7023 PRINT"■■■■■■■■       ▒▒▒▒      "
7024 PRINT"■■■■■■■■     /| ΦΦ |\    "
7025 PRINT"■■■■■■■■     || ◡ ||     "
7026 PRINT"■■■■■■■■      \ ◡ /      "
7027 PRINT"■■■■■■■■       \_/       "
7028 PRINT"■■■■■■■■        ||       "
7029 PRINT"■■■■■■■■    /‾‾**‾‾\     "
7030 PRINT"■■■■■■■■/       ◡     \"
7031 PRINT"■■■■■■■■/ /         \ \"
7032 PRINT"■■■■■■■■■■■■_           "
7033 FOR Y=1TO200:NEXT Y
7040 PRINT"■■■■YOU'RE TIME WAS";TI$
7050 END
9000 PRINT"YOU HAVE 3 SKINS"
9001 PRINT"YOU CAN AFFORD ONE OF THE FOLLOWING:"
9010 PRINT"(1) POTION"
9020 PRINT"(2) A PIECE OF THE LADDER"
9030 PRINT"(3) 10 GOLD COINS"
9031 PRINT"WHAT NUMBER";
9032 GETP$:IFP$=""THEN 9032
9034 IFP$="1"THEN C=C+1
9035 IFP$="2"THEN B=B+1
9036 IFP$="3"THEN G=G+10
9038 GOTO 280
9500 PRINT"WHAT DIFFICULTY LEVEL 1 TO 4"
9510 PRINT"(4 IS THE HARDEST)"
9520 GETS$:IFS$=""THEN 9520
9530 IF S$="1"THEN N=30:J=20
9540 IF S$="2"THEN N=40:J=30
9560 IF S$="3"THEN N=50:J=40
9570 IF S$="4"THEN N=60:J=50
9590 GOTO 2
```

Another LUNAR LANDER

```
1 TS=0:C$="                ":S=5:N$="MZ-80K":GOSUB 500
2 TEMPO7:H=0:B=0:C=1:F=50:PRINT"@"
4 PRINT"                          "
5 PRINT"@FUEL=";C$
8 PRINT"                                                  "
9 FOR G=1TOF
10 D=INT(820*RND(200)):IFD<200THEN10
11 POKE53248+D,219:NEXT
14 A=53248+160:IFH=1THEN 30
20 IFC=0THENPOKEA+Q,107:MUSIC"A3":GOTO 200
25 FORR=1TO340
30 GETA$
35 POKEA,64
40 IFA$="V"THEN Q=-40
50 IFA$="N"THEN Q=+40
60 IFA$="G"THEN Q=-1
70 IFA$="J"THEN Q=+1
74 IFPEEK(A+Q)=67 THEN Q=+40:GOTO30
75 IFPEEK(A+Q)=43 THEN B=B+1:F=10:H=1:GOTO 9
77 IFPEEK(A+Q)=219 THEN C=C-1:GOTO20
83 IFPEEK(A+Q)=112 THEN C=C-1:GOTO20
84 IFPEEK(A+Q)=177 THEN C=C-1:GOTO20
85 IFPEEK(A+Q)=178 THEN C=C-1:GOTO20
86 IFPEEK(A+Q)=119 THEN C=C-1:GOTO20
87 IFPEEK(A+Q)=60  THEN C=C-1:GOTO20
89 A=A+Q:POKEA,199
106 PRINT"@              STAGE";B
107 MUSIC"#F0L#F0"
110 IFR=40THENC$="          "
112 IFR=80THENC$="          "
120 IFR=340THENC$="NO FUEL       ":GOTO 300
122 IFR=120THENC$="         "
124 IFR=160THENC$="        "
126 IFR=200THENC$="       "
128 IFR=240THENC$="      "
130 IFR=300THENC$="     ":POKE 57347,4
170 PRINT"@FUEL=";C$
172 NEXT R
180 GOTO 30
200 REM
205 FORM=1TO5
206 PRINT"                              GAME OVER"
207 FORU=1TO400:NEXT
208 PRINT"                              "
209 FORU=1TO400:NEXT
210 NEXTM
211 TEMPO 2
220 MUSIC"G3":PRINT"@"
222 TEMPO1:PRINT"                      YOU GOT TO STAGE ";B
230 PRINT"      "
231 PRINT"      "
232 PRINT"      "
233 PRINT"      "
234 PRINT"      "
235 PRINT"      "
236 PRINT"      "
237 PRINT"      "
238 PRINT"      "
239 PRINT"      "
```

```
240 PRINT"▒▒▒  ▀▀▀▀.H.▀▀▀▀."
241 PRINT"     ▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒     "
242 PRINT"     XX XX   XX XX       "
243 PRINT"     ________________    ":MUSIC"‾D1‾E1‾CCG3"
244 IF B>S THEN S=B:GOTO250
249 PRINT"▒▒▒▒BUT YOU DID'NT MAKE THE HIGH-SCORE":MUSIC"‾C2C2_C2":GOTO272
250 PRINT"AND YOU GOT THE TOP SCORE"
260 PRINT"ENTER NAME":INPUTN$
261 R$=Q$:Q$=P$:P=O$:O$=N$
262 T=Y:Y=X:X=Z:Z=S
270 MUSIC"A2"
272 PRINT"▒▒      <<<HIGH-SCORE>>"
273 PRINT"      ";N$;"      ";S
280 PRINT"▒AGAIN?"
284 GETA$
286 IFA$="Y"THEN 2
287 IFA$="N"THEN END
288 GOTO 284
300 PRINT"▒UNLUCKY YOU'VE RAN OUT OF FUEL!!!":GOTO 200
500 PRINT"▒"
510 PRINT"▒  ▒ ▒ ▒▒▒ ▒▒▒ ▒▒▒."
512 PRINT"▒  ▒ ▒ ▒▒▒ ▒▒▒ ▒▒▒"
513 PRINT"▒▒▒ ▒▒▒ ▒ ▒ ▒ ▒ ▒▒▒."
514 PRINT"▒▒    ▒▒▒ ▒▒▒ ▒▒▒ ▒▒▒ ▒▒▒."
516 PRINT"▒ ▒▒▒ ▒▒▒ ▒ ▒ ▒▒▒ ▒▒▒"
518 PRINT"▒▒▒ ▒ ▒ ▒ ▒ ▒▒▒ ▒▒▒ ▒▒▒."
520 PRINT"▒▒▒▒        ▒▒▒▒▒▒▒▒          "
522 PRINT"      \_▒OOOOO▒_/        "
524 PRINT"       ▒OOOOOO▒          "
526 PRINT"      ▒▒▒▒▒▒▒▒▒▒         "
528 PRINT"     ▒●●●●●●●●●●▒        "
530 PRINT"    ||||||||||||||||||||||  "
532 PRINT"      ||||||||||||||||||    "
534 PRINT"        ||||||||||||||      "
536 PRINT"          ||||||||||        "
538 PRINT"           |||||            "
540 PRINT"        \\ || //            "
542 PRINT"_/\_/\______/\/\__/\/\_"
544 PRINT"enter c to continue.  "
546 GETW$
548 IFW$="C"THEN 550
549 GOTO 546
550 MUSIC"‾G3":PRINT"▒"
552 PRINT"▒▒▒▒▒▒▒▒LUNAR LANDER"
554 PRINT"         ____________"
558 PRINT"The object of this game is"
560 PRINT"to land your space-ship as"
562 PRINT"many times as possible while"
563 PRINT"dodging ASTROIDS,before"
564 PRINT"your fuel runs out.After each"
565 PRINT"stage 10 more astroids appear!"
566 PRINT"When your fuel is nearly empty"
568 PRINT"the 'LED' will turn red!"
570 PRINT"        ▒▒▒           "
572 PRINT"         ▒            "
574 PRINT"      ◢  Y  ◣         "
576 PRINT"     ▒▒G   J▒▒        "
578 PRINT"      ◥  N  ◤         "
580 PRINT"         ▒            "
582 PRINT"        ▒▒▒           "
584 PRINT"     ↑CONTROLS↑         "
```

```
586 PRINT"■press 'p' to play"
590 GETU$
592 IFU$="P"THEN RETURN
596 GOTO 590
```

THE HUMAN CANNONBALL by **Paul Vella** (13)

```
10 REM HUMAN CANNONBALL,JAN 1982
20 PRINT"■■■■■■■■■■■■■■HUMAN CANNONBALL "
30 PRINT"■■■■  Copyright:  Paul Vella age 13"
40 PRINT"■■■■──────────────────────────────────────────"
50 PRINT"■■■  In this game you are a daring human"
60 PRINT"■cannonball who is trying to land on a"
70 PRINT"■small platform."
80 PRINT"■  The only control you have is the "
90 PRINT"■amount of gun powder which you must set"
100 PRINT"■before take-off."
110 PRINT"■■■■Good luck,you'll need it!"
120 PRINT"■■              ANY KEY TO START"
130 USR(62)
140 GET Z$:IF Z$=""THEN 140
150 PRINT"■"
160 X=17:Y=5
165 REM **POSITION PLATFORM**
170 D=INT(17*RND(1))
180 D=D+13
190 POKE4466,15:PRINTTAB(1);"      / /"
200 POKE4466,16:PRINTTAB(1);"     / /  "
210 POKE4466,17:PRINTTAB(1);"    / /   "
220 POKE4466,18:PRINTTAB(1);" ▬o▬o▬  "
230 POKE4466,16:PRINTTAB(D);"  ___    "
240 POKE4466,17:PRINTTAB(D);"   ||    "
250 POKE4466,18:PRINTTAB(D);"   ||    "
260 POKE 4466,19:PRINT"──────────────────────────────────────────"
270 POKE4466,18:PRINTCHR$(99)
280 POKE4466,0:PRINT"                                        "
290 POKE4466,0:INPUT "AMOUNT OF GUNPOWDER (0-16) ?";S
300 IF S>=17 THEN GOTO 280
305 REM **FLIGHT ROUTINE**
310 IF S>0 THEN POKE4466,X:PRINTTAB(Y);CHR$(99)
320 IF S<=0 THEN POKE4466,X:PRINTTAB(Y);CHR$(102)
330 POKE4466,X:PRINT TAB(Y);" "
340 S=S-1
350 IF S>0 THEN X=X-1:IF S>0 THEN Y=Y+1
360 IF S<=0 THEN X=X+1:IF S<=0 THEN Y=Y+1
370 POKE4466,2:IF (X=17)*(Y=D+4)THEN PRINT"SAFE LANDING!":GOTO 440
380 IF X>19 THEN PRINT"CRASH!":TEMPO 7:MUSIC"C2CCCCCCCCCCCC":GOTO 530
390 GOTO 310
400 POKE4466,4:PRINT"ANY KEY FOR ANOTHER TRY"
410 GET Z$:IF Z$=""THEN 410
420 PRINT"■"
430 GOTO 150
```

```
435 REM **SAFE LANDING ROUTINE**
440 FOR T=1 TO 1
450 FOR A=10 TO 1 STEP-1
460 POKE4514,A
470 FOR B=0 TO 255 STEP A
480 POKE4513,B:USR(68)
490 NEXT B,A:USR(71)
500 NEXT T
510 POKE4466,16:PRINT TAB(D+3);CHR$(99)
520 GOTO 400
530 S=S-S-S-S+4:POKE4466,20:PRINT TAB(S);CHR$(101)
540 FOR Z=1 TO 800:NEXT
545 REM **STRETCHER BEARERS**
550 FOR L=S TO 36
560 FORM=3 TO S
570 POKE4466,20:PRINTTAB(M-2);" ";CHR$(99);" ";CHR$(99)
580 FORZ=1 TO 50:NEXTZ
590 NEXTM
600 FORL=S TO 36
610 POKE4466,20:PRINT TAB(L-2);CHR$(128);CHR$(99);CHR$(101);CHR$(99)
620 FOR F=1 TO 50:NEXTF
630 NEXT L
640 GOTO 400
650 END
```

1983 SUBSCRIPTION FORM

Please complete this form in BLOCK CAPITALS.

Membership No: / / / /

NAME: .....

ADDRESS: .....

.....

.....

.....

1983 Membership @ £5.50 (£10.00 overseas) |_|

1982 Back Issues @ £7.50 (£12.00 overseas) |_|

1981 Back Issues @ £3.00 (£6.00 overseas) |_|

I enclose Cheque/P.O. made payable to SHARPSOFT LTD for £.....

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

Expiry Date / /

Signature .....

Please complete the section below so that we may endeavour to cater for all our Members' interests and future requirements.

Please indicate your machine type:

| MZ-80A | | MZ-80B | | MZ-80K | | PC3201 |

Whether you have any of the following:

EXPANSION UNIT ONLY	FLOPPY DISCS	PRINTER P3 P4 P5 P6 EPSON DAISYWHEEL OTHER	OTHER

Which area(s) your system is used in:

| Personal | | Business | | Education | | Medical |

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