ISSUE 7

SHARPSOFT

C O N T E N T S

All our thanks again to John Trippick for his lively illustrations and cartoons.

SHARPSOFT USER NOTES

Issue No. 7

This issue marks the start of our third year of publication. We regularly receive requests from readers for more issues per year. The demand for information concerning SHARP computers seems to grow at an exponential rate! Our aim in publishing these User Notes has always been to provide information for our readers, on a regular basis, at a reasonable cost. If we were to increase the number of issues published per year our subscription charges would also have to increase. High charges would, we believe, make this publication too expensive for some of our readers. In the long term higher subscription charges would defeat our original aim - that the publication should be easily available to all SHARP users, at the lower cost possible.

A second and related issue concerns the type of material we publish in each issue. We do try in each issue to achieve a balance in the articles, programs and letters - so that there is something of interest for all readers regardless of which SHARP computer you own. Hence in the future we propose to continue publishing, in their present form, three issues of these User Notes per year. However, in addition to the User Notes we will in the future also publish a range of small low cost books on specific programming subjects and MZ-80K, MZ-80A and MZ-80B topics. Often these books will cover material which is more machine dependent than the general articles present in the User Notes.

Over a year ago we published, as an experiment, "A beginners guide to the SHARP MZ-80K microcomputer" by R.G. Meadows. This book has been very popular with beginners probably because it is ideal for the person who is new to computing. A new version of this book written specifically for the MZ-80A computer has just been completed by Dick Meadows. This should help MZ-80A owners who require a simple introduction to SHARP BASIC. Dick Meadows is also writing a new book called "Starting SHARP PASCAL". This book is similar in style to his BASIC books and aims at helping SHARP computer users understand the SHARP tape based interpretive PASCAL language.

Towards the end of last year I wrote a number of BASIC graphics programs for the MZ-80B. These programs were intended as learning exercises on my part. Roughly four months later, and a lot of work, these exercises have developed into a book called "A Practical Guide to the MZ-80B Computer Graphics". This book includes roughly fifty BASIC programs which range from simple demonstrations of the SHARP BASIC high resolution graphics extensions to a complete three dimensional perspective graphics drawing package.

Both the PASCAL and graphics books will be published towards the end of April 1983. Write to us at SHARPSOFT if you would like further information regarding the publication dates, contents and cost.

This issue of the User Notes includes a selection of articles, programs, hints, tips and comments from our readers. Looking back to Issue No. 1 and found that one of our original aims was to publish at least "one" program per issue. This figure has crept up to at leat five programs per issue!

In future issues we would like to publish more SHARP PASCAL "hints" and programs. If you have written a PASCAL program or found an interesting language feature or "bug" in the SHARP PASCAL DO drop us a line at SHARPSOFT.

Issue 8 of these User Notes will be published towards the end of June or early July 1983.

Mike Brinson

EDITOR

FORTH - Bugs, comments and programs

by

Mike Brinson

The FORTH column this issue contains mainly programs. This article is in response to a number of readers' requests asking for example programs. If you have written any FORTH programs do please send them to me for future publication.

1. Fig-FORTH portable editor extensions

Written by Kim Harris and first published in FORTH DIMENSIONS VOL. II/6, p. 161.

The EDITOR extensions listed in the following screen can be added to the fig-FORTH EDITOR V2.0 published in Issue 5 of these User Notes. The new editor words should be inserted between lines 6 and 7 of screen 12, see page 6 of Issue 5. You must, of course, edit screen 12 to do this.

NEW start-line No. ---

The word NEW allows one or more lines to be replaced by strings entered from the keyboard, for example 2 NEW (CR) allows replacement from line 2. If you enter some text and a (CR), NEW will prompt you for a new line 3 and so on. This continues until you replacce line 15 or enter only a (CR) at the start of the line. then that line and any remaining lines are listed unchanged.

UNDER start-line No. ---

The word UNDER is similar to NEW, it lets you insert one or more lines starting at a specified line number. Any lines pushed off line 15 are lost.

```
 0 ( Fig FORTH portable editor extensions )
 1 ( by Kim Harris FORTH DIMENSIONS II/6, P161 )
 2 DECIMAL
 3 : ENTER? OVER = ; ( start.line no. current.line no. --- f )
 4 : ENTER QUERY 1 TEXT ;
 5 : NULL? TIB @ C@ 0= ;  ( --- f )
 6 : .BS   8 EMIT ;
 7 : NEW
 8       FORTH 16 0 DO CR 1 3 .R SPACE I ENTER? IF ENTER NULL?
 9       IF .BS I SCR @ .LINE ELSE I EDITOR R FORTH 1+ THEN ELSE
10       1 SCR @ .LINE THEN LOOP DROP ;
11 : UNDER
12         FORTH 1+ 16 0 DO CR 1 3 .R SPACE I ENTER? IF ENTER
13         NULL? IF .BS I SCR @ .LINE ELSE I EDITOR I FORTH
14         1+ THEN ELSE I SCR @ .LINE THEN LOOP DROP ;
15 ;S
```

2. Prime number benchmark programs

```
 0 ( BENCH.PRIMES benchmark primes  )
 1 ( upper.limit BENCH.PRIMES )
 2 : BENCH.PRIMES DUP 2 / 1+ SWAP ." STARTING " CR
 3                1 DO DUP I 1 ROT
 4                2 DO DROP DUP I /MOD
 5                DUP 0= IF DROP DROP 1 LEAVE
 6                ELSE 1 = IF DROP 1
 7                ELSE DUP 0 > IF DROP 1
 8                ELSE 0= IF 0 LEAVE
 9                ENDIF ENDIF ENDIF ENDIF LOOP
10                IF 4 .R ELSE DROP ENDIF
11                LOOP DROP CR ." FINISHED " ;
12 ( From FORTH DIMENSIONS II/4, p112 )
13 ;S
14
15
```

```
 0 ( PRIMES bench mark program from BYTE Sept. 1981, p190 )
 1 8190 CONSTANT SIZE                0 VARIABLE FLAGS    SIZE  ALLOT
 2 : PRIMES     FLAGS SIZE 1 FILL
 3              0 SIZE 0
 4              DO FLAGS I + C@
 5              IF I DUP + 3 + DUP I +
 6                 BEGIN DUP SIZE <
 7                 WHILE 0 OVER FLAGS + C! OVER + REPEAT
 8                 DROP DROP 1+
 9              THEN
10              LOOP
11              . ." PRIMES " ;
12 ;S
13
14
15
```

3. More FORTH extensions

```
 0 ( ARRAY 2ARRAY PICK CASE )
 1 : ARRAY                                          ( size ARRAY )
 2         <BUILDS 0 DO 0 , LOOP DOES> SWAP DUP + + ;
 3 : 2ARRAY                                         ( x y 2ARRAY )
 4          <BUILDS 2DUP SWAP , , * 0 DO 0 , LOOP
 5          DOES> DUP @ ROT * DUP + + SWAP DUP + + 4 + ;
 6 : PICK       SP@ SWAP 2 * + @ ;
 7 ( CASE statement, by C. E. Eaker FORTH DIMENSIONS II/3, P37 )
 8 : CASE ?COMP CSP @ !CSP 4 ; IMMEDIATE
 9 : OF    4 ?PAIRS COMPILE OVER COMPILE = COMPILE 0BRANCH
10        HERE 0 , COMPILE DROP 5 ; IMMEDIATE
11 : ENDOF 5 ?PAIRS COMPILE BRANCH HERE 0 ,
12         SWAP 2 [COMPILE] ENDIF 4 ; IMMEDIATE
13 : ENDCASE 4 ?PAIRS COMPILE DROP BEGIN SP@ CSP @ = 0=
14            WHILE 2 [COMPILE] ENDIF REPEAT CSP ! ; IMMEDIATE
15 ;S
```

4. Random number generator

```
 0 ( RANDOM random number generator )
 1 0 VARIABLE SEED
 2 : (RAND) SEED @ 259 * 3 + 32767 AND DUP SEED ! ;
 3 : RANDOM (RAND) 32767 */ ;
 4 ( From FORTH DIMENSIONS II/2, p34 )
 5 ( By J. E. RICKENBACKER )
 6 ;S
 7
 8  To use RANDOM      range-1 RANDOM
 9  for example to output 100 random numbers in the range
10  0 to 1000
11 : TEST.RANDOM 101 0 DO 999 RANDOM . LOOP ;
12
13
14
15
```

5. Non-destructive stack print

```
 0 ( Stack print aids )
 1 : .S  ( Display stack )
 2     SP@ S0 @ = IF CR ." Stack empty "
 3                 ELSE SP@ S0 @ SWAP
 4                     DO CR I @ DUP DECIMAL 4 .R
 5                        HEX ."     (" 0 4 D.R ." H)"
 6                        2 +LOOP
 7                 CR ENDIF DECIMAL ;
 8 ;S
 9
10
11
12
13
14
```

6. Eight Queens problem

```
 0 ( Eight Queens problem by J. Levan  )
 1 ( FORTH DIMENSIONS II/1, P6 )
 2 : 2* DUP + ;
 3 : MYSELF  ( SO THAT A WORD CAN CALL ITSELF BY RECURSION )
 4         LATEST PFA CFA , ; IMMEDIATE
 5 : IARRAY  ( form an array of I's )
 6         <BUILDS 0 DO 1 , LOOP DOES> SWAP 2* + ;
 7 8 IARRAY A   16 IARRAY B  16 IARRAY C  8 IARRAY X
 8 : SAFE
 9         SWAP OVER OVER OVER OVER - 7 + C @ >R
10         + B @ >R DROP A @ R> R> * * ;
11 : MARK
12         SWAP OVER OVER OVER OVER - 7 + C 0 SWAP !
13         + B 0 SWAP ! DROP A 0 SWAP ! ;
14 --->
15
```

```
 0 ( Eight Queens  second screen )
 1 : UNMARK
 2         SWAP OVER OVER OVER OVER - 7 + C 1 SWAP !
 3         + B 1 SWAP ! DROP A 1 SWAP ! ;
 4 0 VARIABLE TRIES
 5 : PRINTSOL ." Found on try " TRIES @ 6 .R 8 0
 6           DO I X @ 1+ 5 .R LOOP CR ;
 7 : TRY
 8      8 0 DO 1 TRIES +! ?TERMINAL IF QUIT THEN DUP I
 9      SAFE IF DUP I MARK DUP I SWAP X ! DUP 7 <
10      IF DUP 1+ ?STACK MYSELF ELSE PRINTSOL THEN
11      DUP I UNMARK THEN LOOP DROP ;
12 : DO-IT
13       0 TRIES ! 0 CR TRY ;
14 ;S
15
```

7. Greatest common division program

```
 0 ( Greatest common divisor of two numbers )
 1 ( by R. L. Smith FORTH DIMENSIONS II/6, p 167 )
 2 : GCD
 3      BEGIN
 4       SWAP OVER MOD -DUP 0=
 5      UNTIL ;
 6 : G-C-D
 7         GCD CR
 8         ." The greatest common divisor is " . CR ;
 9 ;S
10
11 Enter two numbers, then type G-C-D.
12 The greatest common divisor of these numbers is then displayed.
13
14
15
```

8. Cursor Control - a full screen editor and plotting functions

A contribution from **Mr. S. Wallis**

Dear Sharpsoft,

Thank you for the User Notes and FORTH. However, I don't think much of the text editors. (Why not use proper cursor control?) I enclose a screen aditor (screens 22 & 23) for the MZ-80K.

After typing 22 LOAD (CR), a screen may be edited using [screen number] EDIT (CR). If the screen is on tape, press PLAY as requested. If you want to start a new screen, press BREAK. Otherwise, the screen will be retrieved from memory.

The cursor keys, HOME, SML and CAP will now work as expected and INST and DEL operate on the whole screen. The familiar reverse cursor characters can be obtained by pressing SHIFT/SPACE, poress it again and they regain their normal functions. CR will not clear the screen.

When you want to store the screen and return to direct mode, press SHIFT/CR. If you wish to abort the screen leaving it unchanged, press SHIFT/BREAK.

The screen editor handles screens of 999 characters rather than the usual 16 lines of 64 characters. The LOAD command does not mind but LIST produces strange listings. Use EDIT to list the screen normally but screens cannot be listed on the printer this way. For this reason, I wrote screen 34, which will list screens in their usual format without altering their contents. Use LISTX instead of LIST. The screen was written with the screen editor and so is self-demonstrating.

Now lower case characters can be used but beware, the cimputer will crash if they are used as definitions.

Some of the other definitions used may be useful:

bl ... bn n M/C cccc defines a machine code definition that executes and jumps to NEXT.

SHIFT leaves a flag on the stack indicating if SHIFT is being pressed.

INSERT and DELETE perform these functions.

OUTCHAR outputs a character (cursor in reverse field) and DOWN moves the cursor down a line, if it will not cause the screen to scroll.

KEY2 is the same as KEY except that INST does not toggle the printer on/off.

FORTH printing is very slow because it uses the monitor subroutine at 0012H, far too slow for real-time games. The enclosed plotting routine (sccreen 1) will plot up to about 50 characters in the time one character is printed.

Strings to be plotted may include left, right and downwards cursor characters. A screen address mujst be specified where the string will be plotted and each character must be plotted 0 to 255 bytes further on from the last in the string.

To use the routine (after 1 LOAD), all strings must be defined as constants using:

& string& CONSTANT [name]

with one space after the first "&" but none before the final "&". Strings may then be plotted using:

[address] [name] PLOT

The routine will only plot characters that lie on the screen area so that objects may easily be moved on the top and bottom of the screen. This also makes the routine crash-proof. The top line of the screen is assumed to be reserved for the score etc.

If this is required to be changed the numbers 208, 40, 211 and 232 preceeded by 254 in lines 5 to 7 representing the top left and bottom right of the screen may be altered.

Screen 2 is a demonstration of the routine. Even faster plotting is achieved by using PL instead of PLOT when the effect of BLINK (to prevent snow) in the last PLOT still remains.

```
  1 LIST
SCR # 1
  0 ( Plotting commands 16.5.82 )
  1 : M/C CREATE DUP >R DUP 1+ ALLOT 0 DO -2 ALLOT C, LOOP
  2      R> 1 - ALLOT 195 C, 4677 , SMUDGE ;
  3 225 125 205 185 11 111 229 7 M/C A-D
  4 205 166 13 3 M/C BLINK
  5 209 225 197 26 254 13 40 31 19 79 6 0 9 26 19 79 124 254 208
  6 32 4 125 254 40 124 56 232 254 211 32 3 125 254 232 48 223 113
  7 24 220 193 40 M/C PL
  8 : & BLK @ IF BLK @ BLOCK ELSE TIB @ THEN DUP IN @ + HERE SWAP 0
  9 BEGIN OVER C@ DUP 0= OVER 38 = OR 0= WHILE DUP 20 = IF DROP 1 -
 10 ELSE DUP 19 = IF DROP 1+ ELSE DUP 17 = IF DROP 40 + ELSE A-D
 11 SWAP C, C, 1 THEN THEN THEN SWAP 1+  SWAP REPEAT SWAP DROP IF
 12 1+ THEN ROT - IN ! 13 C, ;
 13 : PLOT BLINK PL ;
 14 ;S
 15
OK
     2 LIST
    SCR # 2
      0 ( Demonstration of plotting commands :-
      1    1. Define graphic "string" using "&". )
      2 &   ▶▬▬___▦▦▦▦▦▦▦  ‾‾‾‾&
      3 ( 2. Store as a CONSTANT e.g. PLANE. )  CONSTANT PLANE
      4 ( 3. When needed to plot [ after initialization ], the format
      5    [address] PLANE PLOT is used. )
      6 ( To move the plane the following word is defined. )
      7 : MOVE 54248 53248 DO I PLANE PLOT LOOP ;
      8 22 EMIT MOVE                    ;S
      9
     10
     11
     12
     13
     14
     15
 22 LIST
SCR # 22
  0 ( Screen Editor 7.10.82 ) DECIMAL
  1 : M/C CREATE DUP >R DUP 1+ ALLOT 0 DO -2 ALLOT C, LOOP
  2     R> 1 - ALLOT 195 C, 4677 , SMUDGE ;
  3 62 248 50 0 224 0 58 1 224 47 230 33 33 0 0
  4 40 1 35 229 19 M/C SHIFT
  5 197 17 229 211 213 205 177 15 183 235 237 82
  6 227 193 120 7 56 15 3 17 230 211 175 50 163 232
  7 237 184 18 60 50 163 232 193 34 M/C INSERT
  8 42 113 17 124 181 202 69 18 197 62 196 205 220 13
  9 205 177 15 229 35 209 229 33 231 211 175 237 82 227
 10 193 50 163 232 237 176 60 50 163 232 193 39 M/C DELETE
 11 42 113 17 124 133 254 63 225 125 202 69 18 205 185 11
 12 205 166 13 205 112 9 21 M/C OUTCHAR
 13 205 229 39 95 195 201 40 7 M/C KEY2
 14 : DOWN 4466 C@ 24 < 4466 +! ;
 15 0 VARIABLE CURFLAG -->
OK
    23 LIST
    SCR # 23
      0 ( Screen Editor Scr 2 )
      1 : EDIT 1 CURFLAG ! DUP SCR ! BLOCK 22 EMIT 999 0 DO DUP
      2 I + C@ DUP 17 < IF DROP LEAVE ELSE 3014 + C@
      3 53248 I + ! THEN LOOP DROP BEGIN KEY2 DUP BL SHIFT + <
      4 IF DUP 13 = IF DROP SHIFT IF UPDATE PREV @ 2+ 999 0 DO
      5 53248 I + C@ 3270 + C@ OVER I + C! LOOP 999 + 25 BLANKS 22
      6 EMIT QUIT THEN 0 4465 C! DOWN ELSE DUP BL = IF DROP
      7 CURFLAG @ 1 XOR CURFLAG ! ELSE DUP 22 = IF OUTCHAR
      8 ELSE CURFLAG @ IF DUP 19 = IF 4465 C@ 4466 C@ + 63 <   IF
      9 EMIT ELSE DROP THEN ELSE DUP 17 = IF DROP DOWN ELSE EMIT THEN
     10 THEN ELSE OUTCHAR THEN THEN THEN THEN ELSE 100 OVER < OVER
     11 96 < OR IF OUTCHAR ELSE DUP 96 = IF DROP DELETE ELSE
     12 DUP 97 = IF DROP INSERT ELSE DUP 100 = IF DROP 22 EMIT
     13 QUIT THEN 103 OVER - 57347 C! 98 - 4464 C! THEN THEN
     14 THEN THEN AGAIN ;
     15 ;S
    OK
```

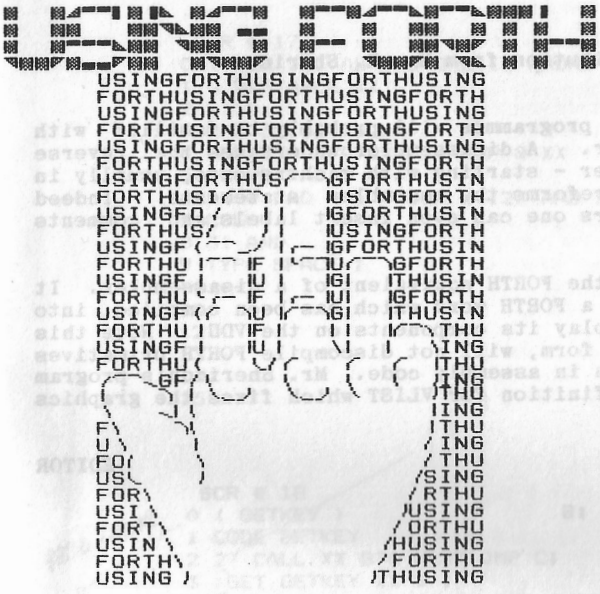
```
24 LIST
SCR # 24
  0 ( Edited screens listing command ) DECIMAL : LISTX DECIMAL CR ."
  1  SCR # " DUP DUP SCR ! . -1 99 ROT BLOCK DUP 1 - 32 ROT DUP 1000
  2  + SWAP DO DUP I C@ = IF OVER 1+ I - IF 34 = I + >R 2DUP - R + 6
  3 3 OVER < IF DROP CR ROT 1+ DUP 3 .R SPACE ROT ROT 1+ R OVER - TH
  4 EN SWAP ROT DROP R OVER - TYPE R> 32 ELSE SWAP DROP I SWAP THEN
  5 ELSE I C@ 34 = IF DROP 34 THEN THEN LOOP CR DROP DROP DROP BEGIN
  6  1+ DUP 16 < WHILE DUP 3 .R CR REPEAT DROP ; ;S
  7
  8
  9
 10
 11
 12
 13
 14
 15
OK
```

```
24 LISTX
SCR # 24
  0 ( Edited screens listing command ) DECIMAL : LISTX DECIMAL CR
  1 ." SCR # " DUP DUP SCR ! . -1 99 ROT BLOCK DUP 1 - 32 ROT DUP
  2 1000 + SWAP DO DUP I C@ = IF OVER 1+ I - IF 34 = I + >R 2DUP -
  3 R + 63 OVER < IF DROP CR ROT 1+ DUP 3 .R SPACE ROT ROT 1+ R
  4 OVER - THEN SWAP ROT DROP R OVER - TYPE R> 32 ELSE SWAP DROP I
  5 SWAP THEN ELSE I C@ 34 = IF DROP 34 THEN THEN LOOP CR DROP DROP
  6 DROP BEGIN 1+ DUP 16 < WHILE DUP 3 .R CR REPEAT DROP ; ;S
  7
  8
  9
 10
 11
 12
 13
 14
 15
OK
```

In answer to Mr. Wallis' question at the beginning of his letter. Fig-FORTH contains within its defining model a portable text editor (without any cursor control functions). This editor is very important because each make of micro-computer is often fitted with a different terminal or memory mapped VDU and key-board. Hence each configuration is likely to have differing cursor control/escape characters and sequences, for example the SHARP MZ-80B running CP/M is configured so that its VDU and keyboard simulate an ADM3A terminal. Incidentally, the cursor control sequences, in FORTH, for this configuration are:-

```
 0 ( Cursor control functions for MZ80B running CP/M )
 1 ( Simulated Lear ADM3A Terminal )
 2 FORTH DEFINITIONS DECIMAL
 3 : GOTOXY
 4           27 EMIT 61 EMIT
 5           0 MAX 23 MIN 32 + EMIT
 6           0 MAX 79 MIN 32 + EMIT ;
 7 : CLEARSCREEN 26 EMIT ;
 8
 9
10
11
12
13
14
15
```

Once the portable editor is running and debugged it is used to bootstrap a full screen editor into a particular computer. this process is another example of the extensibility of FORTH.


USINGFORTHUSINGFORTHUSING
FORTHUSINGFORTHUSINGFORTH
USINGFORTHUSINGFORTHUSING
FORTHUSINGFORTHUSINGFORTH
USINGFORTHUSINGFORTHUSING
FORTHUSINGFORTHUSINGFORTH

9. FORTH DISCOMPILER

A contribution from **Mr. R. Sheridon**

Assembly language programmes will probably be familiar with the term disassembler. A disassembler undertakes the reverse function of an assembler - starting with machine code, usually in RAM, a disassembler reforms the assembler statements. Indeed with some disassemblers one can even insert labels and comments in the assembly code.

A DISCOMPILER is the FORTH equivalent of a disassembler. It will take the name of a FORTH word which has been compiled into the dictionary and display its components on the VDU. Note this utility, in its basic form, will not discompile FORTH primatives which have been written in assembly code. Mr. Sheridon's program also includes a new definition for VLIST which fixes the graphics character bug.

EDITOR

```
 SCR # 15
0 ( DISFORTH DOC ) ;S
1 ROBERT SHERIDON
2 99 MEDESWELL
3 ORTON MALBORNE
4 PETERBOROUGH
5 CAMBRIDGESHIRE
6 PE2 OPB
7 WRITTEN ON A MZ80K USING SHARPSOFT FORTH
8
9 TO USE THIS DISCOMPILER FIRST YOU MUST
10 PLACE THE PARAMETER FIELD ADDRESS ( PFA )
11 ONTO THE TOP OF STACK BY USING
12 ' ( TICK )
13 THEREFORE THE SYNTAX IS
14 ' WORD DISFORTH
15 THIS DISCOMPILES ONE WORD
```

```
 SCR # 16
0 ( DISFORTH DOC ) ;S
1 AFTER ONE WORD HAS BEEN DISCOMPLED
2 YOU CAN WORK DOWN THROUGH THE DICTIONARY
3 BY USING THE WORD DIS
4 95% OF FORTH WORDS CAN BE DISCOMPILED
5 THE EXCEPTIONS ARE DUMMY WORDS AS 'TASK AND NOOP'
6 AND COLON DEFINITIONS THAT CONTAIN SOME MACHINE CODE
7 SUCH AS 'WARM COLD ABORT'
8 MACHINE CODE WORDS ARE DISCOMPILED BY HEX DUMP
9
10 ALSO INCLUDED IS A NEW VERSION OF ID. WITH VIST
11 AND A GET FUNCTION TOGETHER WITH PAUSE
12
13
14
15
```

```
 SCR # 17
0 ( NEW ID. AS OLD ONE IS WRONG )
1 0 VARIABLE XX
2 : ID.
3 PAD 32 32 FILL
4 DUP PFA LFA OVER - DUP @ XX ! PAD SWAP
5 CMOVE PAD
6 XX @ PAD + 1 - DUP @ 127 AND SWAP !
7 COUNT
8 31 AND
9 TYPE SPACE ;
10 -->
11
12
13
14
15
```

```
 SCR # 18
0 ( GETKEY )
1 CODE GETKEY
2 27 CALL XX STA NEXT JMP C;
3 :GET GETKEY XX @ ;
4 :PAUSE GET ( SPACE PAUSES CURSUR DOWN RESTARTS )
5 32 = IF BEGIN GET 17 =  UNTIL THEN ;
6 -->
7
8
9
10
11
12
13
14
15
```

```
 SCR # 19
0 ( NEW VLIST )
1 : VLIST
2 80 OUT ! CONTEXT
3 @ @
4 BEGIN OUT @
5 C/L >
6 IF
7 CR 0 OUT !
8 THEN DUP
9 ID. SPACE SPACE
10 PAUSE PFA LFA @ DUP 0=
11 ?TERMINAL OR UNTIL
12 DROP ;
13 -->
14
15
```

```
SCR # 20
0 ( FORTH DISCOMPILER SCR # 1 )
1 0 0 0 0
2 VARIABLE PF ( VARIABLE TO HOLD PFA )
3 VARIABLE CF ( VARIABLE TO HOLD CFA )
4 VARIABLE LF ( VARIABLE TO HOLD LFA )
5 VARIABLE NF ( VARIABLE TO HOLD NFA )
6 ( CONVERT TOP OF STACK INTO  )
7 ( FIELD ADDRESSES AND SAVE )
8 : FIELD-ADDR DUP DUP DUP
9 PF ! CFA CF ! LFA LF ! NFA NF ! ;
10 : NAMIT ( PRINT WORD AT NFA )
11 CR NF @ ID. CR CR ;
12 :GETPFA ( PUT PFA ON TOP OF STACK )
13 PF @ @ ;
14 : 2+PF ( ADD 2 TO PF )
15 2 PF +! ; -->
```

```
SCR # 21
0 ( DISCOMPILER SCR # 2 )
1 : DISPOSE-LIT
2 ( THE WORD LIT IS COMPILED IN FRONT OF ALL )
3 ( 16 BIT NUMBERS THIS IS NOT NEEDED )
4 ( SO IT IS DISPOSED OF )
5 GETPFA ( GET PFA ONTO STACK )
6 ' LIT CFA  = ( COMPARE IT WITH THE CFA OF LIT )
7 IF ( DISPOSE OF LIT AND )
8 ( PRINT 16 BIT NUMBER )
9 2+PF GETPFA . SPACE 2+PF THEN ;
10 ' LIT NFA VARIABLE LASTNF
11 ( VARIABLE TO HOLD THE LAST NFA
12 -->
13
14
15
```

```
SCR # 22
0 ( DISCOMPILER SCR # 3 )
1 : SEARCH-BRANCH
2 GETPFA DUP ( GET PFA ONTO STACK TWICE )
3 ' OBRANCH CFA = ( COMPARE WITH CFA OF OBRANCH )
4 SWAP ( PUT PFA BACK ON TOP )
5 ' BRANCH CFA ( COMPARE WITH CFA OF BRANCH )
6 IF 2 SPACES GETPFA NFA ID. 2+PF ( PRINT BRANCH )
7 ( AND MOVE POINTER )
8 GETPFA . 2+PF ( PRINT DISPLACEMENT )
9 DISPOSE-LIT ( IF BRANCH IS FOLLOWED BY A 16 BIT NUMBER )
10 GETPFA ' ;S CFA =
11 ( IS NEXT BYTE ;S )
12 IF QUIT THEN THEN ;
13 ( IF BRANCH IS FOLLOWED BY ;S THEN QUIT )
14 -->
15
```

```
SCR # 23
0 ( DISCOMPILER SCR # 4 )
1 : SEARCH-VARIABLE
2 ( CHECK IF WORD IS A VARIABLE )
3 CF @ @ ( PUT CFA ONTO TOP )
4 ' VARIABLE 6 + =
5 ( COMPARE IT WITH VARIABLE POINTER
6 IF
7 .'' VARIABLE VALUE '' GETPFA . CR CR
8 .'' VARIABLE ADDRESS '' PF @ .
9 QUIT THEN ;
10 -->
11
12
13
14
15
```

```
SCR # 24
0 ( DISCOMPILER SCR # 5 )
1 : SEARCH-USER
2 ( CHECK IF WORD IS A USER VARIABLE )
3 CF @ @ ( PUT CFA ONTO STACK )
4 ' USER CFA 6 + =
5 ( COMPARE IT WITH USER POINTER )
6 IF ( PRINT IT )
7 .'' USER VARIABLE VALUE ''
8 CFA EXECUTE @ CR CR
9 .'' USER VARIABLE ADDRESS ''
10 CF @ EXECUTE .
11 CR QUIT THEN ;
12 -->
13
14
15
```

```
SCR # 25
0 ( DISCOMPILER SCR # 6 )
1 : SEARCH-.''
2 ( CHECK FOR A STRING )
3 GETPFA ( PUT PFA ONTO TOP )
4 ' (.'') CFA = ( COMPARE .'' POINTER )
5 IF CR CR ( PRINT STRING )
6 46 EMIT 34 EMIT SPACE ( PRINT .'' )
7 2+PF @ C@ ( READ FIRST BYTE OF STRING )
8 ( WHICH IS THE LENGTH BYTE )
9 1 PF +! ( INCREMENT PF )
10 SWAP TYPE ( TYPE THE STRING )
11 PF @ 1 - C@ PF @ + PF !
12 ( MOVE PF TO END OF STRING )
13 34 EMIT ( PRINT '' AT END OF STRING )
14 SPACE THEN ;
15 -->
```

```
SCR # 26
0 ( DISCOMPILER SCR # 7 )
1 : SEARCH-CONSTANT
2 ( CHECK IF WORD IS A CONSTANT )
3 CF @ @ ( CFA ONTO TOP )
4 ' CONSTANT CFA 10 + =
5 IF ( COMPARE TOP WITH CONSTANT POINTER )
6 ." CONSTANT VALUE
7 GETPFA . CR ( PRINT VALUE OF CONSTANT )
8 QUIT THEN ;
9 0 VARIABLE VP ( TEMP STORE FOR BASE )
10 .H ( HEX PRINT )
11 BASE @ VP ! HEX . VP @ BASE ! ;
12 : DUMP ( HEX DUMP TO VDU )
13 HEX 0 DO CR DUP 2 .R 8 0 DO DUP
14 C@ 3 .R 1+ LOOP 8 +LOOP DROP ;
15 -->
```

```
SCR # 27
0 ( DISCOMPILER SCR # 8 )
1 : SEARCH-MC
2 ( CHECK IF THE WORD IS DEFINED IN M'CODE )
3 CF @ CF @ @ < GETPFA
4 ( IN COLON DEFINITIONS CFA POINTS TO LOW MEMORY )
5 ( IN M'CODE CFA POINTS UP INTO THE PARAMETER FIELD )
6 ( THEREFORE CFA WILL BE GRATER THAN PFA )
7 ' (;CODE) = OR IF ( TEST FOR (;CODE) )
8 ." M'CODE CANNOT DISCOMPILE USE HEX DUMP "
9 CR ." NAME FIELD ADDRESS " NF @
10 DUP .H
11 ." END OF PARAMETER FIELD = " LASTNF @ .H
12 LASTNF @ NF @ - DUMP QUIT THEN ;
13 -->
14
15
```

```
SCR # 28
0 ( DISCOMPILER SCR # 9 )
1 : SEARCH-VOCAB ( SEARCH TO FIND VOCABULARY )
2 CF @ @ ( GET CFA ONTO STACK )
3 ' FORTH CFA @ = IF ( DOES CFA = VOCABULARY POINTER )
4 ." VOCABULARY " NF @ ID. ." IMMEDIATE "
5 QUIT THEN ;
6 : P-WORD ( DEFINITION OF WORD TO PRINT WORDS )
6 GETPFA 2+ NFA ID. 2+PF ;
7 -->
8
9
10
11
12
13
14
15
```

```
SCR # 29
0 ( DISCOMPILER SCR # 10 )
1 : PRINT-WORD ( STRING TOGETHER WORD SEARCHES )
2 2 SPACES
3 DISPOSE-LIT SEARCH-MC
4 SEARCH-USER SEARCH-VOCAB
5 SEARCH-BRANCH SEARCH-CONSTANT
6 SEARCH-VARIABLE SEARCH-.''
7 P-WORD ; (IF WORD NOT FOUND PRINT IT )
8
9
10
11
12
13
14
15
```

```
SCR # 30
0 ( DISCOMPILER SCR # 11 )
1 : DISCOM ( WORD TO DISCOMPILE )
2 PF @ BEGIN ( START AT BEGINNING OF PFA )
3 DISPOSE-LIT (IF FIRST WORD IS 16 BIT INTEGER DISPOSE-LIT )
4 ?TERMINAL IF QUIT THEN ( CHECK FOR BREAK KEY )
5 PRINT-WORD ( DOIT )
6 GETPFA ' ;S CFA = UNTIL ( UNTIL ;S )
7 CR CR DROP ;
8 -->
9
10
11
12
13
14
15
```

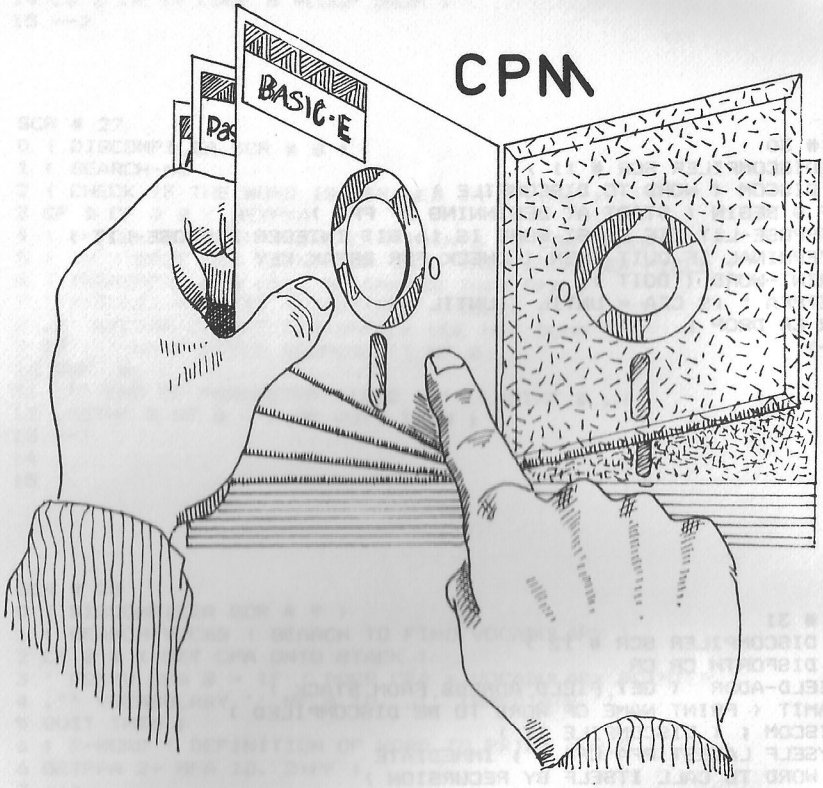
```
SCR # 31
0 ( DISCOMPILER SCR # 12 )
1 : DISFORTH CR CR
2 FIELD-ADDR  ( GET FIELD ADRESS FROM STACK )
3 NAMIT ( PRINT NAME OF WORD TO BE DISCOMPILED )
4 DISCOM ; ( DISCOMPILE IT )
5 MYSELF LATEST PFA CFA , ; IMMEDIATE
6 ( WORD TO CALL ITSELF BY RECURSION )
7 : DIS NF @ LASTNF ! LF @ @ PFA DISFORTH
8 MYSELF ;
9 ;S
10
11
12
13
14
15
```

More FORTH next issue.

CONTROL PROGRAM FOR MICRO-COMPUTERS

The PASCAL programming language is a direct descendant of an earlier language called ALGOL. ALGOL is actually a series of languages (ALGOL 60 and ALGO 68) developed by an international committee, beginning in the late 1950's and finally released in 1960 as ALGOL 60.

The PASCAL compiler, volume 50, introduced last issue has one important omission for applications programmers, namely the REAL, floating point, data type. If you find this omission limits the use of the CP/M library PASCAL then volume 28 will interest you.



CP/M libary volume 28 includes a compiler for a structured language called ALGOL-M. ALGOL-M is similar in concept to BASIC-E, in that it is a p-code type of compiler with REAL numbers of similar accuracy to BASIC-E, with a PASCAL flavour through the inclusion of a CASE statement. ALGOL-M was written by Lt. Mark Moranville while he was studying at the Naval Postgraduate School at Monterey, U.S.A. This compiler was first described in the proceedings of the Second West Coast Fair, U.S.A. in March 1978.

Catalogue description for Volume 28

VOLUME 28

BASIC-E UTILITIES AND GAMES INCLUDING A DATA BASE SYSTEM AND AN ALGOL-LIKE LANGUAGE. UPDATED PROGRAMS FOR THE CACHE MAILLIST.

NUMBER	SIZE	NAME	COMMENTS
		CATALOG.28	CONTENTS OF CP/M GROUP VOL 28
28.1	6K	DATABASE.DOC	DOCUMENTATION FOR INITIAL MODULES OF DATABASE SYSTEM.
28.2	6K	DBENTRY.BAS	D.B. SYSTEM FILE LOAD UTILITY
28.3	7K	DBQUERY.BAS	D.B. SYSTEM QUERY AND UPDATE MODULE
28.4	7K	DBSETUP.BAS	D.B. SYSTEM FILE INITIALIZATION MODULE
28.5	1K	INV.	SAMPLE FILE DEFINITION FOR INVENTORY DATA BASE
28.6	1K	INV.IND	SAMPLE PART # INDEX FOR INVENTORY FILE
28.7	23K	MAILLIST.DOC	UPDATED DOCUMENTATION FOR CACHE MAILING LIST SYSTEM
28.8	11K	MAINT.BAS	UPDATED MAILLIST PROGRAM. HANDLES MORE DATA FIELDS IN FILE.
28.9	11K	REPORT.BAS	IMPROVED MAILLIST REPORT PROGRAM.
28.10	14K	ALGOLM.COM	ALGOLM COMPILER BY LT. MARK S. MORANVILLE-NAVAL POSTGRADUATE SCHOOL. SEE PROCEEDINGS OF SECOND WEST COAST FAIRE, MARCH 1978
28.11	14K	RUNALG.COM	ALGOLM INT FILE INTERPRETER
28.12	9K	ALGINTRO.TXT	ALGOLM INTRODUCTION
28.13	3K	ALGSTART.TXT	ALGOLM RUN INSTRUCTIONS
28.14	2K	COMERR.TXT	ALGOLM COMPILE ERRORS DOCUMENTATION
28.15	2K	RUNERR.TXT	ALGOLM RUN TIME ERRORS DOCUMENTATION
28.16	25K	USRMAN.TXT	ALGOLM USER MANUAL
28.17	3K	ARRAY.ALG	SAMPLE ALGOLM PROGRAM
28.18	1K	BOOLINT.ALG	SAMPLE ALGOLM PROGRAM
28.19	1K	CASETEST.ALG	SAMPLE ALGOLM PROGRAM
28.20	1K	CASETWO.ALG	SAMPLE ALGOLM PROGRAM
28.21	1K	FLYTEST.ALG	SAMPLE ALGOLM PROGRAM
28.22	1K	GOTOTEST.ALG	SAMPLE ALGOLM PROGRAM
28.23	1K	HANOI.ALG	SAMPLE ALGOLM PROGRAM
28.24	2K	LUNAR.ALG	SAMPLE ALGOLM PROGRAM
28.25	2K	PERM.ALG	SAMPLE ALGOLM PROGRAM
28.26	2K	READWORD.ALG	SAMPLE ALGOLM PROGRAM
28.27	1K	RWINT.ALG	SAMPLE ALGOLM PROGRAM
28.28	1K	SIISTR.ALG	SAMPLE ALGOLM PROGRAM
28.29	2K	SORT.ALG	SAMPLE ALGOLM PROGRAM
28.30	1K	STRING.ALG	SAMPLE ALGOLM PROGRAM
28.31	1K	STRTST.ALG	SAMPLE ALGOLM PROGRAM
28.32	1K	BLKTEST.ALG	SAMPLE ALGOLM PROGRAM
28.33	1K	ONEND2.ALG	SAMPLE ALGOLM PROGRAM
28.34	1K	WFILE.OUT	SAMPLE ALGOLM PROGRAM OUTPUT
28.35	1K	WINPUT.INP	DEMO INPUT FILE FOR READWORD.
28.36	1K	RFILE.INP	DEMO INPUT FILE FOR RWINT.
28.37	1K	BLK.INP	DEMO INPUT FILE
28.38	1K	FILE1.INP	DEMO INPUT FILE
28.39	1K	FILE2.INP	DEMO INPUT FILE
28.40	4K	ADM3.MOD	MODS FOR ADM3

ALGOL Language description

Although ALGOL-M was modelled after ALGOL-60, no attempt was made to make it a formal subset of ALGOL-60. This was done intentionally in order to provide a language which would be best suited to the needs of applications programmers using microcomputer systems. However, the basic structure of ALGOL-M is similar enough to ALGOL-60 to allow simple conversion of programs from one language to the other. This was considered particularly important in view of the fact that the standard publication language is ALGOL-60. Therefore, there exists a large source of applications programs and library procedures which can be simply converted to execute under ALGOL-M.

ALGOL-M supports three types of variables: integers, decimals, and strings. Integers may be any value between -16,383 and +16,383. Decimals may be declared with uyp to 18 digits of prevision and strings may be declared as long as 255 characters. The default precision for decimals is ten digits and the default length for strings is ten characters. Decimal and string variable lengths may be integer variables which can be assigned actual values at run-time.

Another form of declaration in ALGOL-M is the array declaration. Arrays may have up to 255 dimensions with each dimension ranging from 0 to +16,383. The maximum 8080 microprocessor address space of 63K bytes limites practical array sizes to something smaller than the maximum. Dimension bounds may be integer variables with the actual values assigned at run-time. Arrays may be of type integer, decimal or string.

Integer and binary coded decimal arithmetic are supported under ALGOL-M. Integers may be used in decimal expressions and will be converted to decimals at run-time. The integer and decimal comparisons of less-than (<), greater-than (>), equal-to (=), not-equal-to (<>), less-than-or-equal-to (<=), and greater-than-or-equal-to (>=) are provided. Additionally, the logical operators AND, OR and NOT are available.

ALGOL-M control structures consist of BEGIN, END, FOR, IF THEN, IF THEN ELSE, WHILE, CASE and GOTO constructs. Function and procedure calls are also used as control structures. ALGOL-M is a block structured language with a block normally bracketed by a BEGIN and an END. Blocks may be nested within other blocks to nine levels. Varibles which are declared within a block can only be referenced within that block or a block nested within that block. Once program control proceeds outside of a block in which a variable has been declared, the variable may not be referenced and, in fct, run-time storage space for that variable no longer exists.

Functions, when called, return an integer, decimal or string value depending on the type of the function. Procedures do not return a value when called. Both functions and procedures may have zero or more parameters which are called by value and both may be called recursively.

The ALGOL-M WRITE statement causes output to the console on a new line. The desired output is specified in a write list which is enclosed in parentheses. String constants may be used in a write list and are characterised by being enclosed in quotation marks. Any combination of integer, decimal and string variables or expressions may also be used in a write list. A WRITEON statement is also available which is essentially the same as the WRITE statement except that output continues on the same line as the output from a previous WRITE or WRITEON statement. When a total of 80 characters have been written to the console, a new line is started automatically. A TAB option may also be used in the write list which causes the following items in the write list which causes the following item in the write list to be spaced to the right by a specified amount.

Console input is accomplished by the READ statement followed by a read list of any combination of integer, decimal and string variables enclosed in parentheses. If embedded blanks are desired in the input for a string variable, the console input must be enclosed in quotation marks. A READ statement will result in a half in program execution at run-time until the input values are typed at the console and a carriage return is sent.

If the values typed at the console match the read list in number and type, program execution continues. If an error as to number or type of variables from the console occurs, program execution is again halted until values are re-entered on the console.

ALGOL-M programs may read data from, or write data to, one or more disk files which may be located on one or more disk drives. When file input or output is desired, the appropriate READ or WRITE statement is modified by placing a filename identifier immediately after READ or WRITE. The actual name of the file may be assigned to the file name identifier when the program is written or it may be assigned at run-time. Various disk drives are referenced by the letters A through Z. A specific drive may be specified by prefixing the actual file name with the desired drive letter followed by a colon. Additionally, if random file access is desired, the file name identifier may be followed by a comma and an integer constant or variable. This integer value specifies the record within th file which is to be used fdor input/output.

Prior to the use of a file name identifier in a READ or WRITE statement, the file name identifier must appear in a file declaration statement. The file name identifier can only be referenced within the same block (or a lower block) as the file declaration. Files are normally treated as unblocked sequential files. However, if blocked files are desired, the record length may optionally be specified in brackets after the file name identifier in the file declaration statement.

ALGOL-M example programs

1. LUNAR.ALG

```
BEGIN
COMMENT UPDATED LUNAR LANDER FROM KILOBAUD AUGUST 78;
DECIMAL F,V,D,B,C;
INTEGER A;
WRITE(TAB 8,"LUNAR LANDER MKII");
WRITE(TAB 8,"++++++++++++++++++");
WRITE(" ");
WRITE("WOULD YOU LIKE INSTRUCTIONS?");
WRITE("1=YES 0=NO");
READ(A);
IF A=1 THEN
  BEGIN
    WRITE("YOU HAVE 120 LBS OF FUEL");
    WRITE("YOU ARE APPROACHING THE LUNAR");
    WRITE("SURFACE AT 50 FT/SEC, AND");
    WRITE("ARE CURRENTLY 500 FT FROM");
    WRITE("THE SURFACE, TO CANCEL");
    WRITE("GRAVITY BURN 5 LB FUEL");
  END;
A:=1;
WHILE A=1 DO
BEGIN
WRITE("HAPPY LANDINGS!!!!");
F:= 120.0; V:=50.0; D:=500.0;
FUEL:WRITE("FUEL ", F);
WRITE("SPEED ",V);
WRITE("DISTANCE ",D);
INPUTBURN:WRITE("ENTER YOUR BURN");
READ(B);
IF B>F THEN GOTO INPUTBURN;
F:=F-B; C:=B-5.0; D:=D-V+C/2.0; V:=V-C;
IF D>0 THEN GO TO FUEL;
IF V>=5.0 THEN
  BEGIN
    WRITE("****!!!CRASH!!!****");
    WRITE("YOU HIT THE MOON AT ", V," FT/SEC");
  END
ELSE
  BEGIN
    WRITE("WELL DONE - YOU LANDED OK");
    WRITE("LANDING SPEED ",V," FT/SEC");
    END;
WRITE("DO YOU WANT ANOTHER TRY?");
WRITE("1=YES, 0=NO");
READ(A);
END;
WRITE("END");
END
```

2. CASETEST.ALG

```
begin
  comment This is a test program to test the functioning of the
          Algol-m case statement;
  while 1=1 do
      begin
        integer num,input,index;
        write("input test value");

        read(input);
        case input of
          begin
           begin
             write("0",input);
           end;
           begin
            %this is case 1%
            write("1",input);
           end;
           begin
           write("2",input);
             for index:=1 step 1 until input do
               begin
                write(index);
              end;
             end;
           write("3"," ",input);
          end;
      end;
end
eof
```

Volume 28 is one of the outstanding disks in the U.S. CP/M library. Not only does it contain ALGOL-M but is also contains further BASIC-E programs, for our business users the database system may be of interest.

Volume 23 of the U.S. CP/M library is reserved for a threaded language compiler wich is similar in concept to FORTH. This language is called STOIC (stack oriented interactive compiler). We have included this volume alongside Volume 28 becuase it also contains software which extends the FORTH idea through the addition of the REAL data type. If you enjoy programming in FORTH then STOIC will be a welcome addition to your software collection. Further additions and improvements to STOIC have been undertaken by programmers in this country and their upgrades are on U.K. CP/M libary disk No. 1. An extensive review of STOIC and its upgrades is published in CPMUGUK Vol. 1. No. 6, May 1982.

Catalogue description for Volume 23

VOLUME 23

STOIC (STACK ORIENTED INTERACTIVE COMPILER)

STOIC HAS THE CAPABILITIES OF A COMPILER, EDITOR, ASSEMBLER, DEBUGGER, LOADER AND OPERATING SYSTEM. IT APPEARS TO BE CONCEPTUALLY SIMILAR TO FORTH IN THE USE OF AN EXTENSIBLE VOCABULARY OF WORDS, WITH THE ABILITY TO "FORGET" DEFINITIONS AND RE-DEFINE OR CREATE NEW WORDS IN TERMS OF PREVIOUSLY LEARNED WORDS.

THE FIRST TWO MODULES, THE BOOTSTRAP AND KERNEL, WERE WRITTEN IN ASSEMBLER MNEMONICS. THEREAFTER THE MODULES WERE WRITTEN IN STOIC WORDS.

THIS PACKAGE IS A STAND-ALONE SYSTEM AND DOES NOT REQUIRE A RESIDENT FDOS, SUCH AS CP/M OR ISIS.

THE MATERIAL HAS BEEN SIGNIFICANTLY RE-ARRANGED AND REVISED SINCE THE CATALOGS WERE FIRST PUBLISHED. IN PARTICULAR THE KERNEL NOW INCORPORATES WORD DEFINITIONS WHICH PERMIT DIRECT OPERATION THROUGH CP/M. STOICCPM.DOC EXPLAINS THE PROCEDURE FOR BRINGING THE SYSTEM UP. IT WAS NECESSARY TO LEAVE CERTAIN FILES OFF BECAUSE OF LACK OF SPACE, AND THESE CAN BE OBTAINED AS VOLUME 23B. THE OVERFLOW VOLUMES WILL ALSO BE ADDED TO A FUTURE REGULAR VOLUME, AND WERE SELECTED AS THE LEAST NECESSARY FOR A CP/M USER.

FILES OF TYPE .STC ARE WRITTEN IN STOIC AND CAN BE LOADED AS DESCRIBED IN THE VOLUME23.DOC

NUMBER	SIZE	NAME	COMMENTS
		CATALOG.23	CONTENTS OF CP/M GROUP VOL 23
		VOLUME23.DOC	COMMENTS ON THIS VOLUME, ADDITIONAL TO STOICCPM.DOC
		STOICCPM.DOC	COMMENTS ON CP/M STOIC. READ TOGETHER WITH THE VOLUME23.DOC
23A.1	13K	ASSEMBL.DOC	STOIC 8080 ASSEMBLER
23A.2	8K	DICT.DOC	STOIC DICTIONARY DEFINITIONS
23A.3	8K	EDIT.DOC	STOIC DISPLAY EDITOR DOC
23A.4	9K	FILES.DOC	STOIC FILE SYSTEM DOC
23A.5	5K	FLOATPNT.DOC	FLOATING POINT DOC
23A.6	2K	INTRRUPT.DOC	INTERRUPT HANDLER DOC
23A.7	38K	KERNEL.ASM	STOIC KERNEL WITH CP/M WORDS IN 8080 ASSEMBLER
23A.8	24K	KERNEL.DOC	STOIC KERNEL DOC
23A.9	46K	STOIC.DOC	OVERVIEW OF STOIC CONCEPTS AND STRUCTURE
23A.10	23K	STOICBAS.STC	BASIC DEFINTIONS
23A.11	4K	STOICD/A.STC	GRAPHICS FOR D/A HARDWARE
23A.12	9K	STOICEDT.STC	EDITOR
23A.13	3K	STOICFFT.STC	FAST FOURIER TRANSFORM
23A.14	8K	STOICFLE.STC	FILE SYSTEM
23A.15	11K	STOICFLT.STC	FLOATING POINT PACKAGE
23A.16	5K	STOICINT.STC	4 BYTE INTEGER ADDITION
23A.17	3K	STOICINP.STC	INTERRUPT PACKAGE
23A.18	3K	STOICMIS.STC	MISCELLANEOUS WORDS
23A.19	2K	STOICSIN.STC	INTEGER SINE AND COSINE
23A.20	2K	STOICSRT.STC	SORT ROUTINE

STOIC floating point document

FLOATING POINT IN ADDED TO STOIC BY LOADING THE FILE "FP". SUBSEQUENT TO LOADING "FP", FLOATING POINT LITERALS ARE RECOGNIZED BY STOIC, AND A SET OF FLOATING POINT WORDS ARE DEFINED. THE FORMAT OF FLOATING POINT NUMBERS IS AS FOLLOWS:

BYTE	CONTENTS	FORMAT
0	SIGN	=0 FOR POSITIVE =80 HEX FOR NEGATIVE
1	EXPONENT	8-BIT 2'S COMPLEMENT
2,3	FRACTION	UNSIGNED BINARY BINARY POINT IS JUST LEFT OF THE MSB

WHEN A FLOATING POINT NUMBER IS ON THE STACK, THE SIGN/EXPONENT WORD IS ON TOP, THE FRACTION IS AT TOP - 1. PRECISION IS BETTER THAN 4 1/2 DECIMAL PLACES. WHEN A FLOATING POINT NUMBER IS STORED IN MEMORY, THE SIGN/EXPONENT WORD IS AT (ADDRESS), THE FRACTION IS AT (ADDRESS+2).

FLOATING POINT LITERALS ARE THE SAME AS INTEGER LITERALS EXCEPT THAT THEY MAY INCLUDE A DECIMAL POINT AND MAY OPTIONALLY BE FOLLOWED BY AN "E" FOLLOWED BY AN EXPONENT. EMBEDDED SPACES WITHIN THE LITERAL ARE NOT ALLOWED. FLOATING POINT LITERALS ARE ALWAYS DECIMAL.

EXAMPLES:

```
1.
2.0E-1
+12.345
 5.0E-3
```

THE FOLLOWING WORDS MAKE UP THE FLOATING POINT PACKAGE:

FCONSTANT	DEFINES A FLOATING POINT CONSTANT
FVARAIBLE	DEFINES A FLOATING POINT VARIABLE

EXAMPLES:

```
        2.71828 'E FCONSTANT
        0.0 'X FVARAIBLE
```

PI — A FLOATING POINT CONSTANT EQUAL TO 3.1416

F@ — PUSHES THE FLOATING POINT NUMBER ADDRESSED BY THE TOP OF THE STACK.

F! — STORES THE FLOATING POINT NUMBER AT TOP - 1 AND TOP - 2 AT THE ADDRESS AT TOP.

```
D2OVER   PUSH A COPY OF TOP-3,TOP-2

D3OVER   PUSH A COPY OF TOP-5,TOP-4

D2UNDER  STORE TOP-1,TOP AT TOP-5,TOP-4

D3UNDER  STORE TOP-1,TOP AT TOP-7,TOP-6

FMINUS   FLOATING POINT NEGATE

FABS     FLOATING POINT ABSOLUTE VALUE

F+       FLOATING POINT ADD

F-       FLOATING POINT SUBTRACT

F*       FLOATING POINT MULTIPLY

F/       FLOATING POINT DIVIDE

F+!      FLOATING POINT ADD TO MEMORY

FLOAT    CONVERT THE INTEGER AT TOP TO FLOATING POINT

INTEGER  TRUNCATE THE FLOATING POINT NUMBER AT TOP-1,TOP TO
         AN INTEGER.

FRAC     PUSH THE FRACTION PART OF THE FLOATING POINT NUMBER
         ON THE TOP OF THE STACK

FMOD     FLOATING POINT MOD FUNCTION.
         COMPUTED AT ARG2*FRAC(ARG1/ARG2)

FLTZ     FLOATING POINT LESS THAN ZERO
FLEZ     FLOATING POINT LESS THAN OR EQUAL TO ZERO
FGTZ     FLOATING POINT GREATER THAN ZERO
FGEZ     FLOATING POINT GREATER THAN OR EQUAL TO ZERO
FEQZ     FLOATING POINT EQUAL TO ZERO
FNEZ     FLOATING POINT NOT EQUAL TO ZERO

FLT      FLOATING POINT LESS THAN
FLE      FLOATING POINT LESS THAN OR EQUAL TO
FGT      FLOATING POINT GREATER THAN
FGE      FLOATING POINT GREATER THAN OR EQUAL TO
FEQ      FLOATING POINT EQUAL TO
FNE      FLOATING POINT NOT EQUAL TO

FSQRT    FLOATING POINT SQUARE ROOT
FSIN     FLOATING POINT SINE (ARGUMENT IN RADIANS)
FCOS     FLOATING POINT COSINE (ARGUMENT IN RADIANS)
FATAN    FLOATING POINT ARCTANGENT (SINGLE ARGUMENT, RESULT IN RADIANS)
FLN      FLOATING POINT LOG BASE E
FLOG2    FLOATING POINT LOG BASE 2
FLOG10   FLOATING POINT LOG BASE 10
FEXP     FLOATING POINT EXPONENTIAL BASE E
2.0**    FLOATING POINT EXPONENTIAL BASE 2
10.0**   FLOATING POINT EXPONENTIAL BASE 10
```

F= FLOATING POINT E-FORMAT PRINT
F? PRINT FLOATING POINT NUMBER ADDRESSED BY TOP OF STACK

FLITERAL
ACCEPTS A STRING ARGUMENT AT TOP AND ATTEMPTS TO CONVERT THE STRING TO A FLOATING POINT NUMBER. IF SUCCESSFUL, A -1 IS LEFT AT TOP, AND THE VALUE IS RETURNED AT TOP - 1 AND TOP - 2. IF NOT SUCCESSFUL, A ZERO IS RETURNED AT TOP.

<LSQ INITIATE A LINEAR LEAST SQUARES FIT.

LSQ ACCEPTS TWO FLOATING POINT NUMBERS, A Y COORDINATE AT TOP AND AN X COORDINATE BENEATH. THIS POINT (X,Y) IS PROCESSED.

LSQ> WHEN "LSQ" HAS BEEN CALLED FOR ALL INPUT POINTS, "LSQ>" IS CALLED, RETURNING THE SLOPE AND INTERCEPT OF THE LINEAR LEAST SQUARES FIT TO THE GIVEN POINTS. (SLOPE RETURNED AT TOP, Y-INTERCEPT AT TOP - 1)

FRAND RETURNS A RANDOM FLOATING POINT NUMBER UNIFORMLY DISTRIBUTED BETWEEN 0.0 AND 1.0.

Example Program: STOICSRT.STC

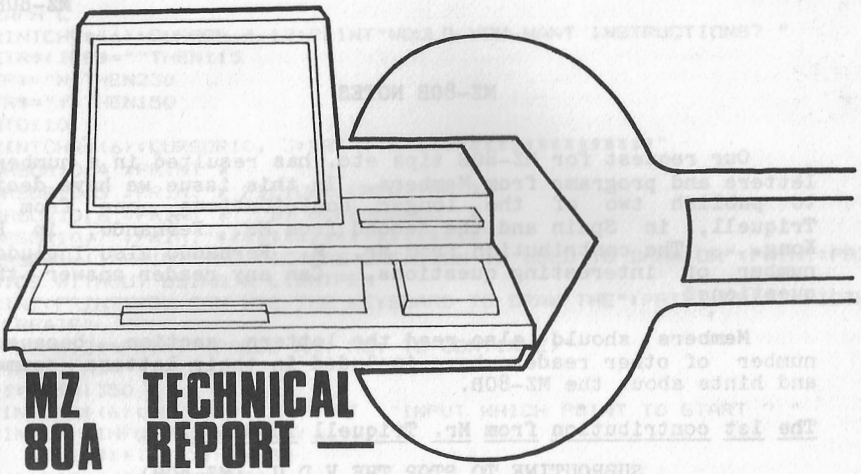
```
% RADIX EXCHANGE SORT
% SORTS A LINEAR ARRAY OF 1-WORD KEYS
% J. SACHS 2/14/77

% ARRAY SIZE SORT
% ARRAY IS ADDRESS OF ARRAY TO BE SORTED
% SIZE IS ARRAY SIZE IN BYTES
% ON RETURN, ARRAY IS SORTED IN ASCENDING ORDER

RADIX @ OCTAL

0 'MASK CONSTANT
0 'SELF CONSTANT

'SORT1 : DDUP 2- LT IF
  DDUP BEGIN
   OVER @ MASK AND EQZ IF
    OVER 2+ 2UNDER ELSE
    2- DUP @ MASK AND EQZ IF
     DDUP XCHG OVER 2+ 2UNDER
    THEN
   THEN DDUP EQ
  END
  MASK 1 NE IF
   MASK U2/ () MASK !
   3OVER OVER SELF EXEC
   OVER 3OVER SELF EXEC
   MASK 2* () MASK !
  THEN 2DROP
 THEN 2DROP
;

() SORT1 () SELF !

'SORT : OVER + 100000 () MASK ! SORT1 ;

RADIX !
;F
```

We will supply CP/M library Volumes 23 and 28 at the followoing costs:

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

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

In our next issue, notes on the CP/M library STAGE2 macro processor and commercial PASCAL compilers.



MZ 80A TECHNICAL REPORT —

Disk BASIC: Release of a new version of SA-6510

General: Please note release of a new version of Disc BASIC (SA-6510) to prevent occurrence of ERROR 50 during read/write of the random file.

Contents: There may be a possible occurrence of ERROR 50 when random file access is performed with DISK BASIC (SA-6510) in use, depending on the use of the floppy disk interface card (MA8AFI).

Cause: As it is due to time constant variation in the ready signal generator circuit on the interface card and timing relation between the index signal and the drive select signal, the problem will be solved by means of software.

Measures: Revision will be met in a part of the program (SA-6510) from the new production. (S2H31B ¢)
New version of program will be indicated with the revised data on the display.

(1) BOOT the SA-6510.

(2) POKE $12DA,$CC
(3) POKE $12DB,$57
Above are executed.

(4) Jump to the monitor after the entry of MON

(5) The revised date "AUG. 20 '82" will be displayed after entry of J 12A0

MZ-80B NOTES

Our request for MZ-80B tips etc. has resulted in a number of letters and programs from Members. In this issue we have decided to publish two of the longer contributions, one from Mr. Triquell, in Spain and the second from Mr. Fernando, in Hong Kong. The contribution from **Mr. M. Fernando** also includes a number of interesting questions. Can any reader answer these questions?

Members should also read the letters section, because a number of other readers have included in their letters comments and hints about the MZ-80B.

The 1st contribution from Mr. Triquell in Spain.

SUBROUTINE TO STOP THE V.D.U. (MZ-80B)

Calling to this subroutine during the execution of a program, e.g. during the execution of a FOR-NEXT loop, and pressing the space key, you may stop and start the V.D.U.

```
4999 REM SUBROUTINE FOR STOP/START THE V.D.U.
5000 F=0:TI$="000000"
5010 GETF$:IFF$<>""THENF=1
5020 IF(F=0)*(TI$<"000001")THEN5010
5030 IFF=1THENGETF$:IFF$=""THEN5030
5045 RETURN
```

Modifying the value of TI$ in line 5020, you may change the execution speed of the program.

The 2nd contribution of Mr. Fernando, in Hong Kong

I attach two graphic programmes for the Sharp MZ-80B. The second is a pattern programme which can draw lines and circles, and thus sketch on the screen.

I shall be grateful if any reader would know of a method of saving on disc, any picture drawn on the graphic screens.

I would also like to know the following:

1) How to append programmes on the MZ-80B (Disk & Tape).

2) Is there any method or program to send printer control codes under CP/M. e.g. for the use of double size or reduced characters.

M. FERNANDO
HONG KONG

```
 100 REM
 105 GRAPH C
 110 PRINTCHR$(6):CURSOR 4,12:PRINT"WOULD YOU WANT INSTRUCTIONS? "
 115 GETR$:IFR$=""THEN115
 120 IFR$="N"THEN230
 130 IFR$="Y"THEN150
 140 GOTO110
 150 PRINTCHR$(6):CURSOR10, 3:PRINT"********************"
 160 CURSOR10,4 :PRINT"*                  *"
 170 CURSOR10,5 :PRINT"* PICTURE CREATION *"
 180 CURSOR10,6 :PRINT"*    BY M.FERNANDO  *"
 190 CURSOR10,7 :PRINT"********************"
 200 CURSOR1,11:PRINT"THE PICTURE CREATION IS USED TO DRAW OR":PRINT:PRINT"CREAT
E GRAPHICS WITHOUT USING A LIGHTPEN"
 210 PRINT:PRINT"YOU CAN USE THE KEYBOARD TO DRAW THE":PRINT:PRINT"PICTURE YOU W
ANT TO CREATE"
 220 CURSOR6,24:PRINT"<PRESS ANY KEY TO CONTINUE>"
 225 GETH$:IFH$=""THEN225
 226 A=1:GOTO1350
 230 PRINTCHR$(6):CURSOR0,12:PRINT  "INPUT WHICH POINT TO START ? "
 235 PRINT:    INPUT" X,Y ";X,Y
 240 IF   (Y>199)+(Y<1)THEN230
 250 IF(X>319)+(X<1)THEN230
 284 PRINTCHR$(6)
 285 GRAPHC,I1,O1
 290 P1=X:P2=Y:SETP1,P2:GOTO330
 300 GETB$:IFB$<>CHR$(32)THEN350
 330 GETA$:IFA$=""THEN330
 350 IFA$="8"THEN610
 360 IFA$="4"THEN660
 370 IFA$="6"THEN710
 380 IFA$="2"THEN760
 390 IFA$="7"THEN810
 400 IFA$="9"THEN920
 410 IFA$="1"THEN1030
 420 IFA$="3"THEN1140
 430 IFA$="S"THEN GOTO230
 440 IFA$="E"THEN END
 450 IFA$="X"THEN1230
 460 IFA$="Z"THEN1250
 470 IFA$="C"THEN1290
 480 IFA$="A"THEN GRAPHO:GOTO1380
 570 SETP1,P2:GOTO300
 580 REM
 590 REM** UP **
 600 REM
 610 P2=P2-1:SETP1,P2:T=P2:IFT=OTHENT=199:P2=T
 620 GOTO300
 630 REM
 640 REM** LEFT **
 650 REM
 660 P1=P1-1:SETP1,P2:T=P1:IFT=OTHENT=319:P1=T
 670 GOTO300
 680 REM
 690 REM** RIGHT **
 700 REM
 710 P1=P1+1:SETP1,P2:T=P1:IFT=319THENT=0:P1=T
 720 GOTO300
 730 REM
 740 REM** DOWN **
 750 REM
 760 P2=P2+1:SETP1,P2:T=P2:IFT=199THENT=0:P2=T
 770 GOTO300
 780 REM
```

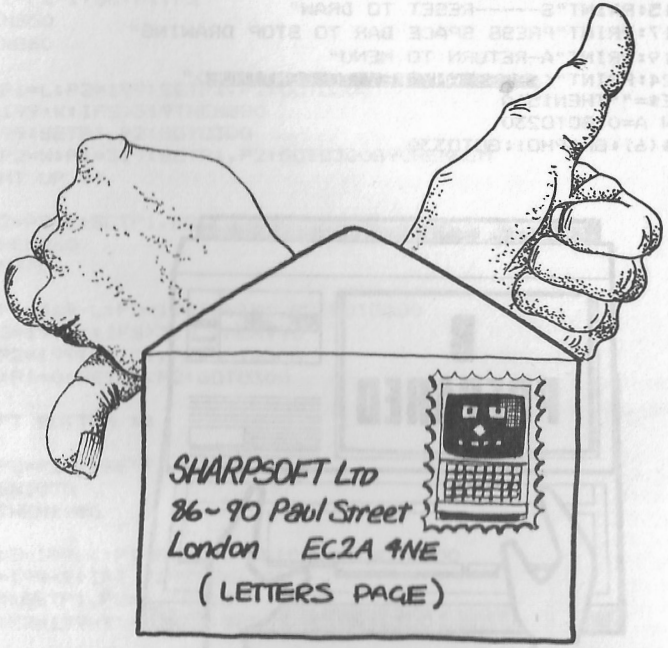
```
790 REM** LEFT TOP **
800 REM
810 P1=P1-1:P2=P2-1:SETP1,P2
820 IFP1=0THEN850
830 IFP2=0THEN860
840 GOTO300
850 L=199-P2:P1=L:P2=199:SETP1,P2:GOTO300
860 K=P1-1:S=199+K:IFS>319THEN880
870 P1=S:P2=199:SETP1,P2:GOTO300
880 W=319-P1:P2=W:P1=319:SETP1,P2:GOTO300890REMREM
900 REM** RIGHT UP **
910 REM
920 P1=P1+1:P2=P2-1:SETP1,P2
930 IFP1>319THEN960
940 IFP2<1THEN970
950 GOTO300
960 L=199-P2:P1=319-L:P2=199:SETP1,P2:GOTO300
970 K=319-P1:S=199+K:IFS>319 THEN990
980 P1=319-S:P2=199:SETP1,P2:GOTO300
990 W=P1:P2=W:P1=0:SETP1,P2:GOTO300
1000 REM
1010 REM** LEFT BOTTOM **
1020 REM
1030 P1=P1-1:P2=P2+1:SETP1,P2
1040 IFP1=0THEN1070
1050 IFP2=199THEN1080
1060 GOTO300
1070 X=199-P2:E=199-X:P1=E:P2=0:SETP1,P2:GOTO300
1080 T=P1-1:Z=199+T:IFZ>319THEN1100
1090 P1=Z:P2=0:SETP1,P2:GOTO300
1100 Y=319-P1:P2=199-Y:P1=319:SETP1,P2:GOTO300
1110 REM
1120 REM** RIGHT BOTTOM **
1130 REM
1140 P1=P1+1:P2=P2+1:SETP1,P2
1150 IFP1=319THEN1180
1160 IFP2=199THEN1190
1170 GOTO300
1180 I=199-P2:H=199-I:P1=319-H:P2=0:SETP1,P2:GOTO300
1190 B=319-P1:G=199+B:IFG>319THEN1210
1200 P1=319-G:P2=0:SETP1,P2:GOTO300
1210 R=319-P1:F=319-R:P2=199-F:P1=0:SETP1,P2:GOTO300
1220 REM
1230 A=P1:S=P2:GOTO300
1240 REM
1250 LINEP1,P2,A,S:GOTO300
1260 REM
1270 REM** DRAW CIRCLE **
1280 REM
1290 CURSOR12,12:INPUT"ENTER THE RADIUS ";R:CURSOR 14,15:PRINT"WORKING....."
1300 M=P1+R:N=P1-R:B=P2+R:V=P2-R
1310 IF(M>319)+(N<0)+(B>199)+(V<0)THEN1290
1320 FORZ=0TO㎡2STEP1/R
1330 SETP1+R*COS(Z),P2+R*SIN(Z)
1340 NEXTZ:PRINTCHR$(6):GOTO330
1350 REM
1360 REM** MAIN MENU **
1370 REM
```

```
1380 PRINTCHR$(6):CURSOR12,1:PRINT"*** MAIN MENU ***"
1390 CURSOR2,3:PRINT"8-----UP":CURSOR20,3:PRINT"4-----LEFT"
1400 CURSOR2,5:PRINT"6-----RIGHT":CURSOR20,5:PRINT"2-----DOWN"
1410 CURSOR2,7:PRINT"7-----LEFT TOP":CURSOR20,7:PRINT"9-----RIGHT TOP"
1420 CURSOR2,9:PRINT"1-----LEFT BOTTOM":CURSOR20,9:PRINT"3-----RIGHT BOTTOM"
1430 CURSOR2,11:PRINT"X-----FIX POINT":CURSOR20,11:PRINT"C-----DRAW CIRCLE"
1440 CURSOR2,13:PRINT"E-EXIT"
1450 CURSOR2,15:PRINT"S-----RESET TO DRAW"
1452 CURSOR2,17:PRINT"PRESS SPACE BAR TO STOP DRAWING"
1455 CURSOR2,19:PRINT"A-RETURN TO MENU"
1500 CURSOR5,24:PRINT"<PRESS ANY KEY TO CONTINUE>"
1510 GETE$:IFE$=""THEN1510
1515 IFA=1THEN A=0:GOTO230
1520 PRINTCHR$(6):GRAPH01:GOTO330
```


```
  4 CURSOR9,10:PRINT"STARBURST PROGRAMME BY"
  5 CURSOR14,12:PRINT"M.FERNANDO"
  6 CURSOR 13,18:PRINT"PRESS ANY KEY"
  7 GETA$:IFA$=""THEN7
  8 PRINTCHR$(6)
 10 GRAPHC
 20 V=.8+3*RND(1)
 50 GRAPH01
 60 SET140,95
 70 FORTHETA=0TO500*VSTEPV
 80 R=THETA/V
 90 X=140+R*COS(THETA)
100 Y=95+R*SIN(THETA)
110 X1=140+R*COS(THETA+V)
120 Y1=95+R*SIN(THETA+V)
150 IF(X>319)*(Y>191)*(X1>319)*(Y1>191)THEN190
160 IF(Y<0)+(Y1<0)THEN190
170 LINEX,Y,X1,Y1
180 NEXT THETA
190 FORT=1TO1000:NEXT
200 GOTO10
```

M E M B E R S' L E T T E R S

DEAR SUN, DEAR SIRS, DEAR SHARPSOFT, DEAR MIKE, OH DEARIE ME!!



Dear Sirs,

Thank you for the excellent Sharpsoft User Notes, how nice it was to see a section for the MZ-80B. As a CP/M User I think the review of user library software and the CP/M notes are an excellent idea.

It was with some concern that I read the letter from Mr. M.J. Stanley about the problems he had. First I feel I must point out that the "Utilities program was not created by me, I only altered and linked the programs in issue 3 of S.U.N.

To answer Mr. Stanley's problem with the Utilities program, his comment about the Poke and Peek addresses is correct. The V 1.1 of BASIC was issued to correct the Copy command fault in the original version, a consequence of this is the start of program text has changed. I have enclosed a modification listing and hope this will be of help to Mr. Stanley and those who are using this version, also those using SB-6510 disk BASIC for which modifications are included.

Mr. Stanley's other problem is more interesting. I tried out the original back-up routine with V 1.1, this was successful. However when booting is completed BASIC reports an error 13. To take account of this and the new text address I have re-worked the original and enclose a new listing and instructions for this. This should produce a back-up copy of the master tape which can be used by the IPL to boot the system. If this does not solve Mr. Stanley's problem perhaps Sharpsoft could pass on my address so he may contact me directly so that I may be able to offer further assistance.

One contribution for MZ-80B users, a way to disable the Break and Shift-Break functions when running BASIC programs. This was done for a friend who works at a well known store (now selling microcomputers), to prevent the stopping of demonstration programs that were being used. When processing each line of a program BASIC calls a routine in the monitor to test if the shift-break is asserted. This can be removed by making the first instruction of the test routine a return ($C9). To do this POKE 1378, 201 (POKE $0562, $C9). This will now make program interruption by shift-break ineffective. To restore the shift-break POKE 1378, 205 (POKE $0562, $CD). During INPUT instructions pressing break will stop the program, this can be prevented by changing the value returned by the break key to a NULL. To change the break key POKE 3431, 00 (POKE $0D67, $000), to restore the break key to normal POKE 3431,11 (POKE $0D67, $0B). Incidentally you may change any key to any code by changing the values in the table which starts at 3400 or $0D48 if you prefer hex, be careful there is nothing more infuriating than a scrambled keyboard. This should work with all versions of BASIC that are using the SB-1510 monitor to which the changes (POKES/PEEKS) are made. Floppy-DOS users do not need to do this of course as the BASIC compiler has the ON BRKEY GOTO and OFF BRKEY commands. I hope the above is of interest and help.

R.A. WYMARK
BIRMINGHAM

****

BASIC Utilities modifications

(From BASIC SB-5510 V 1.0)

Line No.	Change	for V 1.1	for disk BASIC
60430	A=20572	A=20764	A=26460
60930	L2=20571	L2=20763	L2=26459
61300	L2=20571	L2=20763	L2=26459
61340	A=20572	A=20764	A=26460
61360	L2=20571	L2=20763	L2=26459
61430	AL=92	AL=28	AL=92
61430	AH=80	AH=81	AH=103
61560	D=20571	D=20763	D=26459

For V 1.0 BASIC text starts at 20572 or $505C hex.

For V 1.1 BASIC text starts at 20764 or $511C hex.

For SB-6510 V 1.0 disk BASIC text starts at 26460 or $675C hex.

SYSTEM TAPE BACK-UP.

Enter the monitor level with the command "MON". Then using the memory change command "M" enter the following at address $E000 hex.

Address	hex data	Mnemonic
$E000	21 00 00	LD HL,0000
$E003	11 00 80	LD DE,$8000
$E006	01 1C 51	LD BC,$511C
$E009	ED B0	LDIR
$E00B	C3 00 00	JP $000

Exit the memory change mode and jump to the routine you have just created by entering "J" and then the address $E000. This short program will put a copy of the monitor and BASIC interpreter at memory addresses 32768 or $8000 hex to 53532 or $D11C hex and then return to the monitor. Now use the "M" command and change the byte at address $80AE to $C3 this is a jump instruction the following two bytes are $14, $49 which is the address for the jump, if you are using the V 1.0 change these two bytes to $20, $12 (BASIC cold start address), exit the memory change mode. Now save a copy on a fully rewound tape using the "S" command and responding with the start address $8000 and the end address $D11D. Just enter a carriage-return in response to the jump address request. After writing you may rewind and verify the tape if you wish.

You now have a copy of the monitor and BASIC interpreter which should successfully function when the system is powered-up or when you re-boot.

Dear Sharpsoft,

I have recently fitted an external volume control to my Sharp MZ-80K 48K computer. It took my father and I 5-6 hours but we spent time tucking wires into the corners and taping them there. Those 5-6 hours were well worth it! The cost is about 90p depending what sort of potentiometer and knob and where you buy them. The job means taking quite a lot of housing off and undoing many screws. 19 to be exact! If you publish this in the next User Notes here are the step by step instructions for anyone who would like to pursue this venture.

Buy a 4.7-5k ohm linear potentiometer. Buy a knob for it, but make sure it will fit where you are putting it on the computer. Mine is above the L.E.D. by the keypad.

1. Switch the computer off, and disconnect from the mains supply.

2. Carefully turn the computer upside down, and remove the four screws that secure the bottom plate.

3. Lift the bottom plate up, and remove the two silver screws that hold up one end of the support stay.

4. Unplug the four flat cable connectors which connect the printed circuit board to the TV & cassette, you do not need to mark them as long as you fit them back carefully (the two connectors differ by 1 pin only!)

5. Close the bottom plate and remove the four black screws which secure the rear support hinge.

6. Lift away the black bottom part and the printed circuit board.

7. Remove the five outside screws securing the back of the T.V. housing.

8. Remove the four inside screws, and the housing should be removable by sliding it to the rear.

9. Unsolder the volume control potentiometer on the printed circuit board, and replace with wires long enough to reach the computer case.

10 Decide on a location for your control. If on the outer casing be careful not to damage anything in the interior and also not to let the thin metal buckle.

11. Fit your potentiometer and solder the wires on.

12. Re-assembler your Computer.

Be careful and check up first because this may invalidate your guarantee.

J.W. LAMBERT
MIDDLESEX

Care should always be taken when making any hardware modifications as accidental damage could end up being very costly to rectify!

EDITOR

****

Dear Sir,

With reference to Page 43 of your latest S.U.N. You state that you cannot increase the memory of the MZ-80K without probably rebuilding the main Board.

For your information I have attached a leaflet from Watford Electronics giving details of their 'ULTIMUM' Interface System. I have contacted Mr. Harris at Watford Electronics and supplied him with full circuit Diagrams of the MZ-80K and he stated that with a couple of small mods the ULTIMUM can be used with the MZ-80K. It is probable that it can also be used with the MZ-80A and B but as I do not have the circuits for these I cannot find out for you.

I do not know if you will allow it but it may assist MZ-80 users to publish this info in your next S.U.N. as there must be many members who wish to increase the versatility and capability of their machines. I will be purchasing the 'ULTIMUM' System if and when cash allows. I have nothing whtever to do with Watford electronics, I am in fact a Police Officer so I haven't any axe to grind. Hoping that this info can be of use to you and Group Members.

B. HARVEY
LINCS

****

Dear Editor,

I have an MZ-80K with the Knight Commander enhancements to SP-5025 BASIC. My printer is an Epson MX-80F/T2. When patched in using the MHA BASIC patch, the RENUMBER facility in K/C would not work. However this can be corrected by the following pokes:-

```
POKE15983,0     POKE15984,74
POKE16081,0     POKE16082,74
POKE16355,0     POKE16356,74
POKE16494,0     POKE16495,74
```

This re-sets the RENUMBER routine to the new address at the start of the first line in BASIC.

Could you please tell me if you are able to supply an annotated listing of Sharp Cassette BASIC 5025 and if so what is its cost.

R. HARGREAVES
CUMBRIA

We have been in contact with Sharp Electronics (UK) Ltd., and are unable to sell the annotated listing of Sharp SP-5025 BASIC, however, we are currently negotiating to have important portions of the BASIC listing reproduced in future Issues of the User Notes.

EDITOR

Dear Sir,

I have greatly enjoyed the fig-FORTH included with the 1982 Notes but early on became frustrated witht he speed of using tapes as I have become used to discs on BASIC. I have therefore adapted your version of FORTH to run with ZenDOS to provide disc FORTH. What I have failed to do however is to make your FORTH handle small letters. I have amended KEY to change location 1170H but I cannot get this to run. Can someone help please?

R.G. WINDRED
LONDON W.13.

****

Dear Sharpsoft,

Thank you very much for the speedy delivery of the 1982 Newsletters/User Notes and Fig-FORTH package.

Do you know of any references to methods of obtaining a floating point routine within FORTH? If so I would be very interested.

P. BOLTON
ESSEX

A complete floating point package for fig-FORTH was published in the September 1982 issue of Dr. Dobbs Journal. The full reference is:

Forth Floating-Point Package,
by A.J. Monroe,

Dr. Dobb's Journal, Number 71,
September 1982
Page 16.

EDITOR

****

Dear Sir,

Your 'Minor, but significant, bug...' mentioned on Page 2, Issue 6 took you 3 pages of text to explain!

The query on Page 41 by C. Choi regarding the 'PRINTER OFF' message remains unanswered.

I'm afraid that your FORTH package was not sufficiently investigate in depth before release - it often happens!

However having 'tried it out on we dogs', having aroused our interest and reponse you must have sufficient information to enable you to withdraw the present FORTH tapes and introduce a corrected version, even if you charge for the new tapes.

The potential of FORTH is too, valuable for loyal 'SHAR' users of MZ-80K to be left in suspended animation.

DR. W.F. COXON
GWENT

The SHARPSOFT tape version of fig-FORTH came about because we felt that a language which is in the public domain should be made available to hobbyists either FREE or at very low cost. The package was the result of nearly six months work by a friend and myself. Obviously any major software project like this will have bugs - it is almost impossible to field trial every aspect of a package.

FORTH being an extensible language does allow bugs like the one on the EDITOR screen to be fixed by the user. Being able to fix such bugs does pre-suppose that users are able to program in FORTH. After learning the basics of FORTH nearly all new FORTH programmers have been able to make changes to the package for themselves - an example are the EDITOR and VLIST fixes. A letter from one user M.P. Bennett (included in this issue) demonstrates this learning process.

A number of readers are beginning to program in FORTH. Solutions to bug fixes are also being sent in regularly by readers. Eventually we hope to publish all such fixes in the User Notes. By correcting errors are they are found we suggest those Members who are interested in FORTH should patch their own tapes and regenerate upgraded versions using the extensible features of FORTH.

EDITOR

****

Dear Sirs,

I enclose with my Subscription, the listing of some screens I have developed, they may be of interest to some other FORTH users. Screen 16 contains a simple way to get all the IDs to print straight, however it stops you printing lower case characters; it also contains a patch that makes empty buffers pack-out with spaces so the screens type quicker and the way I wrote my own dump word.

Screen 17 is a bit more interesting. This is a crude Screen Editor, it copies an old 'basic' line from the Screen using he routine in the monitor - treats the first number as the line no. and places the text on the page pointed to by prev. Its intended use is, for example: 6 LIST ED the cursor appears at the bottom of the listed Screen and can be moved up and around with the cursor keys and by means of insert and delete the lines can be edited - each line must be 'sent' by typing a return. To exit - either type a blank line or a shift/break.

Screens 18, 19 and 20 make up a simple disassembler that can decode Words - **Warning** some words do not terminate. I suggest this is used with figures or the Shift/break.

Screen 21 shows how the Deforth was worked out by printing for each word the contents of the pfa, cfa and the position of the cfa.

Screen 22 defines two user words 'SET' and 'RESET' these take two parameters X Y... and set or reset the pixel at that point. **Warning** they do no range checking.

Screen 23 defines two user words 'TEMPO' and 'MUSIC' which

act the same way as the basic commands. Tempo takes a single no. from the stack. This is used for the tempo. Music " is used like ." and sounds like the music in the basic book.

Screen 24 defines TUNE! and TIME@ and runs the real time clock. TIME! takes three nos. from the stack.

H M S in 24 hour notation and TIME@ puts them back.

Screen 25 defines a different word which is used as in TI$ calc. I prefer this one since if when you are looking for a screen it prints the no. of any other Screen it passes that is not the one you are looking for. Hence, if you miss a Screen you can quickly rewind and get it.

May I suggest that no one does a cold or warm restart after loading Screen 16 or Screen 25 as the FORTH system will crash, however if you load the FORTH Editor (the line editor) you will be O.K.

P. SEILLY
MIDDX

```
SCR #16
  0 ( P-EMIT & DUMP )
  1 FORTH DEFINITONS HEX
  2 : P-EMIT 7F AND EMIT ;
  3 ( USE IN TYPE )
  4 ' P-EMIT CFA 1C0E ! HERE FENCE !
  5 ( PATCH TO INPROVE EMPTY-BUFFERS )
  6 ' BLANKS CFA 24E7 !
  7 : DUMP OVER + SWAP
  8    DO CR I . I 8 + I
  9     DO I C@ 3 .R
 10      ?TERMINAL IF QUIT THEN LOOP
 11     8 +LOOP ;
 12 -->
 13
 14
 15
SCR #17
  0 ( ED - A SCREEN EDITOR )
  1 CREATE M0003 C3D1 , 0003 , SMUDGE
  2 : ED
  3      BEGIN PAD 1245 OVER M003 DUP
  4        C@ DUP 0D = SWAP 1B = OR
  5        IF DROP QUIT ENDIF
  6        BEGIN DUP C@ 20 = WHILE 1+
  7        REPEAT DUP 1 - 0 SWAP 0 SWAP
  8        (NUMBER) SWAP DROP ROT SWAP
  9        DUP ROT = 0 ?ERROR 1+ SWAP
 10        DUP 0< OVER F > OR 17 ?ERROR
 11        C/L * PREV @ + 2+ DUP C/L + SWAP
 12        DO DUP C@ DUP 0D = IF DROP 20
 13         I C! ELSE I C! 1+ ENDIF
 14        LOOP DROP UPDATE AGAIN ;
 15 -->
```

UNTIL ?

```
SCR #18
  0 ( SIMPLE DOEN LOADERS )
  1 : W. DUP DUP . @ 2+ NFA ID. 2+ CR ;
  2 : N. DUP DUP . @ . 2+ CR ;
  3 : O. DUP DUP . @ OVER + . 2+ CR ;
  4 : T. DUP . COUNT 2DUP TYPE + CR ;
  5 -->
  6
  7
  8
  9
 10
 11
 12
 13
 14
 15


SCR #19
  0 ( DOWN LOADER )
  1 : DOWNLOADER BEGIN DUP W. SWAP @
  2   DUP 1292 = IF DROP O. 0 ELSE
  3   DUP 127A = IF DROP O. 0 ELSE
  4   DUP 12E2 = IF DROP O. 0 ELSE
  5   DUP 12A8 = IF DROP O. 0 ELSE
  6   DUP 1B0B = IF DROP W. 0 ELSE
  7   DUP 1C5B = IF DROP T. 0 ELSE
  8   DUP 1256 = IF DROP N. 0 ELSE
  9   DUP 1B82 = SWAP 1547 = OR
 10   THEN THEN THEN THEN
 11   THEN THEN THEN
 12   ?TERMINAL OR UNTIL DROP ;
 13 -->
 14
 15
SCR #20
  0 ( DEFORTH - SIMPLE
  1 : DEFORTH CR
  2   -FIND 0= 0 ?ERROR DROP DUP CFA @
  3   DUP 1753 = IF ." CONSTANT "
  4   DROP ? ELSE
  5   DUP 176D = IF ." VARIABLE "
  6   DROP . ELSE
  7   DUP 1BCC = IF ." <B DOES> "
  8   DROP . CR @ DOWNLOADER ELSE
  9   DUP 177F = IF ." USER VAR "
 10   DROP C@ 4F80 + . ELSE
 11        1711 = IF DOWNLOADER ELSE
 12   ." MACHINE CODE " .
 13   THEN THEN THEN THEN THEN ;
 14   -->
 15
```

```
SCR #21
  0 ( LST - LOOK UP )
  1 : LST HEX CONTEXT @ @ CR
  2   BEGIN DUP PFA DUP @ 5 .R
  3   DUP CFA DUP @ 5 .R 5 .R
  4   DUP IF NOOP ELSE QUIT ENDIF
  5   ?TERMINAL IF QUIT THEN
  6   AGAIN ;
  7 ;S
  8
  9
 10
 11
 12
 13
 14
 15
```

```
SCR 22
  0 ( SET & REST )
  1 FORTH DEFINITIONS HEX
  2 : CARRAY
  3 <BUILDS 0 DO C, LOOP DOES> + ;
  4 8 4 2 1 4 CARRAY BIT
  5 : SET-BIT
  6  SWAP DUP + + BIT C@ OR ;
  7 : RESET-BIT
  8  SWAP DUP + + BIT C@ -1 XOR AND ;
  9 : GET-BYTE
 10  2 /MOD ROT 2 /MOD ROT 28 * + D000 +
 11  DUP C@ DUP F0 < IF DROP F0 THEN
 12  ROT >R ROT R> ;
 13 : RESET GET-BYTE RESET-BIT SWAP ! ;
 14 : SET   GET-BYTE   SET-BIT SWAP ! ;
 15 -->
```

```
SCR 23
  0 ( TEMPO & MUSIC" )
  1 : TEMPO F AND 8 SWAP - 119E ! ;
  2 CREATE M0030 C3D1 , 0030 , SMUDGE
  3 : (MUSIC") 1245 R> DUP C@ OVER 2+
  4   + >R 1+ M0030 ;
  5 : MUSIC" 22 STATE @
  6   IF COMPILE (MUSIC") WORD HERE
  7    C@ 1+ ALLOT C8 C,
  8   ELSE WORD HERE DUP C@ 1+ +
  9    C8 SWAP ! 1245 HERE + M0030
 10   ENDIF ; IMMEDIATE
 11 -->
 12
 13
 14
 15
```

```
SCR 24
  0 ( TIME! & TIME@ )
  1 CREATE M0033 DIF1 , 33C3 , 00 C,
  2   SMUDGE
  3 CREATE M003B 3BC3 , 00 C, SMUDGE
  4 : TIME! ROT DUP C <
  5   IF  0 119B !
  6   ELSE C - 100 119B ! ENDIF
  7   3C * ROT + 3C * + 1245 SWAP 119B @ M0033 ;
  8
  9 : TIME@ 1243 M003B DROP 0 3C
 10   U/ 0 3C U/ 119B C@ C * +
 11   ROT ROT SWAP ;
 12 ;S
 13
 14
 15
```

```
SCR 25
  0 ( P-T&SCALC )
  1 CREATE M0027 27C3 , 0 C, SMUDGE
  2 : P-T&SCALC
  3   BEGIN 1245 M0027 DUP 10F1 @
  4   - ?TERMINAL 1 - AND
  5   WHILE CR ." FOUND SCR# " 10F1 ? CR
  6   REPEAT DROP ;
  7   ( USE IT IN R/W )
  8   ' P-T&SCALC CFA 2705 !
  9   HERE FENCE !
 10 ;S
 11
 12
 13
 14
 15
```

*** 1ST LETTER ***

Dear S.U.N. 29th January 1983

I've found some words in the FORTH dictionary that arn't in the Glossary (User Notes 4) these are:

XOR	(n1n2---XOR)	Exclusive OR.
U.	(u---)	Print unsigned numnber.
2SWAP	(d1d2---d2d1)	Reverse top two pairs of numbers.
2DROP	(d---)	Discard the top pair of numbers.
H		Dictionary pointer

I'm using "STARTING FORTH" by Leo Brodie to study FORTH. In it he mentions FORTH-79 Standard - how does this relate to fig-FORTH 8080?

*** 2ND LETTER ***

Dear S.U.N. 30th January 1983

Reference User Notes Issue 5, page 7 - the bug in VLIST and ID. I've written a new ID and VLIST and have enclosed a copy of the screen.

The variable FLD is in the FORTH Dictionary but is not used, so I've made use of it. I'M not quite sure how GIV.CNT works. There was some trial and error used in its writing!

I've put the screen into SCR 6 of the EDITOR so that after loading the EDITOR (by typing in 6 LOAD) the ID and VLIST are in the protected Dictionary.

```
SCR 6
  0     ( NEW DEFINITIONS OF ID. & VLIST )
  1     FORTH DEFINITIONS DECIMAL
  2     : ADR-ID  FLD! ;
  3     : GIV.CNT    FLD @ C@ DUP 127 > IF
  4                  128 - DUP 63 > IF
  5                  64 - THEN THEN ;
  6     : ID.     ADR-ID GIV.CNT
  7                0 DO FLD @ 1+ I + C@ DUP
  8                127 > IF 128 - THEN EMIT
  9                LOOP 5 SPACES ;
  10  : NEW.ADRS      FLD @ PFA LFA @ FLD ! ;
  11  : LOOP.LIST      BEGIN ID. NEW.ADRS
  12                   FLD @ DUP ?TERMINAL
  13                   0= * 0= UNTIL ;
  14   : VLIST  CR LATEST LOOP.LIST ;
  15 --      ( OPTIONAL )
```

*** 3RD LETTER ***

Dear S.U.N.

3rd February 1983

After sending you a copy of my VLIST and ID (which I'd been using for some time), I was spurred into improving them. Having had a bit more experience in using FORTH, I was able to produce the definitions as listed.

I said in my last letter that I had put this screen into SCR 6 on the EDITOR tape and includes a non-destructive stack point stack.

I hope that sending you two letters in one week will not confuse you too much!

```
SCR 6
  0     ( VLIST , ID. , DEPTH & .S )
  1     FORTH DEFINITIONS DECIMAL
  2  :  ID.     COUNT 31 AND 0
  3             DO DUP I + C@ 127 AND
  4             EMIT LOOP DROP ;
  5  :  VLIST    CONTEXT @ @ BEGIN
  6              DUP ID. 5 SPACES
  7              PFA LFA @
  8             DUP ?TERMDINAL 0=
  9             * 0= UNTIL DROP ;
  10 : DEPTH     S0 @ SP@ - S / 1 - ;
  11 : .S       CR DEPTH IF
  12            SP@ 2 - S0 @ 2 -
  13            DO I @ . -2 +LOOP
  14            ELSE ." EMPTY " THEN ;
  15 --
```

P. BENNETT
CORNWALL

You are obviously finding it hard to tear yourself away from FORTH - jolly good.

EDITOR

Forth - Entomology

- Don't try to puzzle the Forth interpreter with a word consisting of a graphic symbol.

- ;CODE apparently does not work due to the fact that the word END-CODE does not exist yet.

- CODE exists (though not listed) but seems to need END-CODE as well.

- Caution when using the word R and nothing's on the return-stack.

Please keep up the good work with forth - I'm rather interested in it. If you would like to draw on help from members, let me know which details could/should be tackled.

M. HERMANN Dipl.-Phys.
WEST GERMANY

****

Dear Sharpsoft,

With respect to the bug in the FORTH Editor tape, surely the following sequence is much simpler than that shown in S.U.N. Issue 6.

Load FORTH

Load Editor tape screens 7, 8, 9 load with the error messages telling you that two words already exist in the Dictionary and then screen 9 terminates because of the bug.

Now type Decimal 10 load (CR) Screen 10 loads and you can now use the Editor by typing Editor (CR) use the Editor to amend Screen 9 by the following sequence:-

Type FORTH
Then 9 SCR (or 9 LIST) (CR)
Type EDITOR (CR), L (CR)
Type 12 D (CR) Three times
Type L (CR) to check screen 9. Type flush after winding back the Editor tape and the corrected Screen 9 is sent to tape.

P.S. How about re-writing ID. to tidy up VLIST, surely all it needs is to set BIT 8 of the ASCII character to 0 to keep the ASCII character set to below 128.

How about some proper FORTH programs to show learners like myself how to go about it?

H. WINWOOD
SHEFFIELD

****

Some hints for BASIC SP-5025 etc.

POKE 10029,0 allows CHR$ codes below 32 (except 13).
POKE 10118,0 makes ASC("")=13.
POKE 5679,0 and INPUT or READ will include preceeding spaces in the string, however BASIC will complain about some spaces in the program so POKE 5679,32 again.
POKE 11509,202 allows INP# and POKE 11572,202 allows OUT# of numbers above 240. Useful because INP#254,A returns printer status.
PEEK(57344) returns 249 if shift is being pressed else 248.

S. WALLIS

****

Dear Sirs,

I am using this opportunity to write to you with reference your fig-FORTH tapes. I am very interested in the language and have done as you have suggested in contacting the FORTH Interest Groups UK and USA.

I have had no trouble loading the Editor (which fails due to the bug!) but having tried to input your Bootstrap Editor as carefully as I can this has failed too! I should think you ought to reissue a new Editor on tape. However, you seemed to have helped Mr. Bennett of Cornwall perhaps you can do the same for me?

Having typed the Bootstrap Editor in as suggested. I am unable to save the second Bootstrap Editor to screen 6.. "P" does not work Error 23 comes up, or if all goes well it fails at Flush (CR) with the only response OK! No Play/Record request? In fact I wonder if I have a bad software tape or is it another "bug" epidemic!

Also I notices when I typed O.P. Bootstrap etc. line 1 didn't come up, but "play " request and this appeared sometimes when I tried to type the second Bootstrap Editor lines prefixed with "P". This seems top indicate the first Bootstrap Editor (in memory !?) does not function correctly. Ideas please.

Lastly, as I have the dual disk drives, are you able to give a program to input screens to discs (the glossary mentions disk input/output control) and can one transfer the programs you have supplied (I mean the software) to disc. Your first listing ref "DIP" should be DUP (it helps) - Bootstrap Editor. I look forward to hearing from you and I don't have CP/M.

P.S. Was Mr. D. Willey's "PASCAL Copy" to copy the master tape? if so - it doesn't work unless there is a mistake in the listing!

B. SPOONER
DEVON

****

Dear Sharpsoft,

I write enclosing a short program originally written to provide extra number recognition and counting practise for a class of 4 year olds. The program can easily be expanded, as the language has been kept as simple as possible. On scoring 100 points the child receives a reward of a tune, which can simply be changed in lines 450-470. This display continues until the correct answer is given, and marks are given depending on the amount of unsuccessful attempts. I hope this program proves useful.

```
100 PRINT"◘▣▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
110 FORJ=0TO21:PRINT"▒";TAB(38);"▒":NEXT
120 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒"
130 TEMPO6
140 DATA"BOATS","CATS","HOUSES","DUCKS","BOYS","FISH","MEN"
150 R=INT(RND(2)*9+1)
160 A=INT((RND(45)*7)+1)
170 RESTORE
180 FORE=1TOA
190 READA$
200 NEXTE
210 FORQ=1TOR
220 P=INT(RND(1)*1000+53248)
230 F=(PEEK(P))+(PEEK(P+1))+(PEEK(P+2))+(PEEK(P+3))+(PEEK(P+40)):IFF<>0THEN220
240 F=(PEEK(P+41))+(PEEK(P+42))+(PEEK(P+43)):IFF<>0THEN220
250 ONAGOTO270,290,530,550,570,580,600
260 REM***SHAPE DATA***
270 POKEP+1,58:POKEP+2,123:POKEP+40,66:POKEP+41,67:POKEP+42,67:POKEP+43,86
280 MUSIC"-C1":NEXTQ:GOTO320
290 POKEP,83:POKEP+1,67:POKEP+2,67:POKEP+3,182:POKEP+40,221:POKEP+41,217
300 POKEP+42,221:POKEP+43,217
310 MUSIC"-C1":NEXTQ
320 POKE4466,23:PRINTTAB(3);"HOW MANY ";A$;" ?"
330 GETD:IFD=0THEN330
340 IFD<>RTHEN420
350 POKE4466,23:PRINTTAB(3);"YES!!!!!!!!!!!!!!!!"
360 FORT=1TO1000:NEXT
370 SC=SC+10:PRINT"◘"
380 POKE4466,10:PRINTTAB(5);"YOU HAVE NOW SCORED ";SC
390 PRINTTAB(9);"◘WOW! ! ! ! ! ! "
```

```
400 IFSC>=100THEN450
410 FORT=1TO1000:NEXT:GOTO100
420 POKE4466,23:PRINTTAB(3);"NO,TRY AGAIN    ":SC=SC-1
430 FORT=1TO800:NEXT:GOTO320
440 REM******CONGRATULATIONS******
450 C$= "D5E#FG7DR5G#FGA7E"
460 D$="R5E#FGBAR0A5GR0G5#FE#FD8"
470 MUSICC$;D$
480 PRINT"GO AGAIN ?"
490 GET S$:IF S$=""THEN490
500 IFS$="Y"THENCLR:GOTO100
510 END
520 REM***EXTRA SHAPE DATA***
530 POKEP+1,78:POKEP+2,77:POKEP+41,67:POKEP+42,67:MUSIC""A1"
540 NEXTQ:GOTO320
550 POKEP,87:POKEP+1,191:POKEP+3,108:POKEP+41,67:POKEP+42,67:MUSIC""G1"
560 NEXTQ:GOTO320
570 POKEP+1,207:POKEP+41,67:MUSIC""D1":NEXTQ:GOTO320
580 POKEP,78:POKEP+1,74:POKEP+2,67:POKEP+3,78:POKEP+40,66:POKEP+41,67
590 POKEP+42,67:POKEP+43,66:MUSIC""E1":NEXTQ:GOTO320
600 POKEP,177:POKEP+1,191:POKEP+2,191:POKEP+3,178:POKEP+40,181:POKEP+41,28
610 POKEP+42,29:POKEP+43,182:MUSIC"G1":NEXTQ:GOTO320
620 REM**INSERT EXTRA DATA HERE**
```

B.J. HALLETT
SOMERSET

****

Dear Sirs,

Thank you for sending the Forth discs. As they work off the shelf and are very attractively priced I feel you should mention them more often in conjunction with your tape-based Forth package.

There is one trap I fell into at first. When you flush a screen and your disc happens to be hardware write protected you get no error message, and you'll have to re-enter the screen. It also seems to be a good idea to clear a screen to erase FORMAT's 5EH, prior to an attempt at FLUSHing.

Re some of the member's problems mentioned in Issue 6:

Mr. Thompson (p.40) you can clear the "garbage" headers by FILLing the name field (10F1H-1101H) in the Monitor work area with ODHs first. As the screen numbers are written to tape in binary, no header information will appear on the VDU until the SCR numbers get into the range of printing characters.

Mr. Hayhurst (p.42) you seem to have a case of unread manual. In contrast to standard Pascal our SP-4015 interpreter fortunately doesn't permit the programmer to assign integer values to real variables. In consequence the offending line ought to read either A:FLOAT(K DIV K*K+K-K); or, having declared both A & B integer A:=K DIV K*K+K-K; .

Mr. Howard (p.44) you can persuade Pascal SP-4015 to do an insertion. Enter Z first, then move the cursor into the line no. prompt and modify it to the line no. at which you want to commence insertion, finally move the cursor to the right of the full stop. At this point you may insert until you type CRT on an empty line.

Mr. Shaw (p.48/49) in order to save memory space it is common practice to delimit ASCII strings by setting the last character's MSB, instead of using ODH. this doesn't effect standard ASCII because there bit 7 is not used apart from parity purposes. But Sharp had thjeir own ideas about "ASCII", so we'll have to live with "funny" characters at the end of some strings. One way out would be to define a new VLIST which resets the MSB.

E. RAMM
Fed. Rep. of Germany

The discs mentioned by Mr. Ramm are available from Sharpsoft at fifteen pounds the pair, they will only run on the MZ-80B under CP/M.

EDITOR

****

Dear Sirs,

With reference to Dr. Hayhurst's letter on page 42 Issue 6 regarding his Pascal benchmarking program.

There is a fundamental error in the statement A:=K/K*K+K-K; in the published program, which causes error 17 (incorrect data type).

Variable K has been declared as type integer, therefore K/K which is a REAL expression is wrong. The statement should be A:=K DIV K etc. providing that variable A is declared as type integer also. However as A is declared as type real then the statement should be modified to A:=FLOAT (K DIV K*K+K-K);. This will then allow the integer expression to be evaluated and then converted to real. Data types must not be mixed in this way in Pascal as it defeats one of the fundamental principles of the language.

With reference to W. Howard's letter page 44 in point IV he mentions the machine code routine at 0018H and contrasts this with the routine at 0015H i.e. MSG. Call 0015H prints the string stored from (DE) and obeys the cursor control codes, i.e. if the cursor control code "left arrow" for example is encountered then the cursor is shifted left one position, whereas CALL 0018H causes the cursor control codes to be printed reverse field in the same manner as they are stored in a PRINT statement in BASIC.

D. BERRIDGE
CO. DURHAM

****

Dear Mike,

Thank you for User Notes 6, which I have read with great interest. I am prompted now to write concerning various points I have stumbled over concerning FORTH. Unlike several other subscribers I never quite got far enough to discover the bug in the EDITOR as I gave up too quickly when it seemed that no file was being loaded - I expected a "LOADING xxxx" as produced by BASIC. My first FORTH task is to rewrite an extremely slow BASIC Mortgage program I had written.

1. I wondered how to access my Epson printer. When I discovered Shift/Insert hidden(?) in SUN Issue 4. I tried to reproduce it using EMIT. Using the basic routine:

```
10 GET A$:IF A$="" THEN 10
20 PRINT ASC(A$):GOTO 10
```

produced 97, which is actually a graphics symbol, as produced also by 97 EMIT. How can I toggle on/off the printer within a program? (Re PRINT/P, PRINT/A etc. in SPEED BASIC).

2. There appears to be Peek/Poke protect in FORTH (BASIC has Poke 10167,1 to remove it). Using the simple FORTH word :PEEK QUERY DUP. .; preceded with 100 30000C! I produced PEEK 30000 12339 48, whereas, 30000 . produced 100 correctly. Can you help here? This came up when I tried to work on VLIST. direct access produced correct values of all bytes "peeked" but always failed when used with word definitions.

3. As there is no MUSIC facility, I need to use the BASIC USR command to access the MONITOR routines. Can USR be reproduced in FORTH?

4. Can you confirm that all Peeks/Pokes used in BASIC within MONITOR (up to address 4096) will also work in FORTH?

With reference to the letter on page 52 of SUN Issue 6, my top Asteroids score to date is 229,120 taking 2 hours 10 minutes to complete. A score of 100,000 plus produces an interesting score display (find out for yourselves!) My wife has produced the following BLOCK KUZUSHI scores - Low 1511, Medium 562 and High 167.

B. YOUNG
HERTS

FORTH uses the MZ-80K monitor. Hence all calls or Peeks to the monitor should be O.K. There is no PEEK protect in FORTH. MUSIC words will also have to be written by users.

EDITOR

Dear Editor,

Thank you for User Notes No. 6. Regarding the Notes on how to correct the bug in Screen 9 of the EDITOR tape, there is a much simpler method. No need to generate BOOTSTRAP editor etc. Just 7 LOAD (CR) to load the editor and the tape should stop after Screen 9 (due to the bug). Leave the tape as it is and type 10 LOAD (CR) and screen 10 will be loaded, completing the EDITOR. To call the Editor type EDITOR (CR) and there should be an O.K. Then type 9 list (CR) and Screen 9 will be listed and will become the "current screen", ready for editing.

Follow the instructions from No. 6 onwards as given on page 5 of the Notes Issue 6 and the EDITOR should be O.K. Also, the tape counter numbers given were different from mine. On my tape their were:-

Screen No.		Counter:	
	0		00
	1		06
	2		11
	3		17
	4		23
	5		29
	6		36 etc.

going roughly in steps of 6 counters/screen.

I have entered the String Editor and have found it much better than the prevus Line Editor but it does take getting used to. (I haven't found any use for TOP yet?)

C.M. CHOI
N. HUMBERSIDE

****

Dear Sirs,

Thank you for the 6th USER NOTES. I enjoyed it very much.

Here are two listings which might be interesting and enjoyable. The first program draws graphic figures of random shape on the screen, using the SET-COMMAND.

The second program plays PRELUDE 1 in C from J.S. Bach.

Can somebody perhaps help me with the following problem? I have a SEIKOSHA GP-80D printer. Also I am interested in HIGH RESOLUTION. Can anyone help me with a routine that prints graphic shapes on this printer? I think the routine in the owners manual is insufficient. Keep up the good work.

E. de JONG
THE NETHERLANDS

Because of our limited knowledge we are unable to help with this problem, but if there are any Seikosha owners who have overcome the graphics problem please let other Users know through our letters section.

EDITOR

```
10 DIMM$(255):TEMPO5:M$(1)="_C2EG^C^EG^C^E"
20 M$(2)="_C2FA^D^#EA^D^#E"
30 M$(3)="_G2GB^D^#EG^D^#E"
35 M$(4)="_C2EG^C^EG^C^E"
40 M$(5)="_C2A^C^E^A^C^E^A"
50 M$(6)="_D2D#FA^D#FA^D"
60 M$(7)="_G2GB^D^GB^D^G"
70 M$(8)="_C2CEG^CEG^C"
80 M$(9)="_A2CEG^CEG^C"
90 M$(10)="_D2D#FA^C#FA^C"
100 M$(11)="_G2_BDGBDGB"
110 M$(12)="_G2_#AG#A^#CG#A^#C"
120 M$(13)="_F2_ADA^DDA^D"
130 M$(14)="_F2_#GDFBDFB"
140 M$(15)="_E2CEG^CEG^C"
150 M$(16)="_E2_F_ADF_ADF"
155 M$(17)="_D2_F_ADF_ADF
160 M$(18)="_F2_G_BDF_BDF"
170 M$(19)="_C2_E_GCE_GCE"
180 M$(20)="_C2_G_#ACE_#ACE"
190 M$(21)="_C2_F_ACE_ACE"
200 M$(22)="_D2_F_ACE_ACE"
1000 FORN=1TO2:FORM=1TO22:MUSICM$(M),M$(M):NEXT:NEXT
1010 FORM=1TO20:MUSICM$(M),M$(M):NEXT
1020 MUSIC"_C2_F_A_F2_ACFC_AC_A_F_A3_F_D4_F5_E6_D7_C8"
1030 END
```


```
10 R1=RND(1)*15+1:R2=RND(1)+1.5:R3=INT(RND(1)*10+45)
20 R4=INT(RND(1)*14+8):R5=INT(RND(1)*14+11)
30 DEF FNA(R)=-COS(A1*R)*(EXP(1-R)-1)
40 PRINT"@"
50 FORN=0TO39
60 X=N/39.001
70 FD=47:FU=0
80 Y=SIN(R2-ATN(X/SQR(3-X*X)))
90 FORI=-YTOYSTEP.1
100 R=SQR(X*X+I*I)
110 P=R3+R4*I+R5*FNA(R)
120 IFP>FUGOSUB170:FU=P
130 IFP<FDGOSUB170:FD=P
140 NEXTI,N:USR(62)
150 GETA$:IFA$=""THEN150
160 GOTO10
170 Q=40+N:GOSUB180:Q=40-N:GOSUB180:RETURN
180 IF(P<75)*(P>27)THENSETQ,45-(P-30)
190 RETURN
```

Dear Editor,

I hope that the following routine will be of use to the mathematical readers. May I say that the idea was inspired by a seemingly useless bit of information I read in an earlier "Notes".

You can input a string or a number into the MZ-80K but a function has to be programmed in. The purpose of the routine is that a function can be input as well. The idea is that the function starts as a string variable but the necessary changes are made to the string so that for instance the letters SIN become the reserved word for SINE. Then the DEF FN pointers and the string pointers are altered so that the computer thinks the original string input was a DEF FN. Lines 140-220 allow the input to be in algebraic rather than computer form, that is to say multiplication signs (but not brackets) are implied rather than entered. With the help of the subroutine at 2000, the computer picks up the cases where this has been done, and the multiply signs are inserted.

From 240-430 the computer finds the input string and makes the necessary changes so that it is in DEF FN form. From 450-490 the pointers are changed.

The lines between 500 and 2000 are available for any program you choose to write. I have deliberately made the program as trivial as possible so that you can demonstrate to yourself that it works. If you input say 2x-3 and give x the value 7 it will return 11.

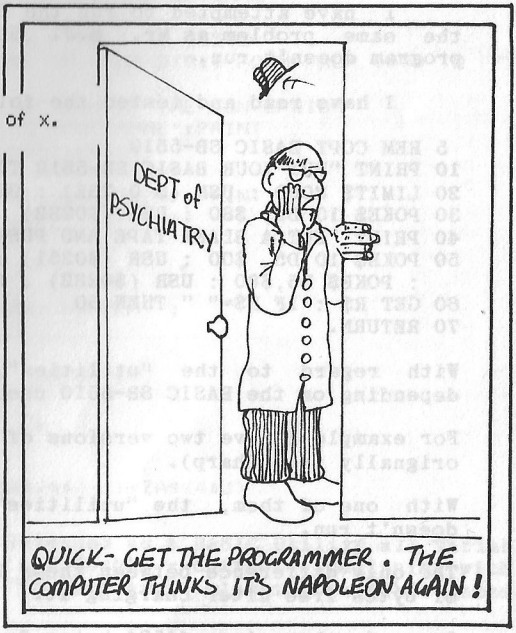
The program will come with the following functions: sin, cos, tan, log, ln and abs. Pi and E can be entered as themselves. All functions need a bracket to enclose the argument, No letter, apart from these should be used except for X the function variable.

While the program itself is trivial it can be used for such varied mathematical topics as graph plotting, Newton's solution for an equation and definite integration. With an extension to cater for two function a program to solve any simple equation be be devised in a few lines, and parametric graphs can be drawn.

G. CHILDS
GLOS.

****

```
100 POKE10167,1
110 CLR
120 PRINT"@Enter your expression in terms of x.
130 INPUTA$
140 L=LEN(A$)
150 N=2
160 Y=ASC(MID$(A$,N,1))
170 Z=ASC(MID$(A$,N-1,1))
180 IF((Y)64)*(Y(89))THENGOSUB2000
190 IFY=40THENGOSUB2000
200 IFY=255THENGOSUB2000
210 N=N+1
220 IFN(LEN(A$)+1THEN160
230 REM***TURN A$ INTO DEFFNA(X)***
240 C=PEEK(17978)+256*PEEK(17979)
250 F=0
260 FORN=CTO50000
270 M=N-F
280 K=PEEK(N)
290 IFK=32THENK=88
300 IFK=43THENK=188
310 IFK=45THENK=189
320 IFK=42THENK=190
330 IFK=47THENK=191
340 IFK=94THENK=207
350 IFK=83THENK=209:F=F+3:N=N+3
360 IFK=67THENK=210:F=F+3:N=N+3
370 IFK=84THENK=211:F=F+3:N=N+3
380 IF(K=65)*(PEEK(N+1)=66)THENK=217:F=F+3:N=N+3
390 IF(K=76)*(PEEK(N+1)=79)THENK=215:F=F+3:N=N+3
400 IF(K=76)*(PEEK(N+1)=78)THENK=216:F=F+2:N=N+2
410 POKEM,K
420 IFK=13THENS=N+1:N=50000
430 NEXT
440 REM***READDRESS START OF DEFFN & $***
450 POKE17972,PEEK(17978)
460 POKE17973,PEEK(17979)
470 S2=INT(S/256)
480 S1=S-256*S2
490 POKE17978,S1:POKE17979,S2
500 E=2.71828
1000 PRINT:INPUT"GIVE YOUR VALUE OF X. ";X
1010 PRINT:PRINT:PRINT"VALUE OF EXPRESSION IS ";FNA(X)
1020 PRINT:PRINT"ANOTHER VALUE? Y/N
1030 GET Z$
1040 IF Z$="N" THEN END
1050 IF Z$="Y" THEN 1000
1060 GOTO 1030
2000 IF (Z=41)+(Z=88)THEN 2005
2001 IF (Z)47)*(Z(58)THEN 2005
2002 RETURN
2005 A$=LEFT$(A$,N-1)+"*"+RIGHT$(A$,L-N+1):L=L+1
2010 RETURN
```



Dear Sirs:

Ref: MZ-80B BASIC Copy Routine & Utilities, Issue No. 5.

I have attempted to run the BASIC Copy Routine and I have the same problem as Mr. M.J. Stanley, that is to say: The program doesn't run.

I have read and tested the following program and it runs OK.

```
 5 REM COPY BASIC SB-5510
10 PRINT "PUT YOUR BASIC SB-5510 TAPE AND PUSH ANY KEY:GOSUB 60
20 LIMIT$ 8000 : USR ($ 0 28E) : USR ($04CE)
30 POKE$ 10 D5, $80 : USR ($02B2)
40 PRINT "PUT A BLANK TAPE AND PUSH ANY KEY : GOSUB 60
50 POKE$ 10 D5, $00 : USR ($0251) : USR ($ 04CE)
   : POKE$ D5,$80 : USR ($0282) : END
60 GET R$ : IF R$=" " THEN 60
70 RETURN.
```

With regard to the "utilities", this program runs or not depending on the BASIC SB-5510 used.

For example I have two versions of BASIC SB-5510 languages (both orignally from Sharp).

With one of them, the "utilities" runs OK, and with the other doesn't run.

The only difference between these BASIC languages is the quantity of bytes free after charging it.

One of them gives 44524 bytes free ("utilities" runs O.K.) and the other gives 44332 bytes free. ("utilities" doesn't run).

These languages don't have any reference.

D. TRIQUELL
SPAIN

****

"DECBINHEX" by Mr. M.A. Hawes
A decimal/hexadecimal conversion program
with a difference - possibly educational.

```
 0000 REM**PROGRAMME TO CONVERT DECIMAL NUMBERS TO BINARY AND HEXADECIMAL
 0010 REM**OR HEXADECIMAL NUMBERS TO BINARY AND DECIMAL
 0020 REM**BY M.A.HAWES  9-11-82
 0030 REM**WILL HANDLE DECIMAL NUMBERS FROM 1 TO 65535
 0040 REM**HEXADECIMAL NUMBERS MUST BE IN FOUR-DIGIT FORM FROM 0000 TO FFFF
 0050 REM**
 0060 REM**A SINGLE 0 CHANGES THE DIRECTION OF THE CONVERSION
 0070 PRINT"{CLR}     DECBINHEX CONVERSION PROGRAMME":PRINT
 0080 PRINT"          DECIMAL 1 - 65535":PRINT
 0090 PRINT"       HEXADECIMAL 0000 - FFFF":PRINT
 0100 PRINT"ENTER A SINGLE 0 TO CHANGE DIRECTION":PRINT
 0110 DIMZA$(16),ZB$(4),ZH$(4),ZD(4),ZS$(4),ZC$(4)
 0120 INPUT"DECIMAL NUMBER ? ";ZD
 0130 IF(ZD>65535)+(ZD<0)THEN60120
 0140 IFZD=0THENPRINT"{CLR}":GOTO60340
 0150 ZD=ZD/65536:PRINT:PRINT"IS BINARY NUMBER ";
 0160 FOR I=1TO16
 0170 ZD=2*(ZD-INT(ZD))
 0180 IFZD>=1THENPRINT"1";:ZA$(I)="1":GOTO60200
 0190 PRINT"0";:ZA$(I)="0"
 0200 IF(I=4)+(I=8)+(I=12)THENPRINT" ";
 0210 NEXT:PRINT:PRINT
 0220 FORJ=1TO4
 0230 ZB$(J)=ZA$(4*J-3)+ZA$(4*J-2)+ZA$(4*J-1)+ZA$(4*J)
 0240 NEXTJ
 0250 FORK=1TO4:RESTORE
 0260 READZF$,ZG$
 0270 IFZF$=ZB$(K)THEN60290
 0280 GOTO60260
 0290 ZH$(K)=ZG$
 0300 NEXTK
 0310 PRINT"OR HEX NUMBER     ";ZH$(1)+ZH$(2)+ZH$(3)+ZH$(4)
 0320 PRINT:PRINT"****************************************"
 0330 PRINT:GOTO60120
 0340 INPUT"HEX NUMBER  XXXX ? ";ZH$
 0350 IFZH$="0"THENPRINT"{CLR}":GOTO60120
 0360 IFLEN(ZH$)<>4THEN60340
 0370 FORL=1TO4:ZS$(L)=MID$(ZH$,L,1):RESTORE
 0380 READZF$,ZG$:IF(ZF$="99")*(ZG$="99")THEN60410
 0390 IFZG$=ZS$(L)THEN60410
 0400 GOTO60380
 0410 ZC$(L)=ZF$
 0420 NEXTL
 0430 IF(ZC$(1)="99")+(ZC$(2)="99")+(ZC$(3)="99")+(ZC$(4)="99")THEN60340
 0440 PRINT:PRINT"IS BINARY NUMBER    ";ZC$(1)+" "+ZC$(2)+" "+ZC$(3)+" "+ZC$(4)
 0450 FORM=1TO4
 0460 ZD(M)=ASC(MID$(ZH$,M,1))
 0470 IFZD(M)<58THENZD(M)=ZD(M)-48:GOTO60490
 0480 ZD(M)=ZD(M)-55
 0490 NEXTM
 0500 ZT=4096*ZD(1)+256*ZD(2)+16*ZD(3)+ZD(4)
 0510 PRINT:PRINT"OR DECIMAL NUMBER ";ZT
 0520 PRINT:PRINT"****************************************"
 0530 PRINT:GOTO60340
 0540 DATA"1111","F","1110","E","1101","D","1100","C","1011","B","1010","A"
 0550 DATA"1001","9","1000","8","0111","7","0110","6","0101","5","0100","4"
 0560 DATA"0011","3","0010","2","0001","1","0000","0","99","99"
```

Intended as a BASIC Utility all variables are double - character beginning with Z to help avoid clashing variable names.

```
10 PRINT:PRINT
20 PRINT"Employing Bairstow's process,this pro"
30 PRINT"gram finds solutions for the equation"
40 PRINT"P(s)=0, where P(s) is a polynomial in s"
50 PRINT"with real coefficients."
60 PRINT"COMMAND=RT,OPTION:NO yields roots only."
70 PRINT"COMMAND=RT,OPTION:YES gives also inter"
80 PRINT"mediate results."
90 PRINT"COMMAND=CH for checks on root accuracy."
100 PRINT"COMMAND=END to finish."
110 PRINT
120 PRINT"# I.Sawczenko.Department of Electronic "
130 PRINT"and Communications Engineering. The Polytechnic of North London.#"
140 PRINT
150 PRINT
160 INPUT "Degree of P(s):N=";N
170 N1=N
180 DIM A(N+2),B(N+2),C(N+2),D(N+2),E(N+2)
190 FOR I=N+2 TO 0 STEP -1
200 A(I)=0:B(I)=0:C(I)=0:D(I)=0:E(I)=0
210 NEXT I
220 FOR I=N TO 0 STEP -1
230 PRINT "Coefficient A(";I;")";
240 INPUT A(I)
250 E(I)=A(I)
260 NEXT I
270 FOR I=N+2 TO 0 STEP -1
280 A(I)=E(I)
290 NEXT I
300 PRINT
310 INPUT "COMMAND=";C$
320 PRINT
330 IF C$="RT"THEN370
340 IF C$="END"THEN1150
350 IF C$="CH"THEN980
360 GOTO310
370 INPUT "OPTION:";D$:PRINT
380 PRINT"     DETERMINATION OF ROOTS:"
390 PRINT:IF D$="YES" THEN410
400 P=0.1:L=0.1:GOTO430
410 PRINT "Quadratic factor(guess):"
420 INPUT"P=";P:INPUT"L=";L
430 IF N<2 THEN920
440 IF D$="YES" THEN460
450 P=0.1:L=0.1
460 FOR J=1 TO 40
470 FOR K=N TO 0 STEP -1
480 C(K)=A(K)+P*C(K+1)+L*C(K+2)
490 NEXT K
500 FOR K=N-2 TO 0 STEP -1
510 D(K)=C(K+2)+P*D(K+1)+L*D(K+2)
520 NEXT K
530 M=L*D(1)+P*D(0)
540 D=D(0)↑2-M*D(1)
550 IF D=0 THEN630
560 DL=(M*C(1)-D(0)*C(0))/D
570 DP=(D(1)*C(0)-D(0)*C(1))/D
```

Program "ROOTS" from Mr I Sawczenko of the Polytechnic of North London.

```
580 L=L+DL
590 P=P+DP
600 IF DL=0 THENIF DP=0 THEN620
610 NEXT J
620 IF D$="NO"THEN710
630 PRINT "Remainder(final):"
640 PRINT "R1=";C(1),"R0=";C(0)-P*C(1)
650 PRINT "Quadratic factor(final):"
660 PRINT "P=";P,"L=";L
670 PRINT "Quotient(final):"
680 FOR K=N TO 0 STEP -1
690 PRINT "C(";K;")=";C(K)
700 NEXT K
710 US=P↑2-4*(-L)
720 RT=SQR(ABS(US))
730 IF US<0 THEN770
740 PRINT"ROOT=";TAB(9);0.5*P+0.5*RT,"  +J0"
750 PRINT"ROOT=";TAB(9);0.5*P-0.5*RT,"  +J0"
760 GOTO790
770 PRINT"ROOT=";TAB(9);0.5*P;"  +J";0.5*RT
780 PRINT"ROOT=";TAB(9);0.5*P;"  -J";0.5*RT
790 FOR I=N+2 TO 0 STEP -1
800 B(I)=C(I)
810 NEXT I
820 FOR I=N+2 TO 0 STEP -1
830 A(I)=0:C(I)=0:D(I)=0
840 NEXT I
850 FOR I=N TO 2 STEP -1
860 A(I-2)=B(I)
870 NEXT I
880 N=N-2
890 PRINT
900 IF D$="NO" THEN430
910 GOTO410
920 IF N=1 THEN940
930 GOTO950
940 PRINT "ROOT=    ";-A(0)/A(1);"  +J0":PRINT
950 PRINT "    ALL ";N1;"  ROOTS DETERMINED ":PRINT
960 N=N1
970 GOTO270
980 N=N1
990 FOR J=N+2 TO 0 STEP -1
1000 A(J)=0:B(J)=0
1010 NEXT J
1020 PRINT"      CHECK ON ROOT Si:":PRINT
1030 INPUT "With      Re(Si)=";R1:INPUT"and      Im(Si)=";R2
1040 PRINT"substituted into the polynomial"
1050 FOR I=N-1 TO 0 STEP -1
1060 B(I)=B(I+1)*R1+A(I+1)*R2
1070 A(I)=A(I+1)*R1-B(I+1)*R2+E(I+1)
1080 NEXT I
1090 R3=A(0)*R1-B(0)*R2+E(0)
1100 R4=B(0)*R1+A(0)*R2
1110 PRINT "yields     Re(P(Si))=";R3
1120 PRINT "and        Im(P(Si))=";R4
1130 PRINT
1140 GOTO270
1150 CLR:END
```

```
Employing Bairstow's process,this pro
gram finds solutions for the equation
P(s)=0, where P(s) is a polynomial in s
with real coefficients.
COMMAND=RT,OPTION:NO yields roots only.
COMMAND=RT,OPTION:YES gives also inter
mediate results.
COMMAND=CH for checks on root accuracy.
COMMAND=END to finish.

# I.Sawczenko.Department of Electronic
and Communications Engineering. The Polytechnic of North London.#


Degree of P(s):N= 3
Coefficient A( 3) 1
Coefficient A( 2) 2
Coefficient A( 1) 2
Coefficient A( 0) 1

COMMAND=RT
OPTION:NO

    DETERMINATION OF ROOTS:

ROOT=     -0.5  +J 0.8660254
ROOT=     -0.5  -J 0.8660254

ROOT=     -1   +J0

    ALL  3  ROOTS DETERMINED
COMMAND=CH
      CHECK ON ROOT Si:

With       Re(Si)=-1
and        Im(Si)= 0
substituted into the polynomial
yields      Re(P(Si))= 0
and         Im(P(Si))= 0


COMMAND=CH
      CHECK ON ROOT Si:

With       Re(Si)=-0.5
and        Im(Si)= 0.8660254
substituted into the polynomial
yields      Re(P(Si))= 0
and         Im(P(Si))= 0

COMMAND=RT
OPTION:NO

    DETERMINATION OF ROOTS:

ROOT=     -0.5  +J 0.8660254
ROOT=     -0.5  -J 0.8660254

ROOT=     -1   +J0

    ALL  3  ROOTS DETERMINED


COMMAND=END
```

$P(s) = s^3 + 2s^2 + 2s + 1$

Output from "ROOTS/P"

```
Employing Bairstow's process,this pro
gram finds solutions for the equation
P(s)=0, where P(s) is a polynomial in s
with real coefficients.
COMMAND=RT,OPTION:NO yields roots only.
COMMAND=RT,OPTION:YES gives also inter
mediate results.
COMMAND=CH for checks on root accuracy.
COMMAND=END to finish.

# I.Sawczenko.Department of Electronic
and Communications Engineering. The Polytechnic of North London.#
Degree of P(s):N= 5
Coefficient A( 5) 1
Coefficient A( 4) 10
Coefficient A( 3) 50
Coefficient A( 2) 150
Coefficient A( 1) 249
Coefficient A( 0) 260

COMMAND=RT
OPTION:NO

    DETERMINATION OF ROOTS:        Output from "ROOTS/P"

ROOT=    -1  +J 2
ROOT=    -1  -J 2

ROOT=    -2  +J 3
ROOT=    -2  -J 3

ROOT=    -4  +J0

    ALL  5  ROOTS DETERMINED
COMMAND=CH
     CHECK ON ROOT Si:

With      Re(Si)=-1
and       Im(Si)= 2
substituted into the polynomial
yields     Re(P(Si))= 0
and        Im(P(Si))= 0


COMMAND=CH
     CHECK ON ROOT Si:

With      Re(Si)=-2
and       Im(Si)=-3
substituted into the polynomial
yields     Re(P(Si))= 0
and        Im(P(Si))= 0


COMMAND=CH
     CHECK ON ROOT Si:

With      Re(Si)=-4
and       Im(Si)= 0
substituted into the polynomial
yields     Re(P(Si))= 0
and        Im(P(Si))= 0

COMMAND=END
```

$$P(s) = s^5 + 10s^4 + 50s^3 + 150s^2 + 249s + 260$$

PARLIAMENT GAME
by
Ken Gaston

```
 90 DIM A$(10)
 100 PRINTCHR$(6):CLR
 110 PRINT"      ********************************"
 120 PRINT"      *                              *"
 130 PRINT"      *           PARLIAMENT         *"
 140 PRINT"      *                              *"
 150 PRINT"      *              GAME            *"
 160 PRINT"      *                              *"
 170 PRINT"      ********************************"
 172 PRINT:PRINT:PRINT:PRINT
 173 H$="+#C3"
 174 F$="-#C3"
 175 FOR D=0 TO 800:NEXT D
 180 PRINT:PRINT:PRINT"      DO YOU KNOW THE RULES?(Y or N)":PRINT"          ----------
----------------------"
 185 GET A$:IF A$=""THEN 185
 190 IF A$<>"Y" THEN1800
 199 REM INITIALISE *****
 200 H=40+INT(RND(1)*21):E=35+INT(RND(1)*31)
 210 M=45+INT(RND(1)*21)
 220 T=0:A=0:R=1
 230 CONSOLE C40
 298 REM
 299 REM MAIN LOOP*****
 300 T=T+1:A=A+1
 310 PRINT:PRINT
 320 PRINTCHR$(6):PRINT TAB(16);"YEAR ";T:PRINTTAB(16);"--------":PRINT
 330 IF T<30+R+INT(RND(1)*(6+R))THEN 370
 340 PRINT"WELL DONE ,OLD MAN..YOUR CONSTITUENCY PARTY 'ENCOURAGE' YOUR RETIREME
NT FROM PARLIAMENT."
 345 MUSIC F$
 350 GOTO 2000
 360 PRINT"     TOTAL MUST ADD UP TO 20":PRINT"     ----------------------"
 365 TEMPO 1:MUSIC"-D":PRINT
 370 INPUT"     ENTER MINISTERIAL DUTIES    ";D:PRINT
 380 INPUT"     ENTER CONSTITUENCY DUTIES   ";C:PRINT
 390 INPUT"     ENTER PARLIAMENTARY DUTIES  ";P:PRINT
 400 IF D+P+C>20 THEN 360
 410 F=20-D-P-C
 420 PRINT"     REMAINDER (FAMILY DUTIES) ";F:MUSIC"RRRRRR":PRINTCHR$(6)
 430 IF(A=5)+(A=4+INT(RND(1)*2))THEN 1500
 440 IF A<>INT(RND(1)*30)THEN 499
 450 PRINT"CRISIS !! EARLY ELECTION FORCED UPON     GOVERNMENT.":PRINT
 460 GOTO1500
 499 REM   CALCULATIONS *****
 500 H=INT(H*((D-5)+(P-5)*2)/50+H-1)
 510 IF H>100 THEN H=100-INT(RND(1)*15)
 520 IF H<0 THEN H=INT(RND(1)*6)
 530 E=INT(E*((C-5)*3+(D-5)+(P-5)+(M-50)/5)/150+E-A+C-1)
 540 IF E>100 THEN E=99-INT(RND(1)*10)
 550 IF E<0 THEN E=INT(RND(1)*3)
 560 M=INT(M*((F-5)*3+(E-40)/10+(H-50)/10)/100+M-A)
 570 IF M>100 THEN M=100-INT(RND(1)*5)
 580 IF M<0 THEN M=INT(RND(1)*15)
```

```
590 GOSUB 1300
599 REM PRESET EVENTS  *****
600 IF D>INT(RND(1)*4) THEN 640
610 PRINT"PARLIAMENT IS CONCERNED ABOUT YOUR MINISTERIAL PERFORMANCE."
620 PRINT"A PUBLIC ENQUIRY IS SET UP TO CHECK YOUR ACTIVITIES.":MUSIC F$:MUSIC"
RRRRRR":PRINTCHR$(6)
630 E=INT(E*14/15):M=INT(M*12/13):H=INT(H*8/9)
640 IF C>INT(RND(1)*5) THEN 720
649 PRINT TAB(15);"YOUR":PRINTTAB(15);"-----"
650 PRINT"LOCAL PARTY ARE UPSET BY YOUR ATTITUDE":PRINT"----------------------------
-------------"
660 PRINT"  THEY STAGE A VOTE OF NO CONFIDENCE.":PRINT"  ----------------------
-------------":PRINT
665 MUSIC F$
670 J=INT(31-E/3-C):IF J<1 THEN J=INT(RND(1)*3)
680 PRINT"VOTES IN YOUR FAVOUR.............";31-J
690 FOR D=0 TO 1400:NEXT D
700 PRINT"VOTES AGAINST YOU.................";J:FOR C=1 TO 2000:NEXT C:PRINT:PRI
NT
702 PRINT:PRINT
705 FOR I=1 TO 1000:NEXT I
710 E=INT(E*32-J/31):M=INT(M*(J+30)/60)
720 IF P>INT(RND(1)*4) THEN 760
730 PRINT"   MAJOR INTERNATIONAL SCANDAL OVER "
740 PRINT" BRITISH GOVERNMENT DISORGANISATION.":MUSIC"RRRRRRRR":TEMPO7:MUSIC F$
750 M=INT(M*10/11):H=INT(H*15/16):E=INT(E*13/14)
760 IF F>INT(RND(1)*4) THEN 800
770 PRINT"FAMILY CRISIS LEAKED TO MEDIA."
780 PRINT"CONSIDERABLE BAD PUBLICITY GENERATED.":MUSIC F$:FOR I=1 TO2000:NEXT I
785 PRINT:PRINT:PRINT
790 M=INT(M*3/5):E=INT(E*12/13):H=H+INT(RND(1)*7)-3
798 REM
799 REM  RANDOM EVENTS *****
800 J=INT(RND(1)*10):IF J=0 THEN 900
810 IF J=1 THEN 950
820 IF(J=2)+(J=5)THEN 1000
830 IF  (J=3)*(R<>9)+(J=3)*(T>30) THEN 1060
840 IF  (J=4)*(R<>9)*(T>6)+(J=4)*(T>30) THEN 1070
850 PRINTCHR$(6):PRINT"    GENERALLY AN UNEVENTFUL YEAR.":TEMPO 7:MUSIC H$:MUSI
C F$
855 PRINT:PRINT:PRINT
860 GOTO 1200
900 PRINT"  FAMINE IN THE FAR EAST.":PRINT
905 MUSIC F$
910 PRINT"  DO YOU SEND AID?(Y or N) "
915 GET A$:IF A$=""THEN 915
920 IF A$ <>"Y" THEN 940
930 FOR W=1 TO 1000:NEXT W
935 E=INT(E*(30-(RND(1)*11))/25):M=INT(M*10/9):H=INT(H*10/11):GOTO 1200
940 E=E*INT((30-(RND(1)*21))/25):GOTO 1200
950 PRINT"  FOREIGN REFUGEE CRISIS.":PRINT
955 MUSIC F$
960 PRINT"  WILL YOU ACCEPT IMMIGRANTS ?(Y or N)
965 GET A$:IF A$="" THEN 965
970 IF A$<>"Y" THEN 990
980 E=E*INT((30-(RND(1)*11))/25):M=INT(M*10/9):GOTO1200
990 E=INT(E*(30-(RND(1)*21))/25):H=INT(H*13/12):GOTO 1200
1000 PRINT"  TIME FOR A PAY INCREASE FOR M.P.'s":PRINT"  ----------------
-----------"
```

```
1005 MUSIC H$
1010 J=INT(RND(1)*5)
1020 PRINT:PRINT"  ELECTORATE SUGGEST.....";J*3+5;"%"
1030 PRINT:PRINT"  SOME M.P.'S WANT.....";J*5+20;"%":PRINT
1040 INPUT"  WHAT DO YOU SUGGEST?(%)..... ";X:E=E+(J*4+6)-X
1045 PRINT:PRINT:PRINT
1050 M=M-(J*4)+X:H=H-(J*4+6)+X:GOTO 1200
1060 PRINTCHR$(6):PRINT"YOUR SUPERIOR IS SUDDENLY TAKEN ILL.":MUSIC F$:GOTO 108
0
1070 PRINTCHR$(6):PRINT"YOUR SUPERIOR DECIDES TO RETIRE.":MUSIC F$
1080 PRINT"WILL YOU TAKE HIS PLACE...?";
1085 FOR X=1 TO 5:FOR Y=1 TO 500
1090 NEXT Y:PRINT"?";:NEXT X
1100 IF H+M*5>INT(RND(1)*70)+40 THEN 1120
1110 PRINT:PRINT"YOU KEEP YOUR PRESENT POSITION.":MUSIC H$:GOTO 1400
1120 PRINT:PRINT"YOU SUCCEED: NEW RANK-";:R=R+1:PRINT:MUSIC H$:GOTO1400
1198 REM
1199 REM  END OF LOOP  *****
1200 FOR U=1 TO 1000:NEXT U:IF H>100 THEN H=100-INT(RND(1)*15)
1210 IF H<0 THEN H=INT(RND(1)*6)
1220 IF E>100 THEN E=99-INT(RND(1)*14)
1230 IF E<0 THEN E=INT(RND(1)*3)
1240 IF M>100 THEN M=99-INT(RND(1)*5)
1250 IF M<0 THEN M=INT(RND(1)*10)+2
1260 GOSUB 1300
1270 GOTO300
1280 REM
1290 REM
1298 REM
1299 REM  REPORT  *****
1300 PRINT:PRINT:PRINT:PRINT"          CURRENT OPINION POLL.":PRINT"
----------"
1310 PRINT"       ELECTORAL SUPPORT    "$E;" %"
1320 PRINT"       SUPPORT OF M.P.'S    "$H;" %"
1330 PRINT"       YOUR MORALE RATING   "$M;" %":PRINT:PRINT:PRINT
1335 FOR Z=1 TO 1000:NEXT Z
1340 IF M>INT(RND(1)*30) THEN 1370
1350 PRINT" YOU ARE FORCED TO RESIGN FOR PERSONAL REASONS.":MUSIC F$
1360 GOTO 2000
1370 FOR K=0 TO 2000:NEXT K
1380 RETURN
1398 REM
1399 REM  RANK  *****
1400 IF R<>1 THEN 1410
1405 PRINT:PRINT:PRINT"      PARLIAMENTARY PRIVATE SECRETARY.":GOTO 1480
1410 IF R<>2 THEN 1420
1415 PRINT:PRINT:PRINT"      PARLIAMENTARY SECRETARY.":GOTO 1480
1420 IF R<>3 THEN 1430
1425 PRINT:PRINT:PRINT"      JUNIOR MINISTER.":GOTO 1480
1430 IF R<>4 THEN 1440
1435 PRINT:PRINT:PRINT"      CABINET MINISTER.":GOTO 1480
1440 IF R<>5 THEN 1450
1445 PRINT:PRINT:PRINT"      SECRETARY OF STATE.":GOTO1480
1450 IF R<>6 THEN 1460
1455 PRINT:PRINT:PRINT"      PRIME MINISTER.":GOTO1480
1460 PRINT:PRINT:PRINT"      LIFE PEER-YOU ARE PROMOTED TO THE HOUSE OF LORDS."
1470 PRINT"YOUR CAREER IS OVER.":GOTO 2000
1480 PRINT:GOTO 1200
1498 REM
```

```
1499 REM  ELECTION  *****
1500 PRINT"     A GENERAL ELECTION IS CALLED.":PRINT"
     ":MUSIC H$:PRINT
1510 PRINT"WILL YOU STAND FOR RE-ELECTION?(Y or N)":PRINT"
     ----------"
1515 GET A$:IF A$=""THEN 1515
1520 IF A$<>"Y" THEN 2000
1530 PRINT:PRINT"RESULTS ARE COMING THROUGH.";
1540 FOR X=1 TO 6:FOR Y=1 TO 300
1550 NEXT Y:PRINT".";:NEXT X
1560 PRINT:PRINT
1570 A=25000-(E*260)-INT(RND(1)*400)-INT(EXP(4))
1575 IF A<1000 THEN A=1000+INT(RND(1)*1000)
1580 PRINT"CONSERVATIVE PARTY.......";A
1590 FOR X=0 TO 1000:NEXT X:PRINT
1600 PRINT"LABOUR PARTY.............";400+INT(RND(1)*200)
1610 FOR X=0 TO 1400:NEXT X:PRINT
1620 PRINT"LIBERAL PARTY............";INT(RND(1)*200)
1630 FOR X=0 TO 1800:NEXT X:PRINT
1640 PRINT"NATIONAL FRONT...........";INT(25000/350)
1650 FOR X=0 TO 2200:NEXT X:PRINT
1660 PRINT"S.D.P. PARTY (YOU).......";25000-A
1670 FOR X=0 TO 2600:NEXT X
1680 IF A>12500 THEN 1700
1690 PRINT"WELL DONE.":MUSIC H$:PRINT:A=0:GOTO 500
1700 PRINT:PRINT"YOU SEEM TO HAVE LOST!":MUSIC F$:PRINT:PRINT
1710 PRINT"DO YOU WANT A RECOUNT?(Y or N)
1715 GET A$:IF A$=""THEN 1715
1720 IF A$<>"Y" THEN 2000
1725 PRINTCHR$(6)
1730 IF Y<>300 THEN 1740
1733 PRINT:PRINT
1735 PRINT"SORRY,NOT ANOTHER!":PRINT"
     ":MUSIC F$:GOTO 2000
1740 Y=300:PRINT:PRINT"O.K. HERE GOES...."
1750 FOR L=0 TO 900:NEXT L
1760 GOTO 1560
1798 REM
1799 REM  GAME RULES *****
1800 PRINTCHR$(6):CONSOLE C80
1810 PRINT"                PARLIAMENT GAME****RULES"
1815 PRINT"                ----------"
1820 PRINT"*****You start the game as a Parliamentary Private Secretary,and aim
to rise to the rank of";
1830 PRINT" Prime Minister by making decisions about how much time you spend on
these activities:"
1835 PRINT"*****You are a member of the S.D.P.Party.":PRINT
1840 PRINT"                 1.PARLIAMENTARY DUTIES.":PRINT
1850 PRINT"                 2.MINISTERIAL RESPONSIBILITIES.":PRINT
1860 PRINT"                 3.CONSTITUENCY DUTIES.":PRINT
1870 PRINT"                 4.FAMILY RESPONSIBILITIES.":PRINT
1880 PRINT"*****You have 20 points to split between these each year.You may be
called upon to make policy decisions as play proceeds.":PRINT:PRINT:PRINT
1900 PRINT"          READY FOR PAGE 2?-PRESS ANY KEY":PRINT"
     --------------------"
1905 GET A$:IF A$=""THEN 1905
1910 PRINTCHR$(6)
```

```
 1920 PRINT"            PARLIAMENT GAME****RULES:SECOND PAGE.":PRINT"
----------------------------------":PRINT:PRINT
 1930 PRINT"***Your decisions will determine three quantities-If any one falls t
oo low...the Game ends.":PRINT
 1940 PRINT"                  1.PERSONAL MORALE.":PRINT
 1950 PRINT"                  2.PARLIAMENTARY SUPPORT.":PRINT
 1960 PRINT"                  3.ELECTORATE SUPPORT.":PRINT:PRINT
 1970 PRINT"**GOOD LUCK!!": PRINT
 1980 PRINT"**READY TO START?(Y or N)":PRINT"--------------------------"
 1985 GET A$:IF A$=""THEN1985
 1990 IF A$="Y"GOTO 200
 1995 GOTO 1800
 1998 REM
 1999 REM END OF GAME *****
 2000 FOR X=0 TO 1599: NEXT X
 2010 PRINTCHR$(6)
 2020 PRINT TAB(15);"GAME OVER.":PRINT TAB(15);"----------"
 2025 PRINT TAB(9);"YOU LASTED";T;" YEARS.":PRINT
 2028 G$="A#D1A#D1A#D1A#D1A#D1A#D1A#D"
 2029 TEMPO 6:MUSIC G$
 2030 PRINT"WOULD YOU LIKE TO PLAY AGAIN ?(Y or N)"
 2035 GET A$:IF A$=""THEN 2035
 2040 IF A$="Y" THEN 100
 2050 PRINT : PRINT "PROGRAM END."
 2060 END
```

Misprint in FORTH

ISSUE 6, Page 3:

: LINE DIP FFFO and 17 ?ERROR SCR <CR>

should have read:

: LINE DUP FFFO and 17 ?ERROR SCR <CR>

Update to WP2

ISSUE 6, Page 64:

line 1180 FOR X = A to I: PRINT/T A$(X): NEXT Y

should be deleted - otherwise any stored file will be truncated at the first comma.

Many thanks J. Tremayne

******

Corrected version of "APPEND" by Mr. M.A. Hawes first published in Issue 5 page 51.

```
1 PRINT"E":POKE10167,1:AP=PEEK(17972)-2:AQ=PEEK(17973)
2 IFAP<OTHENAP=AP+256:AQ=AQ-1
3 FORN=1TO5:READLP,LQ:POKELP,AP:POKELQ,AQ:NEXTN
4 DATA10947,10948,10972,10973,10984,10985,11081,11082,11105,11106
5 PRINT"ON <READY> 'LOAD' FIRST/NEXT PROGRAMME  IN USUAL WAY THEN TYPE 'RUN'
6 PRINT:PRINT"IF NOT APPENDING TYPE 'CONT';THIS WILL  RUN MERGED PROGRAMMES
7 PRINT:PRINT"BREAK IN 7 IS IN ORDER & MAY BE IGNORED":PRINT:STOP
8 AP=6:AQ=72:FORN=1TO5:READLP,LQ:POKELP,AP:POKELQ,AQ:NEXTN
9 DATA10947,10948,10972,10973,10984,10985,11081,11082,11105,11106
```

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