MEMOPAD

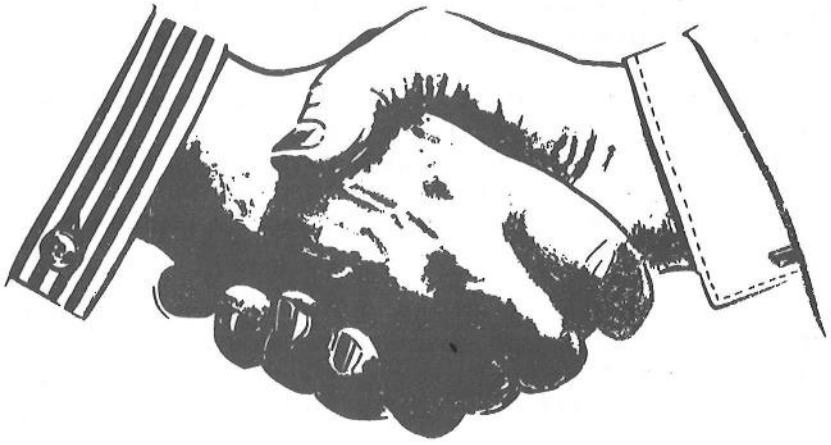
IN THIS ISSUE

EDITORIAL ..... PAGE 427
GENPAT HIT LIST ..... PAGE 428
HIGH SCORES ..... PAGE 428
SOFTWARE REVIEW ..... PAGE 429
USING CP/M - PART 1 ..... PAGE 429
BOMBER ..... PAGE 646
USING CP/M - PART 1 ..... PAGE 431
BOMBER ..... PAGE 434
'NEXTINST' UTILITY ..... PAGE 435
MTX RECOVER ..... PAGE 438
MORSE CODE ..... PAGE 443
MORSE ENCODER ..... PAGE 443
SUBSCRIPTIONS ..... PAGE 444
VIEWPOINT ..... PAGE 445
PASCAL SAVE AND LOAD TO DISC ..... PAGE 449
BLITZ ..... PAGE 451
DISASSEMBLER ..... PAGE 451
SPITFIRE ..... PAGE 454
MACHINE CODE CONVERSIONS ..... PAGE 455
SCREEN DUMP ROUTINE ..... PAGE 461
COLOUR ETCH-A-SKETCH ..... PAGE 464
DANGER ISLAND ..... PAGE 465
THE TOWERS OF HANOI ..... PAGE 470
WALLPAPER FOR THE MIND ..... PAGE 473
CP/M SOFTWARE LIST ..... PAGE 474
HARDWARE LIST ..... PAGE 475
SOFTWARE LIST ..... PAGE 476

ISSUE 3 & 4 SPECIAL

VOL 0011

MEMOPAD



Edited by Tim Marstian

Artwork by Anthony `Joe 90´

Executive Editor
Keith Hook

MEMOPAD IS PUBLISHED BY SYNTAXsoft FOR THE MEMOTECH USER GROUP
UNIT B20, THE NORTHBRIDGE CENTRE, ELM STREET,
BURNLEY, LANCS. BB10 1PD
TELEPHONE: (0282) 38596

AVAILABLE BY SUBSCRIPTION ONLY.

Issue 3 & 4 special

memopad vol 0011

Editorial

The late arrival of this magazine is definitely not our fault! I have just had to re-write the editorial because the magazine has been laying here waiting for the Burnley postmen to come off their wild-cat strike.

We are now trying to get one magazine prepared in advance in order to save these delays. The situation has arisen due to the fact that most of the contributors are now so busy that to catch them on a good day is very difficult.

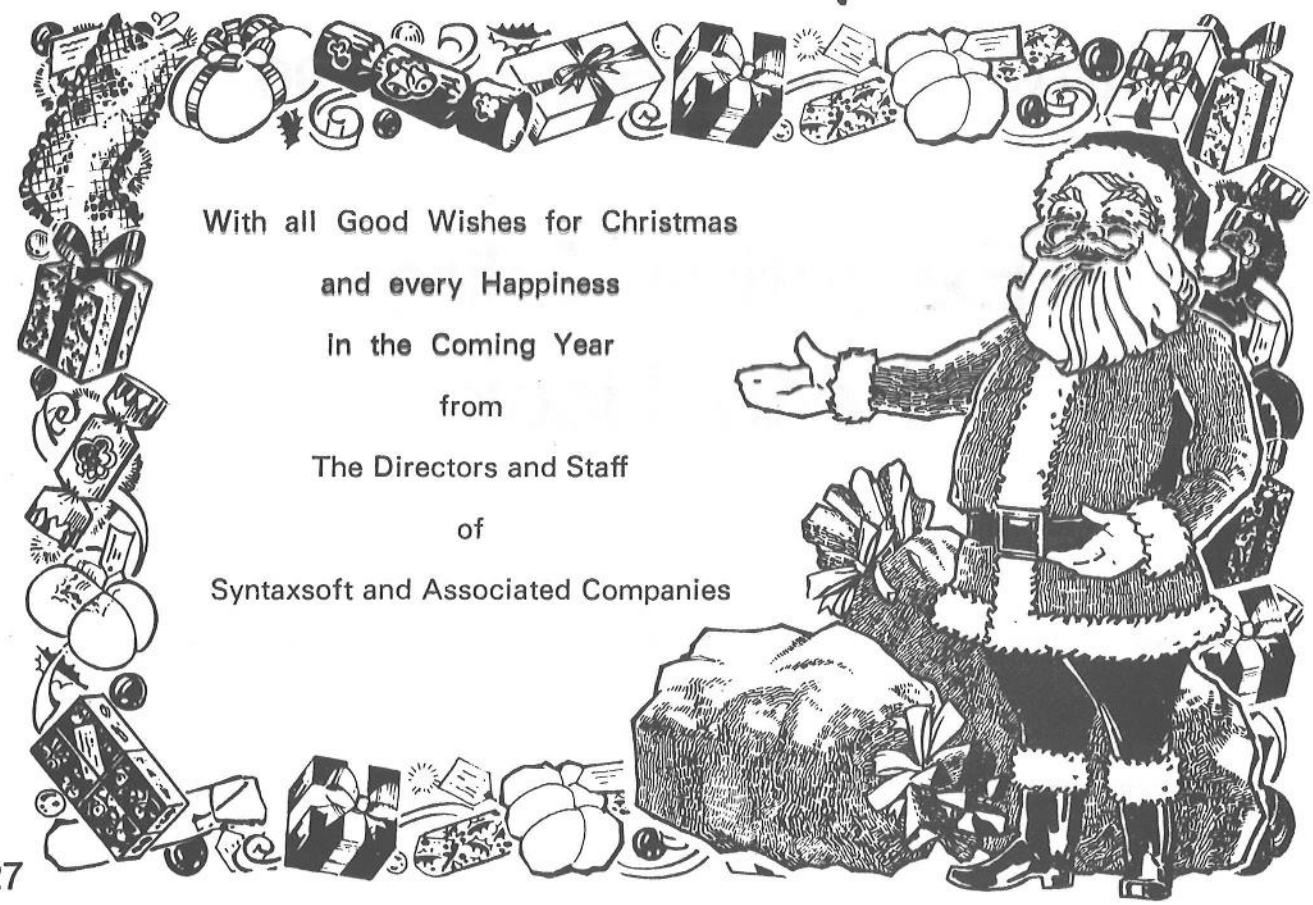
It is amazing how well the Black Beauty has been selling on the run up to Christmas. No advertising, no publicity, yet the machine sells. The new 3.5" drive has surpassed all expectations. Sales have been tremendous. However, be warned, due to the state of the Pound, the Yen and the Dollar there has already been one price increase and if imports continue to fluctuate there will be a need to revise the prices once again.

The CP/M software list is published at the end of the magazine along with games software that has been tested and is known to work on the SDX systems. We shall be adding to this list and it is hoped that Syntaxsoft will have a leaflet with a brief description of each piece of CP/M software available in the New Year.

I do hope that the postal strike is settled in order that the magazine reaches you in time for the festive season.

On a personal note, although I haven't really had anything to do with this magazine due to illness, it is the last edition I will be editing. Who is taking my place is not clear at this moment but I wish him/her every success, and I thank you all for the letters and help I have received over the past fourteen months.

Have a very happy Christmas and a happy New Year. *TM*



GENPAT HIT LIST....

arcade

1.	Tournament Snooker	Magnificent 7
2.	Attack of the Killer Tomatoes	Orion
3.	Highway Encounter	Vortex
4.	Miner Dick	Xaviersine
5.	Cobra	Xaviersine
6.	Sepulcri Scelerati	Megastar
7.	Chess	Continental
8.	Rolla Bearing	Megastar
9.	Downstream Danger	Megastar
10.	Pot Hole Pete	Continental

adventure

1.	Lords of Time	Level 9
2.	Emerald Isle	Level 9
3.	Alice in Wonderland	Continental
4.	Colossal Adventure	Level 9
5.	Murder at the Manor	Sentient

educational

1.	Words and Pictures	Continental
2.	Spellicopter	Sentient
3.	Physics 1	Continental
4.	First Letters	Continental
5.	Maths 1	Continental

High Scores

Can you do better ?

ASTRO-PAC	213,430	Michael Hunt
KNUCKLES	1,147,360	T. Eriksson
CHAMBEROIDS	Completed 4 mins	T. Eriksson
MISSION ALPHATRON	175,340	Matthew Moss
TAPEWORM	175,980	Richard Franks
TOADO	356,414	John Quinn
POT HOLE PETE	106,630	Richard Franks
MAXIMA	1,479,710	S. Olander
STAR COMMAND	140,430	Ian Nichols
PHAID	26,000	Sally Street
OBLOIDS	62,400+	M. Burley
KILOPEDE	82,253	Richard Nash
3D TACHYON FIGHTER	12,500	C. Walker
CONTINENTAL RAIDERS	106,240	Sean Haverty
BLOBBO	148,283	Elizabeth Mahon
QUANTUM	14	Ian Cartwright
QOGO 2	205,000	R. Siddall
MINEFIELD	2,100	C. Walker
FLUMMOX	251,530	C. Walker
TURBO	18,610	Michael Hunt
FATHOMS DEEP	3,450	Matthew Moss
AGROVATOR	675,000+	P. Howard
FIREHOUSE FREDDIE	29,620	T.Eriksson
QXD	43,960	T. Eriksson
ARCADIANS	25,900	Adrian Joynson
MISSILE COMMAND	27,580	Adrian Joynson
LITTLE DEVILS	34,320	Leslie Banks
FELIX IN THE FACTORY	14,740	Peter Crighton
HUNCHY	8,457	John Quinn
SON OF PETE	17,233	T. Eriksson
HAWKWARS	25,800	Gordon Hurd
ESCAPE FROM ZARCOS	76 Items	G Hill + D Stockall
SALTY SAM	40,642	Andrew Johnson
MISSION OMEGA	9,350	R.Barmer
ICEBURG	17,431	Alan Dobson
SNOWBALL	1000	Victor Stepney & Andy Crick
EMERALD ISLE	768/1000	Victor Stepney
SUPERBIKE	23.9Kms	B. Clark
ROLLA BEARING	27,000+	Victor Stepney
DR. FRANKIE	65,435	J Graham
TARGET ZONE	17,470	D J Chamberlain
MINER DICK	22,520	R. Siddall
JUMPING JACK	26,120	Andrew Miller
SURFACE SCANNER	72,060	T. Eriksson
CAVES OF ORB	496/500	Victor Stepney
SEPULCRI SCELERATI	7,925	Andrew Miller GAME COMPLETED
S.M.G.	77,000	Victor Stepney
RETURN TO EDEN	1000	Andy Crick
QUAZZIA	26,660	Andrew Miller
OBLITERATION ZONE	32,670	Alan Dobson
ASTROMILON	142,342	D J Chamberlain
CRYSTAL	32,425	Gordon Hurd GAME COMPLETED
DRIVE THE CEE-5	12,907	Victor Stepney
HIGHWAY ENCOUNTER	123,120	Leslie Banks
KARATE KING	4,570	G Hill
DOWNSTREAM DANGER	8,976	G Hill

Mike Mash has completed Qogo 2 and has quoted the final message - "At last, you have found the Qogan diamonds"

memopad
Issue 3 & 4
special
ual 0011

SOFTWARE REVIEW

Nevada Fortran

MAKER ELLIS COMPUTING

SUPPLIER THE SOFTWARE TOOLSHOP

PRICE #39.95 (inc V.A.T.)

"God, how exciting! It must be as gripping as the Classified Phone Book!" was the comment of a colleague who found me reading the manual. I think that we all tend to react in that way to the oldest of all high-level languages, partly because it is usually very badly taught, and partly because of an evil reputation for programs consisting of reams of unstructured and unintelligible gibberish.

Fortran IV does not deserve this reputation. It is complex, but less so than Assembler or Forth: it does not force you to structure your programs as Pascal does, but it is better-structured than any form of Basic and, of course, unbelievably more powerful and concise, with the ability to produce the most compact object-code you are ever likely to see - a test program which I tried out occupied 1.5 Kbytes of Pascal (excluding the runtime routines) and compiled to 560 bytes of Fortran! It's speed has been exaggerated by it's enthusiasts, and the Nevada compiler is probably not the fastest on the market, but the others do cost ten times as much, and it is faster (and vastly more powerful) than a good compiled Basic (which will also cost you ten times as much). It is perhaps the only important language whose learning is not made more difficult by previous knowledge of Basic.

The Nevada compiler has been around since 1974, this being the third (1984) revision and incorporating many of the post-ANSI 66 extensions. Although it implements a subset of Fortran IV, it is surprising how little is missing. All of these ommisions fall into 3 groups:

Rarely used maths functions such as the type COMPLEX and hyperbolic transcendentals;

Functions which tend to use up too much space or run too slowly on small systems;

Where a facility exists in both a structured and an unstructured form, the Nevada compiler supports only the structured form (e.g. the very confusing Statement Function facility is replaced by the tidier Subprogram Function).

DOUBLE PRECISION can be declared, but is implemented as single-precision. If you download the disk onto your local minicomputer you will however, get true double-precision. Only users of rather specialised maths are likely to notice the difference: the interger dynamic range is - 99999999 to +99999999 and the REAL range from 0.99999999E-127 to 0.1E+127!

The directive EXTERNAL is not implemented, but COPY and CHAIN are, so there is no difficulty in accessing library files or using overlays. Full system programming facilities are provided, including in-line assembly, operating system interfacing, and the ability to LOAD and CALL external assembler programs within a Fortran program, passing variables to and from the assembler routine.

The powerful file-handling facilities which are one of the few excuses for the serious use of Basic are a pale shadow of their Fortran originals which the Nevada compiler fully supports, allowing full interaction with CP/M and sequential or random-access operations from high level down to bit-level on up to 8 simultaneously-open channels.

The compiler is a single-stage job, with automatic assembler linking (you can use the assembler separately, and it has it's own run-time package) and a startlingly long list of options enabling you to redefine almost every startup parameter, run it in an optimising or non-optimising mode and do all the exotic things with memory allocation which one expects from Fortran. he 16K runtime package can either be linked with the .OBJ file to a directly-executable .COM file, or invoked at runtime to save disk-space at the expense of some convenience and speed. Source code is written with Hisoft's ED80 (#10 extra) or the non document mode of Newword or Wordstar.

Fortran remains of huge commercial and scientific importance. It and the Algol/Pascal group are the only general-purpose languages which have ever been universally accepted.

For #39.95 Memotech users can have hands on experience with Fortran up to a surprisingly advanced level, and can write code which can be transported - often directly - to any large systems to which they may have access.

Yours sincerely,
Brian Houghton ★

Using CP/M - Part 1

Theoretically, programs written for one CP/M machine should run without any modification on another - this is not always the case.

It is good practice to persevere with the writing of programs using CP/M rather than being clever and using the routines stored in the computers ROM. Your program is also far easier to port across to another machine.

Persevering with CP/M functions enables you to write suites of programs that are only bounded by your own imagination. There is no reason to go out and buy a "Recover Program", once you have mastered how CP/M stores files in the directory you can write your own utilities and generally they will only take you a couple of hours at the most.

We have covered some aspects of CP/M in other issues, but before we start writing programs we must be fully conversant with the CP/M operating system, so let's just take a little time to reiterate significant parts from past articles.

The Console Command Processor (CCP) decodes the text, typed in from the keyboard, following the normal A" prompt and checks it against any of the built in commands - DIR, REN, ERA etc. If it doesn't recognise one of these commands it goes on to check the disc directory to see if the text corresponds to a file with a COM extention. If a file is matched to the text the CCP then proceeds to load it into memory from address 0100H which is the start of the Transient Program Area (TPA). Once the file has been successfully loaded into memory the CCP executes a CALL 100H which should then run the program.

The CCP also scans the rest of the input and then places it in Default Disc Input Buffer (see fig. 1.). If it finds that you have typed a file name or file names (VDEB TEST.PRO) it places them in the default file control block at 005CH file 1 and 006CH filename 2. From here, it is up to your program to make use of them.

BDOS is the most important section of CP/M as far as we are concerned. It is the work horse of the system and we can make calls to this section in order to perform special tasks.

BDOS can be called by using the vector at 0005H as the section contains the code JP BDOS. Before calling BDOS the C register must be loaded with a function number which tells BDOS which task has to be carried out - extreme care must be exercised when using the C register because by loading it with the wrong function number you could find you have just eliminated one of your disk files.

Parameters can also be passed to BDOS by using the DE registers. Single byte parameters are passed using the E register and 16 bit parameters in the register pair. It should be stated here that users who purchased there FDX systems in the early days, and have the big loose leaf folder should not use the function table printed as it is totally wrong! Sometimes BDOS passes parameters back and these are normally stored in the accumalator on return from the BDOS call.

LISTING ONE

```
        ORG 100H
FUNC:   EQU 09          ; CODE NUMBER
                        ; FOR WRITING A STRING
        LD C, FUNC      ; SEND FUNC IN C
        LD DE, MESS     ; DE = STRING ADDRESS
        CALL 0005       ; CALL BDOS
        JP 0000         ; NOW COMPLETED
                        ; DO A WARM BOOT
MESS:   DEFM 'HELLO! THIS IS A TEST'
```

If you take a look at listing one you will see a simple example of how to print a message to the screen using a BDOS function call. The $ sign is very important when printing strings to the screen. In fact, omit it, and you can see the results that occur. CP/M uses the $ to terminate a string output.

Once you have assembled and run the program you will notice that there is a coniderable delay between printing the message and jumping to the prompt sign. This is caused by the JP to location 0000 which then re-loads the system tracks. There is a way around this which will totally avoid this delay. Remember that we have already stated that the CCP executes any program by doing a call to 100H. If we ensure that the stack is preserved before the program is run we can then use a RET instruction to retain to the CP/M prompt.

```
LG              ORG 100H
FUNC:           EQU 09
LD              (STACK), SP
LD              SP, OURSTK
```

```
.....................................

        REST OF CODE

LD              SP, (STACK)

                RET

OUR STACK       DEFW 0000
STACK           DEFW 0000
```

Because the stack always builds down in memory, it is important to utilise our own stack when writing CP/M programs. If one of our programs makes extensive use of the stack and we don't utilise our own, if we then continue to use the CP/M stack and our program makes extensive use of it, we could be in the dangerous position of over writing the Default Disc Input Buffer which would make it impossible for our program to perform disc I/O.

Next month we will take a look at the FCB (File Control Block) and how we can use the information to restore erased files and search the disc directory.

FNC	HEX	FUNCTION NAME	INPUTS	OUTPUTS
0	00	System Reset	none	none
1	01	Console Input	none	A=char
2	02	Console Output	E=char	none
3	03	Reader Input	none	A=char
4	04	Punch Output	E=char	none
5	05	List Output	E=char	none
6	06	Direct Console I/O	see def	see def
7	07	Get I/O Byte	none	A=(0003)
8	08	Set I/O Byte	E=iobyte	none
9	09	Print string$	DE=.string	none
10	0A	Read Console Buffer	DE=.buffer	string
11	0B	Get Console Status	none	A=status
12	0C	Get Version Number	none	HL=version
13	0D	Reset Disk System	none	see def
14	0E	Select Disk	E=disk	see def
15	0F	Open File	DE=.fcb	A=dir code

16	10	Close File	DE=.fcb	A=dir code
17	11	Search for first occurence	DE=.fcb	A=dir code
18	12	Search for next occurence	none	A=dir code
19	13	Delete File	DE=.fcb	A=dir code
20	14	Read Sequential	DE=.fcb	A=error code
21	15	Write Sequential	DE=.fcb	A=error code
22	16	Create New File	DE=.fcb	A=dir code
23	17	Rename File	DE=.fcb	A=dir code
24	18	Return Login Vector	none	HL=login vector
25	19	Return Current Disk	none	A=disk number
26	1A	Set DMA Address	DE=dma	none
27	1B	Get RBR Address	none	HL=.RBR
28	1C	Write Protect Disk	none	see def
29	1D	Get R/O Vector	none	HL=R/O vector
30	1E	Set File Attributes	DE=.fcb	see def
31	1F	Get DPB Address	none	HL=.DPB
32	20	Set or Get USER Code	see def	see def
33	21	Read Random	DE=.fcb	A=error code
34	22	Write Random	DE=.fcb	A=error code
35	23	Compute File Size	DE=.fcb	r0, r1, ov set
36	24	Set Random Record	DE=.fcb	r0, r1, ov set
37	25	Reset Drive	DE=drive vector	A=00
38	26	No operation		
39	27	No operation		
40	28	Write Random with Zero Fill	DE=.fcb	A=error code

Bomber

```
1 GOTO 22
2 CLEAR
3 USER  SAVE "BOMBER"
4 GENPAT 1,143,62,62,28,62,62,62,28,8
5 GENPAT 1,144,24,24,102,102,219,219,219,255
6 GENPAT 1,145,24,24,24,24,60,126,231,231
7 GENPAT 1,146,24,60,126,219,153,153,255,255
8 GENPAT 1,147,255,153,255,153,255,153,255,255
9 GENPAT 1,148,231,231,231,255,255,231,231,231
10 GENPAT 1,149,255,219,165,219,165,219,165,255
11 GENPAT 1,150,255,129,129,129,129,129,129,255
12 GENPAT 1,151,0,112,248,248,255,255,127,16
13 GENPAT 1,152,7,1,50,116,255,255,255,6
14 GENPAT 1,153,248,32,64,128,248,248,240,0
15 GENPAT 2,147,17,26,26,26,26,26,26,26
16 GENPAT 2,148,31,26,24,26,26,31,26,24
17 GENPAT 2,149,24,26,24,26,24,26,24,26
18 GENPAT 2,150,16,18,11,20,22,24,26,16
19 GENPAT 2,151,5,149,245,101,53,53,53,21
20 GENPAT 2,152,53,53,53,53,53,53,53,5
21 GENPAT 2,153,53,53,53,53,53,53,49,53
22 VS 4: COLOUR 4,1: CLS : FOR X=0 TO 31: CSR X,22: PRINT CHR$(150);: NEXT X
23 FOR X=0 TO 31: LET HE=14+INT(RND*5): LET CH=INT(RND*3+147)
24 COLOUR 1,1: CSR X,HE: PRINT CHR$(CH-3)
25 FOR Z=HE+1 TO 21: CSR X,Z: PRINT CHR$(CH);: NEXT Z: NEXT X
26 CSR 0,23: FOR X=0 TO 7: COLOUR 3,8-X: LINE 0,X,255,X: NEXT X
27 LET B=100: LET PY=1: LET PX=0: LET P$=" "+CHR$(151)+CHR$(152)+CHR$(153): LET BY=0: LET BX=0: LET BF=0: LET B$=CHR$(143)
28 CSR 0,0: PRINT "Bombs Left";B
29 CSR PX,PY: PRINT P$;
30 IF ASC(SPK$)>143 THEN  GOTO 40
31 LET PX=PX+1: IF PX>31 THEN  LET PX=0: LET PY=PY+1
32 IF PX=25 AND PY=22 THEN  GOTO 38
33 IF BF=1 THEN  GOSUB 36 ELSE  PAUSE 37
34 IF INKEY$="" THEN  GOTO 29
35 IF BF=1 OR B=0 THEN  GOTO 29 ELSE  LET BF=1: LET B=B-1: LET BX=PX+1+32*(PX=31): LET BY=PY+1: CSR BX,BY: PRINT B$: CSR 10,0: PRINT
 B;"  ": GOTO 29
36 CSR BX,BY: PRINT " ": CSR BX,BY+1: IF SPK$<>" " THEN  FOR Z=BY TO 22: CSR BX,Z: PRINT " ": NEXT Z: LET BF=0: RETURN
37 LET BY=BY+1: IF BY=22 THEN  LET BF=0: RETURN  ELSE  CSR BX,BY: PRINT B$: RETURN
38 FOR X=2 TO 13: COLOUR 4,X: CRVS 5,1,X-2,X-2,36-2*X,28-2*X,32: COLOUR 2,X: CLS : NEXT X
39 VS 4: CSR 5,10: PRINT "SURVIVED:YOUR SCORE=";B*B: GOTO 41
40 FOR X=0 TO 15: PAUSE 15: COLOUR 4,X: NEXT X: COLOUR 4,6: CSR 10,10: PRINT "YOU'RE DEAD"
41 CSR 3,21: PRINT "PRESS ANY KEY TO PLAY AGAIN": PAUSE 100
42 IF INKEY$<>"" THEN  GOTO 22 ELSE  GOTO 42
```

'Nextinst' Utility by Eric Roy

The following utility for the Memotech provides an extra command for the Panel functions. It is called while in the Panel by pressing the key 'N'.

It's purpose is to advance the display cursor to point to the next valid instruction, correctly decoding and skipping the data bytes that follow the RST 10 and RST 28 instructions, so that the display cursor is always pointing to the next vaild instruction. This should save you time when single stepping as trying to decode these data bytes by hand is a chore in itself.

In addition the instruction pointed to by the display cursor is shown in mnumonic form above the display window. e.g. >0000: DI So all you need do is set PC register to the address shown to continue single stepping.

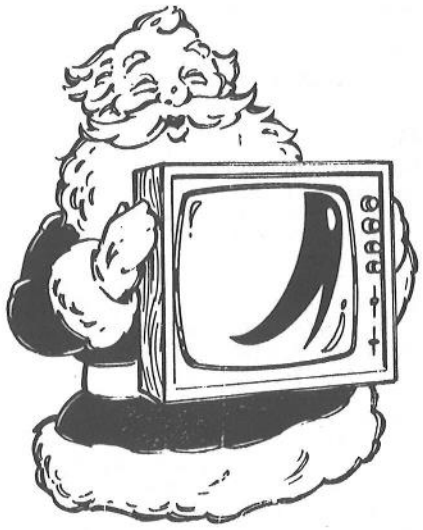
After entering the listing and saving it, run it, then type PANEL <ret>. Once in the panel set up the display pointer to the address you want to start from with the 'D' Display function, then exit from Display by pressing . (dot) or BRK. Pressing 'N' will now advance the cursor to the next instruction.

One final point that may be of interest to all Z80 programmers is the FINST subroutine, which returns the length of the instruction pointed to by DE in register A. e.g. LD DE,ADDR : CALL FINST on return A contains the length of the instruction starting from ADDR. ★

```
10 REM ********************************
20 REM ****   PANEL 'N' COMMAND    ****
30 REM ****  Memotech MTX micros   ****
40 REM ****                        ****
50 REM ****  By Eric Roy  Nov.86   ****
60 REM ********************************
70 REM
100 CODE
```

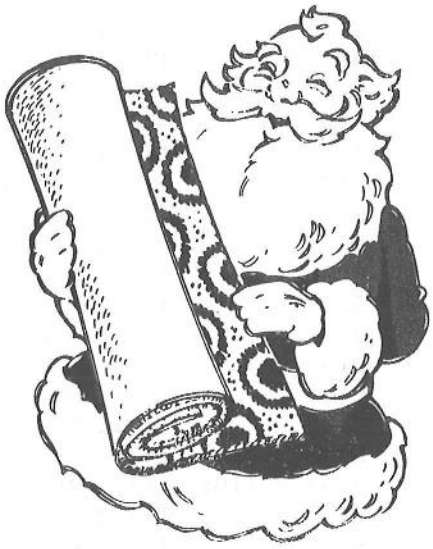
```
811B          DB £60      ; Select VS 0
811C          DB £23      ; CSR
811D          DB £24      ; 4
811E          DB £30      ; 16
811F          DB £25      ; Erase to end of line
8120          DB £81      ; Print >
8121          DB ">"
```

```
80F1 VPANEL:  LD HL,NCOM ; Vector panel command
80F4          LD (£FA9F),HL
80F7          LD A,£C3
80F9          LD (£FA9E),A
80FC          RET
80FD NCOM:    CP "N"      ; Key N ?
80FF          JR NZ,NC1   ; No
8101          LD DE,(£FDA1)       ; DE -> instruction
8105          PUSH DE
8106          CALL FINST ; Get instr. length
8109          POP HL      ; HL -> instruction
810A          LD E,A
810B          LD D,00     ; DE = length
810D          LD A,(HL)   ; Get instruction
810E          CP £D7      ; Test for RST 10 & 28
8110          JR Z,RST10
8112          CP £EF
8114          JR Z,RST28
8116          ADD HL,DE
8117          LD (£FDA1),HL       ; -> next instr.
811A NC1:     RST 10
```



```
8122          LD BC,(£FDA1)       ; BC -> instruction
8126          PUSH BC
8127          CALL £1B50 ; Print addr. in hex
812A          RST 10
812B          DB £81
812C          DB £3A
812D          POP BC
812E          XOR A       ; Set system vars to 0
812F          LD (£FDBB),A        ; before printing instr.
8132          LD (£FDB0),A        ; These are used to list
8135          LD (£FD87),A        ; labels and comments.
8138          LD HL,NC2   ; HL = Return addr.
813B          PUSH HL     ; Save on stack
813C          PUSH BC     ; BC -> instruction
813D          CALL £2BBA
8140          CALL £29B6
8143          LD B,01     ; Number of instr to print
8145          JP £2457    ; Print them
8148 NC2:     RET
8149 RST10:   CALL INCPTR         ; Get byte after RST
814C          BIT 5,A     ; More data ?
814E          PUSH AF
814F          BIT 7,A     ; Number of data bytes ?
8151          CALL NZ,R10B        ; Yes
8154          POP AF      ; More data bytes
8155          JR NZ,RST10         ; Yes - get next
8157 R10A:    CALL INCPTR         ; Adjust pointer
815A          JR NC1      ; Finished
815C R10B:    BIT 6,A     ; Output BC ?
815E          RET NZ      ; Yes
815F          AND £1F
8161          LD B,A      ; B = num. of data bytes
8162 R10C:    CALL INCPTR
8165          DJNZ R10C
8167          RET
8168 RST28:   CALL INCPTR         ; Get byte after RST 28
816B          BIT 7,A     ; Jump table address ?
816D          JR Z,R10A
816F          BIT 6,A     ; More data bytes
8171          PUSH AF
8172          CP £AB      ; Test for MOREHEX
8174          JR NZ,R28B ; Try SYNPROC
8176          CALL R28D
8179 R28A:    POP AF      ; More data bytes ?
817A          JP NZ,RST28         ; Yes
817D          JR R10A
817F R28B:    CP £A1
8181          JR NZ,R28A ; Not SYNPROC
8183 R28C:    CALL INCPTR         ; Get byte
8186          CP £C9      ; C9 is end of data
8188          JR NZ,R28C ; No
818A          POP AF
818B          JR R10A
818D R28D:    CALL INCPTR
8190          BIT 7,A     ; Last letter ?
8192          JR Z,R28D   ; No - get next byte
8194          RET
8195 INCPTR:  LD HL,(£FDA1)
```

```
8198            INC HL
8199            LD (£FDA1),HL       ; Inc pointer
819C            LD A,(HL)  ; Get byte (for RST routines)
819D            RET
819E FINST:     LD HL,LTBL0         ; 2 byte instr. table
81A1            LD BC,25
81A4            LD A,(DE)  ; Test byte in table
81A5            CPIR       ; Match ?
81A7            JR Z,F2    ; Yes
81A9            CP £ED     ; Instr. starting with ED
81AB            JR Z,F1A   ; can be 2, or 4 bytes long
81AD            CP £DD     ; Instr. starting with DD or FD
81AF            JR Z,F3A   ; can be 2, 3 or 4 bytes !ong
81B1            CP £FD
81B3            JR Z,F3A
81B5            LD HL,LTBL3         ; 3 byte instr. table
81B8            LD BC,26
81BB            CPIR       ; Match ?
81BD            JR Z,F3    ; Yes
81BF F1:        LD A,01    ; If no match then instr.
81C1            RET        ; must be 1 byte long.
81C2 F1A:       LD HL,LTBL1         ; ED 4 byte instr. table
81C5            LD BC,06   ; 6 of them
81C8            INC DE
81C9            LD A,(DE)  ; A = second byte of instr.
81CA            CPIR       ; Match ?
81CC            JR Z,F4    ; Yes
81CE F2:        LD A,02    ; If no match ED instr.
81D0            RET        ; must be 2 bytes long.
81D1 F3:        LD A,03
81D3            RET
81D4 F3A:       LD HL,LTBL4         ; DD & FD 4 byte instr. table
81D7            LD BC,05
81DA            INC DE
81DB            LD A,(DE)  ; A = second byte
81DC            CPIR       ; Match ?
81DE            JR Z,F4    ; Yes
81E0            LD HL,LTBL2         ; DD & FD 2 byte instr. table
81E3            LD BC,11
81E6            CPIR       ; Match ?
81E8            JR Z,F2    ; Yes
81EA            JR F3      ; Must be 3 bytes long
81EC F4:        LD A,04
81EE            RET
81EF LTBL0:     DB £06,£0E,£10,£16,£18,£1E,£20
81F6            DB £26,£28,£2E,£30,£36,£38
81FC            DB £3E,£C6,£CB,£CE,£D3,£D6
8202            DB £DB,£DE,£E6,£EE,£F6,£FE
8208 LTBL1:     DB £43,£4B,£53,£5B,£73,£7B
820E LTBL2:     DB £09,£19,£23,£29,£2B,£39,£E1
8215            DB £E3,£E5,£E9,£F9
8219 LTBL3:     DB £01,£11,£21,£22,£2A,£31,£32
8220            DB £3A,£C2,£C3,£C4,£CA
8225            DB £CC,£CD,£D2,£D4,£DA
822A            DB £DC,£E2,£E4,£EA,£EC
822F            DB £F2,£F4,£FA,£FC
8233 LTBL4:     DB £21,£22,£2A,£36,£CB
8238            RET
```



Symbols:

NCOM	80FD	VPANEL	80F1
FINST	819E	RST10	8149
RST28	8168	NC1	811A
NC2	8148	INCPTR	8195
R10B	815C	R10A	8157
R10C	8162	R28B	817F
R28D	818D	R28A	8179
R28C	8183	LTBL0	81EF
F2	81CE	F1A	81C2
F3A	81D4	LTBL3	8219
F3	81D1	F1	81BF
LTBL1	8208	F4	81EC
LTBL4	8233	LTBL2	820E

memopad vol 0011 Issue 3 & 4 special

MTX Recover by A. F Wilson

From a previous article, MTX TOKENS, (issue 9, vol.0010, Memopad) where I discussed how the MTX series stores basic as keywords and how to get the best advantage in memory usage and speed of execution, I would like to follow on and discuss how some of the concepts of the earlier article can be used to great effect in understanding how the MTX creates auto-run code, how to relocate code when loading to run on either the MTX 500 or 512, and more importantly how to RECOVER reset programmes: i.e. how to disable the copy protection that certain games programs employed so that we can convert the cassette game to run on DISC.

A summary of the other article is given. This example shows the way Memotech MTX basic is stored in memory, i.e. it uses keywords like the Spectrum and Commodore 64. A simple example of saving a program is given.

```
10 SAVE "RECOVER"
```

This is represented as 15 bytes in memory.

```
#0F #00                              =line length (2 byte value)
#0A #00                              =line number (2 byte value), i.e.10
#B5                                  =Keyword token for SAVE
#22                                  =hex code for the quote sign, '
#52 #45 #43 #4F #56 #45 #52          =hex values for RECOVER
#22                                  =quote again
#FF                                  =end of basic line marker.
```

The object of this article is to explain how to copy protected auto-run programs, and how to relocate a program to run on the MTX 500 or 512. In a later article I will explain how to use this knowledge to de-protect some cassette games so that we can transfer them to disc. This is where the article is derived from - we are recovering protected data.

I will now discuss the four main copy protection criteria: Break key disable, make listing invisible, auto-run and deciding which model the program is to be run on.

1.Disable the Break Key.

This can eaily be done from basic:

```
POKE 64862,13
```

Also just as easy in assembly language:

```
LD A,13
LD (INTFFF),A
```

INTFFF is a system variable in the Memotech operating system, it is at 64862 or #FD5E. EVERY 125th OF A SECOND there is an interrupt; the basic directs itself to the above address to see if it has to perform any tasks before returning to the basic interpreter. INTFFF is a one byte memory location. It is broken down to the following:

```
Bit   0: Routine for continuous sound called
      1: Break key disable test
      2: Keyboard auto-repeat enabled
      3: Cursor flash and sprite movement enabled
      4: USER bit, a user defined jump location for code
      5: As for 4
      6: As for 4
      7: The clock bit is toggled every 125th of a second.
```

The only bit of interest to us at the moment is bit 1; for a fuller explanation of the others refer to Genpat's, the official Memotech Users Group, datasheets. On the MTX, break disabled is done by setting bit 1 to zero with the other 3 lower bits set to 1, this is where the 13 comes from:

```
bit    : 0 1 2 3 4 5 6 7
value  : 1 0 1 1 0 0 0 0
```

The best way to use the disable technique is to include it in the assembly code listing at the beginning of the program code.

2. Invisible Basic

The best way to describe this is to show the short assembly code listing and to describe it in detail afterwards.

Listing 1:-

```
1 CODE
      LD HL,#FAA4    ;point HL to the top of NODDY
      LD DE,#FAA5    ;
      LD BC,#000C    ;11 bytes to be moved
      XOR A          ;clear A and the flags
      LD (HL),A      ;store zero at location pointed at by HL
      LDIR           ;zero 12 bytes from #FAA4 to #FAAF
      LD HL,#FA9E    ;point HL to FEXPAND, panel extention
      LD (HL),#C7    ;
      RET
```

MTX basic needs to keep track of programs in memory. It does this by assigning specific memory locations which are in the system variables. For example, take the above listing. Although it is in assembly language, it has been assigned a basic line number and special keyword code. If the user goes into the front panel and presses D (for display)and enters FAA4 <RET>, followed by the <BRK> key should see a hex dump at the bottom of the screen. The information we are interested in are the 12 bytes from FAA4. These represent memory pointers for the basic interpreter, as without this the program wouldn't exist. This is what we are looking at:

Memory Location	Code in hex	Description
FAA4 & FAA5	1B 40	Top of Noddy text language
FAA6	00	Page number of Noddy
FAA7 & FAA8	1B 40	Top of Basic
FAA9	00	Page number of Basic
FAAA & FAAB	00 40	Virtual addresses are calculated from this value, note it's 00 80 on a 500
FAAC to FAAF	1B 40 00 00	Top of Basic page information.

By changing some of the values in these addresses can result in a number of odd effects. One of these effects is to make the Basic listing invisible, and also erase some important information about how long a program is, its position in memory, i.e. information a hacker would need. The first 6 lines of listing 1 loads zeroes into these 12 memory locations, the basic interpreter thinks it's not switched on and crashes. The next two lines stop the computer from seizing up by performing a #C7. When the computer crashes it FEXPAND, to see if a panel jump routine is to be called first. Here we store #C7, the code for RST #00, this performs a warm start up. This ensures the program stays intact and avoids a crash. Ingenius, eh!

Note listing 1 is 27 (#1B) bytes long, as the first 8 bytes are due to the keyword token and line length etc., see the saving example earlier plus 1 byte for #FF, the end marker.

3.Auto-run

This is very easy to perform:

```
10  SAVE "RECOVER"
100 the program
```

When the program is run it goes to line 10, performs the save, and then runs the program as its next instuction in the listing. So when we go to load this back into the computer, the basic interpreter is already pointing to line 100 and automatically runs.

4.500 or 512

The contents of location #FA7A can be used to simulate a 500 on a 512. On a 500 this will, when peeked from basic or viewed from panel, equal zero or be equal to one on a 512. Therefore, if we change this value to zero on a 512, then the 512 believes it`s a 500. This is very useful when we want to create code which will run on both machines. An MTX 512`s ram starts at #4000 and at #8000 on a 500, but note both machines have the same addressing point, #8000. Therefore, if our program is <32k, we can load it into memory starting at #8000. Once it`s started to load, the program jumps to a piece of code (see listing 2) to check location #FA7A to see if it`s a 500 or a 512. If it's a 512, the code is moved to #4000, else it's left to load at #8000.

Listing 2:-

```
100 CODE
        DI                  ;disable basic interrupts
        LD SP,(#FA96)       ;save the start of the machine stack in the stack
                             pointer register
        LD HL, #FD4F        ;this is the USRRST location which is called
                             every time there is an interrupt. If the
location is occupied, the code at the location
                             pointed to by USRRST is automatically executed.
        LD DE,START         ;register DE=the start of the program code
        LD (HL),E           ;
        INC HL              ;
        LD (HL),D           ;load start address at USRRST
        LD A,(#FA7A)        ;is it a 500 or a 512, let A=0 or 1
        OR A                ;clear flags leave A register untouched
        JP Z,START          ;if A=0 then it's a 500, continue to load the
                             program at #8000 onwards
        LD DE,END           ;else move the code down to #4000 using a
        LD HL,END-#4000     ;block move command
        LD BC,END-START+1   ;
        LDDR                ;
        JP START            ;now run the new moved program code
```

START : start of program code

NB: END= the last address of the code to be moved in hexadecimal. Also, start address is in hex.

Summary

Therefore to copy protect a program, just follow this simple procedure;-

```
1   CODE                    ;i.e. the Basic invisible coding
10  SAVE "program name"     ;
100 CODE                    ;
        coding for MTX 500 or 512
then    coding for the actual program
```

Now by typing RUN <RET>, the program will hide itself and save itself, and when you load it back it will auto-run.

Now that we know how to protect a game, let's see how we can use this knowledge to deprotect games which use this technique. I have chosen to demonstrate this technique on TOADO, which everyone who owns an MTX should have. Note: Other games which use this protection method are, Kilopede, Chess, Draughts; but also note some of these incorporate other protection schemes, i.e. Chess, but the general rules that we will discuss in this article will give you clues to unravelling the other methods games companies use. The primary reason for deprotecting games is not to make hundreds of 'bootleg' copies, but to get cassette games deprotected so that they can be adapted to run on disc.

First, type ROM n, depending on the type of disc system you have, then insert a blank formatted disc into the disc drive. Now load TOADO from tape in the usual way. Once TOADO has loaded, press the reset keys. This should erase everything. Now go into PANEL and type display 4000, press<.>, so that you don't change anything. This gives the following Panel dump:

Panel 1:

```
4000 : 08 07 06 05 04 03 02 01
4008 : A4 FA 11 A5 FA 01 0C 00
4010 : AF 77 ED 80 21 9F FA 36
4018 : C7 C9 FF 0D 00 0A 00 B5
4020 : 22 54 4F 41 44 4F 22 FF
4028 : 10 20 64 00 C2 08 20 F3
4030 : ED 7B 96 FA 21 4F FD 11
4038 : etc.
```

This part of the program is the key to the deprotection of the program. NB. if using an MTX 500, the dump will start at #8000. Also note that some games might not employ the IS IT A 500 OR 512 computer and might just load it into #8000, so keep this in mind. Whenever the reset keys are pressed, the information held in the system variables is erased, as are the first 8 bytes from the #4000 (or #8000). If we press L and enter 400B whilst in Panel, then listing 1 should appear. This tells us that the programmer has used the basic invisible code. This code finishes at 401A. The next 13 bytes consists of 10 SAVE "TOADO". This takes us up to 4028 in the Panel dump, and the most important part.

```
4028  :  10 20   :line length
         64 00   :line number in basic is 64 hex or 100 dec.
         C2      :the TOKEN or CODE in basic
         08 20   :symbol table pointer, end-code-7
         F3      :DI, disable basic
         etc     :the rest of listing 2.
```

These few bytes will give us vital information, especially in the length of code, which we will use to save the code to disc.

Before continuing, here is a quick check to see if our code will run even though we have RESET the machine. Find the memory address for the DI command, which is #402F, and change this into its decimal equivalent i.e. 16431. Leave Panel, B followed by Y. Once back in basic, type RAND USR(16431) <RET>. This should restart the code, if not, then maybe the programmer has used another technique.

Now RESET the machine again. The information we need now is the line length in decimal and the start of basic line 100 in decimal. Line 100 starts at #4028 or 16424, and its line length is #2010 8208 decimal. Now type:

USER WRITE "TOADO.COD",16424,8208 <RET>

This will save the Actual program code as a data file. We have deliberately not saved the code from #4000 to #4027. We need these 39 bytes for another purpose. Type the following listing into the computer:

Listing 3:

```
10 USER READ "TOADO.COD",16424     ;this loads the TOADO data file back
                                    into memory.
20 RAND USR(16431)                 ;this is the start of the program.
```

Now type USER SAVE "TOADO.BAS" <RET>, this will save this 2 line command which is <39 bytes long. If we now type:

USER LOAD "TOADO.BAS" <RET>
then RUN <RET>

the program should auto-run itself.

Lines 10 and 20 reside just below the program code in memory. Note that not all programs use the basic invisible part, so there isn't enough room for both the read and rand user commands, one of these will have to be typed manually. This occurs in DRAUGHTS.

The user can save the codes of a number of programs and then have another program which asks the user which program he wants to run, and gives him all the LOAD and Running instructions rather than have a lot of small basic files on disc.

I hope this article has achieved its objectives and I also hope it was an interesting read for programmers of other machines who I hope will go away and see if these effects can be used on their machines.✶

Morse Code by Paul Wood

```
0 REM **********************
1 REM ***** MORSE CODE *****
2 REM ****                ****
3 REM **** by Paul Wood ****
4 REM ****                ****
5 REM ****    3/3/85      ****
6 REM **********************
10 CSR 12,0: PRINT "MORSE TUTOR."
20 CSR 12,1: PRINT "~~~~~~~~~~~~~"
30 CSR 7,3: PRINT "Enter letter or Number."
40 GOSUB 1000
50 LET A=ASC(INKEY$)
60 IF A=-1 THEN  GOTO 50
70 CSR 16,6: PRINT CHR$(A)
80 IF A>=48 AND A<=57 THEN  LET A=A-48: GOTO 150
90 IF A>=65 AND A<=91 THEN  LET A=A-55: GOTO 150
100 IF A>=97 AND A<=123 THEN  LET A=A-87: GOTO 150
110 GOTO 50
150 LET B=C(A+1)
160 IF B=0 THEN  GOTO 50
170 LET D=B-(INT(B/10)*10)
180 IF D=1 THEN  LET P=100
190 IF D=2 THEN  LET P=300
200 SOUND 1,200,15: PAUSE P: SOUND 1,0,0
210 LET B=INT(B/10): PAUSE 100
220 GOTO 160
1000 DIM C(36)
1010 DATA 22222,22221,22211,22111,21111,11111,11112,11122,11222,12222
1020 DATA 21,1112,1212,112,1,1211,122,1111,11,2221,212,11211,22,12,222,1221,2122
,121,111,2,211,2111,221,2112,2212,1122
1030 FOR X=1 TO 36: READ C(X): NEXT
1040 RETURN
```

and Morse Encoder

```
1 REM *****MORSE ENCODER*****
2 REM MESSAGE LENGTH 300 CHARACTERS
3 REM R.S.G.B. STANDARD
4 REM
5 REM DIMENSION FOR ALPHA,NUMERIC & OTHER CODE
10 DIM AL(26,5): DIM NU(10,6): DIM OTH(10,10): DIM W$(300)
20 FOR I=1 TO 26: READ AL(I,1): FOR J=1 TO AL(I,1): READ AL(I,J+1): NEXT J: NEXT
 I
30 DATA   2,1,3,4,3,1,1,1,4,3,1,3,1,3,3,1,1,1,1,4,1,1,3,1,3,3,3,1,4,1,1,1,1,2,1,
1,4,1,3,3,3,3,3,1,3,4,1,3,1,1,2,3,3,2,3,1,3,3,3,3,4,1,3,3,1,4,3,3,1,3,3,1,3,1
40 DATA   3,1,1,1,1,3,3,1,1,3,4,1,1,1,3,3,1,3,3,4,3,1,1,3,4,3,1,3,3,4,3,3,1,1
50 FOR I=1 TO 10: READ NU(I,1): FOR J=1 TO NU(I,1): READ NU(I,J+1): NEXT J: NEXT
 I
60 DATA   5,3,3,3,3,3,5,1,3,3,3,3,5,1,1,3,3,3,5,1,1,1,3,3,5,1,1,1,1,3,5,1,1,1,1,
1,5,3,1,1,1,1,5,3,3,1,1,1,5,3,3,3,1,1,5,3,3,3,3,1
70 FOR I=1 TO 6: READ OTH(I,1): FOR J=1 TO OTH(I,1): READ OTH(I,J+1): NEXT J: NE
XT I
80 DATA  6,1,1,3,3,1,1,6,1,3,1,3,1,3,6,3,3,1,1,3,3,5,3,1,1,3,1,5,3,1,1,1,3,8,1,1
,1,1,1,1,1,1
```

```
99 REM MAIN LOOP
100 CLS : PRINT "CONTROL SPEED OF TRANSMISSION."
110 PRINT "RATE 1 IS THE FASTEST"
120 PRINT "RATE 10 IS THE SLOWEST"
130 PRINT "PLEASE INPUT RATE ";: INPUT SP
140 PRINT : IF SP<1 OR SP>10 THEN  GOTO 130
150 LET SP=40+8*SP: REM TRANSMISSION RATE.
160 LET GAP=3*SP: LET WD=6*SP: REM GAP BETWEEN LETTERS & GAP BETWEEN WORDS
200 REM NOW FOR THE MESSAGE
210 PRINT : PRINT "PLEASE INPUT TEXT"
220 PRINT "300 CHARACTERS OR LESS (FOR AN 'ERROR'  INSERT A ! )"
230 PRINT : INPUT W$
240 REM DECODE SEQUENCE
250 FOR I=1 TO LEN (W$)
260 LET X=ASC(W$(I))
270 IF X>64 AND X<91 THEN  LET X=X-64: GOTO 500
280 IF X>47 AND X<58 THEN  LET X=X-47: GOTO 600
290 IF X=63 THEN  LET X=1: GOTO 700
300 IF X=46 THEN  LET X=2: GOTO 700
310 IF X=44 THEN  LET X=3: GOTO 700
320 IF X=45 THEN  LET X=4: GOTO 700
330 IF X=61 THEN  LET X=5: GOTO 700
340 IF X=33 THEN  LET X=6: GOTO 700
350 IF X=32 THEN  PAUSE WD
400 NEXT I
410 PRINT "MORE TEXT ?(Y/N)";
420 LET Z$=INKEY$: IF Z$="" THEN  GOTO 420
430 IF Z$="Y" THEN  GOTO 230
450 STOP
500 REM ALPHA GENERATOR
510 FOR J=1 TO AL(X,1): LET Y=AL(X,J+1): GOSUB 1000: NEXT J: PAUSE GAP: GOTO 400
600 REM NUMBER GENERATOR
610 FOR J=1 TO NU(X,1): LET Y=NU(X,J+1): GOSUB 1000: NEXT J: PAUSE GAP: GOTO 400
700 REM OTHER GENERATOR
710 FOR J=1 TO OTH(X,1): LET Y=OTH(X,J+1): GOSUB 1000: NEXT J: PAUSE GAP: GOTO 4
00
1000 SOUND 0,100,15: PAUSE (SP*Y): SOUND 0,0,0: PAUSE SP: RETURN
```

Subscriptions

IF YOUR MEMBERSHIP NUMBER IS BETWEEN 0000 - 001900 (ALL LETTERS), YOU ARE NOW DUE TO RENEW YOUR SUBSCRIPTION IF YOU HAVE NOT ALREADY RENEWED ONCE. TO MAKE SURE OF RECEIVING YOUR NEXT ISSUE OF MEMOPAD PLEASE SEND YOUR SUBSCRIPTION TO REACH US NO LATER THAN 5TH JANUARY 1987.

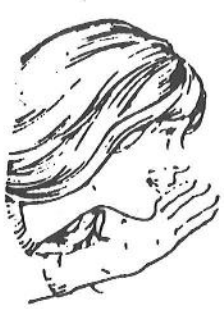
2ND YEAR RENEWALS ARE NOW DUE FOR RENEWAL (00900 - 01120 INCL.) UNLESS ALREADY RENEWED.

PLEASE NOTE NEW MEMBERSHIP PRICES ARE AS FOLLOWS:-

£18.00 U.K. MEMBERS

£27.50 E.E.C. MEMBERS

£36.50 OUTSIDE EUROPE ★



View Letters TO THE EDITOR point

Gordon Carter from Trowbridge in Wiltshire wrote in with the following.

Dear Sir,

The enclosed routine should solve the problem for D. Clement who wishes to use other colours for programming. It should be entered and run, when it may be overwritten by another program or deleted by using the AUTO facility. The second routine will be even shorter than the "Bit Mapping - Part 2" routine for printing to the screen, just set bit 7 of the last character and call #1A70.

Yours faithfully,
Gordon F. Carter

```
1 REM PROGRAMMING COLOURS
10 FOR A=62720 TO 62728
20 READ B
30 POKE A,B
40 NEXT
60 POKE 65519,245
70 POKE 65517,195
80 STOP
90 DATA  62,23,211,2,62,135,211,2,201
100 REM 2ND ITEM OF DATA (23) IS COLOUR AS 16*INK+PAPER COLOURS
110 REM i.e. ink 1,paper 7, (16+7=23)

1 REM PRINT TO SCREEN
10 CODE

401D          RST 10
401E          DB #4C                    ;VS 4 CLS
401F          LD HL,BUFFER
4022          CALL #1A70
4025          RET
4026 BUFFER:  DB "THIS IS A TES",#D4
4034          RET

Symbols:
BUFFER   4026

20 GOTO 20
```

I wonder whether any readers have yet succeeded in sorting out the machine-code address modes of the SDX/FDX 80-column board.

This is obviously quite a sophisticated piece of hardware which should be capable of doing as much in other languages running under CP/M as with the two BASICs supplied with the system. By using the series of CONTROL and ESCAPE codes given in the manual, I have been able to write simple graphics libraries in PASCAL and FORTRAN providing CSR, CLS, PLOT, LINE, ATTR and COLOUR commands and access to the alternate character sets and bit-imaged graphics sets, but the device is certainly capable of screen windowing via a CRVS command setting up a type-3 screen, although the GENPAT command on the disc BASICs appears to work only with the 9918A co-processor on the 40-column screen.

Incidentally, has anyone a copy of TVI.COM (the Televideo terminal emulator) which was not supplied with my systems, as I have acquired several programs which run more conviently on a Televideo installation?

I append a copy of the FORTRAN graphics mentioned in my letter. Readers may find it useful and it may also stimulate an interest in a language not normally thought of either as suitable for micros or as having a useful graphics facility.

Dear Brian,

Send in a disc and we will supply you with a copy of TVI.COM.

```
C  SCREEN FUNCTIONS
C  FLAG FOR PLOT ROUTINE
       LOGICAL FUNCTION WITHIN (IX, IY)
       LX = (IX .GE. 0) .AND. (IX .LE. 159)
       LY = (IY .GE. 0) .AND. (IY .LE. 95)
       WITHIN = LX .AND. LY
       RETURN
       END
C
       SUBROUTINE PAGE
       CALL PUT (12)
       RETURN
       END
C
       SUBROUTINE CSR (IX, IY)
       CALL PUT (3)
       CALL PUT (IX + 32)
       CALL PUT (IY + 32)
       RETURN
       END
C
       SUBROUTINE PLOT (IX, IY)
       CALL PUT (1)
       CALL PUT (IX + 32)
       CALL PUT (IY + 32)
       RETURN
       END
C
       SUBROUTINE LINE (IX1, IY1, IX2, IY2)
       CALL PUT (2)
       CALL PUT (IX1 + 32)
       CALL PUT (IY1 + 32)
       CALL PUT (IX2 + 32)
       CALL PUT (IY2 + 32)
       RETURN
       END
C
       SUBROUTINE ATTR (M)
       CALL PUT (6)
       CALL PUT (M)
       RETURN
       END
C  WAIT FOR CARRIAGE RETURN
       SUBROUTINE WAITCR
       INTEGER CR
9999   CALL CIN (CR)
       IF ( COMP (CR, #0D00, 1) .NE. 0 ) GO TO 9999
       RETURN
       END
C  PRINT GRAPHICS CHARACTER
       SUBROUTINE GCHAR (N)
       CALL ATTR (132)
       CALL PUT (N)
       CALL ATTR (5)
       RETURN
       END
```

Dear sir,

First of all I must thank Mr. E. Roy for the EDSDX patch, THANKS.

As I use EDASM a lot I wanted an easier way of loading it, the following is the way in which I achieved this.

First load EDASM and then add the following lines:

```
120 USER SAVE"EDASM.RUN"
130 RUN
```

Now save EDASM by typing GOTO 120. After it has been saved to disc it will auto run and this may cause spurious lines to be displayed, ignore this and reset the system.

Next type in the short progam that follows and save this as an auto run program by typing GOTO 150.

In future all thats needed to load Disc EDASM is to load this program and type USER E <RET>.

```
10 PRINT ">>>>>>>>> LOADING DISC EDASM <<<<<<<<<"
20 PRINT "-----------------------------------------"
30 PRINT
40 PRINT "LOADING DISC UTILITIES...";
50 USER  READ "EDSDX",63000
60 RAND USR(63000)
70 PRINT "** OK **"
80 PRINT
90 PRINT "LOADING  EDASM........."
100 PRINT
110 PRINT : PRINT "PROGRAM LENGTH 18K"
120 PRINT : PRINT "14 SEC LOAD TIME......."
130 PRINT : PRINT "TYPE <USER E> TO START."
140 USER  LOAD "EDASM.RUN"
150 USER  SAVE "ANY NAME"
160 RUN
```

Yours faithfully
L. R. Whalley

The following comments were sent in by Brian Penny from Bower Grange, Bedlington, Northumberland.

Dear Sir,

I would like to thank you for your letter dated 16th November, and would inform you that I have now received the replacement PAL Board from Paul Parry at M.C.L., also the MTX 500 seems to be ok now. The old unit has been returned to them by Recorded Delivery today, and I would like to thank you for your kind assistance and help over this problem, as it was appreciated.

Yours sincerely
Brian Penny

E. Tomlinson from Bowness on Windermere wrote in with a couple of queries.

Dear Sir,

I am writing to seek your assistance with work on the MTX 512.

1. As an experiment, I have been endeavouring to develop a simple form of word processor using the keyboard scan routine to print to the screen and a screen dump as given in the recent issue of 'MEMOPAD' to obtain a printout.

This routine is illustrated by program 1, but the printout following the 1st print appears haphazard as you will note on the copy following the listing.

Dumping screen inputs to the printer using this same routine appears to work satisfactorily. For other methods of text printing on the screen as witness programs 2-4 enclosed.

Your advice on this reason of failure would be appreciated.

2. Some time ago I was advised by Memotech that the following routine would allow saving of data to tape, but though I have changed BC for DE I still find the screen appears blank and the keyboard locked and there was no recording on tape.

Again a fuller explanation of its use and possible cause of failure would be helpful.

```
LD HL, (START ADDRESS OF DATA)
LD BC, (NO OF BYTES)
LD A, 0 (0 FOR SAVE, 1 FOR LOAD)
LD (#FD68), A
CALL #AAE
RET
```

I tried to use the above listing preceeded by the following:-

```
5 DIM A(10)
10 FOR J=1 TO 10
20 LET A(J)=J*5
30 POKE (49999+J), A(J)
40 NEXT J

THEN

50 CODE

LD HL, 50000
LD DE, 10                (Changing BC for DE as advised by you)
LD A, 0
LD (#FD68), A
CALL #AAE
RET
```

Yours faithfully,

E. Tomlinson

ED I fail to understand why you cannot get the data save/load routine to work. I can only surmise that you typed in the machine code routine then typed in the Basic program as an afterthought. If you did, and you did not re-assemble your code, the program would lock up.

Whenever you insert code or Basic IN FRONT of machine code you must go back into the machine code - in your case by typing "ASSEM 50" and then exit. This will re-assemble the code at the correct addresses.

I have personally tested this routine and it works perfectly. Your second routine will NOT work because you are using BC. and you are calling a ROM routine! The ROM expects the necessary set up in HL & DE. The correct routine is:

```
TAPE:   LD  HL, START ADDRESS
        LD  DE, NO OF BYTES
        LD  A,OOH  ;  OOH to SAVE / 01 to LOAD
        LD  (#FD68),A
        CALL #OAAE ; Go to ROM
        RET        ; RETURN TO CALLER
```

Unfortunately, your program is too long to include in Viewpoint. However,if you delete address #4012 and insert the following two lines at #4012 the routine should work perfectly.

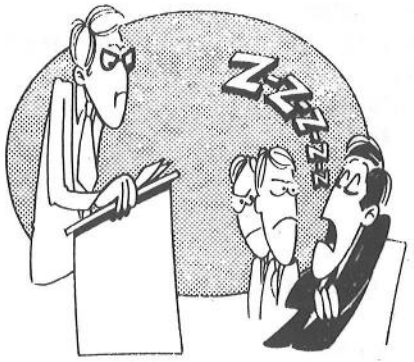
```
#4012  LD  B,A
       CALL #CE3
```
★

Pascal Save and Load to Disc by L Whalley

When using these two programs the following rules must apply.

1: After writing your Pascal program exit to BASIC using the [B] command. Then load the "SAVE.PAS" program and enter the name of the file and [RET]. This will save two programs onto disc NAME.SRC and NAME.VAR.

2: To load a file from disc first make sure you have initialise Pascal by typing ROM 2 then RET. Exit from Pascal using the [B] command and load the "LOAD.PAS" program and follow the instructions. After loading is complete enter Pascal, BUT DO NOT typing [RET]. Type CRTL X and there's your file.★



SAVE AND LOAD PASCAL FILES TO DISC.

```
10 REM SAVE PASCAL SOURCE CODE.
11 REM ------------------------------
12 REM
30 LET L=PEEK(21307)
40 LET H=PEEK(21308)
50 LET ADDR=H*256+L
60 LET LENGTH=ADDR-32768
```

```
70 CLS
80 PRINT "ENTER NAME OF FILE TO SAVE..."
90 PRINT
100 INPUT NAME$
120 CLS : PRINT "SAVING ";NAME$+".SRC";" TO DISC"
125 USER WRITENAME$+".VAR",21303,6
130 USER WRITENAME$+".SRC",32768,LENGTH
140 STOP
150 USER  SAVE "SAVE.PAS"
160 RUN
200 REM ***** XPLAN *****
210 REM ----------------------
220 REM
230 REM LINE 30:
240 REM GET LOW ORDER ADDRESS OF FILE.
250 REM LINE 40:
260 REM GET HIGH ORDER ADDRESS OF FILE.
270 REM LINE 50:
280 REM  CALCULATE END ADDRESS OF FILE.
290 REM LINE 60:
300 REM CALCULATE LENGTH OF FILE.
310 REM
320 REM ----------------------------------------
325 REM
330 REM ALL FILES SAVED HAVE .SRC
340 REM APPENDED TO THE FILE NAME
345 REM TO SHOW IT IS THE  SOURCE FILE.
350 REM THE RELEVENT VARIABLES ARE
360 REM SAVE ALONG WITH THE FILE.
370 REM ALL THIS IS DONE FOR YOU ALL
380 REM YOU HAVE TO DO IS TYPE IN THE
390 REM FILE NAME (NO EXTENSIONS).
395 REM
400 REM ******************************
```


```
10 REM LOAD PASCAL FILES FROM DISC.
20 PLOD "HELP": REM LEAST YOU FORGET.
25 CLS : PRINT "THE FILES ON THIS DISC ARE AS FOLLOWS :"
28 PRINT
30 USER DIR
40 PRINT
42 PRINT "----------------------------------------"
44 PRINT
46 PRINT "Enter the file name or <RET> if no file"
48 PRINT "is to be loaded."
49 PRINT
50 INPUT NAME$
52 IF NAME$="" THEN  GOTO 100
70 CLS
80 PRINT "LOADING ";NAME$+".SRC"
85 USER  READ NAME$+".VAR",21303
90 USER  READ NAME$+".SRC",32768
100 STOP
110 USER  SAVE "LOAD.PAS"
120 RUN
200 REM
205 REM ----------------------------------------
208 REM
210 REM THE NODDY PROGRAM IN LINE 10
220 REM IS JUST AN AID TO YOU LEAST YOU
230 REM FORGET JUST WHAT IT IS YOU
240 REM DO.
250 REM AS PER THE XPLAN GIVEN.....
260 REM
270 REM ----------------------------------------
```

Blitz by M. R. Allcorn

```
1 GOTO 22
2 CLEAR
4 GENPAT 1,143,62,62,28,62,62,62,28,8
5 GENPAT 1,144,24,24,102,102,219,219,219,255
6 GENPAT 1,145,24,24,24,24,60,126,231,231
7 GENPAT 1,146,24,60,126,219,153,153,255,255
8 GENPAT 1,147,255,153,255,153,255,153,255,255
9 GENPAT 1,148,231,231,231,255,255,231,231,231
10 GENPAT 1,149,255,219,165,219,165,219,165,255
11 GENPAT 1,150,255,129,129,129,129,129,129,255
12 GENPAT 1,151,0,112,248,248,255,255,127,16
13 GENPAT 1,152,7,1,50,116,255,255,255,6
14 GENPAT 1,153,248,32,64,128,248,248,240,0
15 GENPAT 2,147,17,26,26,26,26,26,26,26
16 GENPAT 2,148,31,26,24,26,26,31,26,24
17 GENPAT 2,149,24,26,24,26,24,26,24,26
18 GENPAT 2,150,16,18,11,20,22,24,26,16
19 GENPAT 2,151,5,149,245,101,53,53,53,21
20 GENPAT 2,152,53,53,53,53,53,53,53,5
21 GENPAT 2,153,53,53,53,53,53,53,49,53
22 VS 4: CTLSPR 2,0: INK 1: PAPER 5: COLOUR 4,1: CLS : FOR X=0 TO 31: CSR X,22: PRINT CHR$(150);
: NEXT X
23 FOR X=0 TO 31: LET HE=14+INT(RND*5): LET CH=INT(RND*3+147)
24 COLOUR 1,1: CSR X,HE: PRINT CHR$(CH-3)
25 FOR Z=HE+1 TO 21: CSR X,Z: PRINT CHR$(CH);: NEXT Z: NEXT X: SOUND 3,1003,14
26 CSR 0,23: FOR X=0 TO 7: COLOUR 3,8-X: LINE 0,X,255,X: NEXT X
27 LET B=100: LET PY=1: LET PX=0: LET P$=" "+CHR$(151)+CHR$(152)+CHR$(153): LET BY=0: LET BX=0:
LET BF=0: LET B$=CHR$(143)
28 CSR 0,0: PRINT "Bombs Left";B
29 CSR PX,PY: PRINT P$;
30 IF ASC(SPK$)>143 THEN GOTO 40
31 LET PX=PX+1: IF PX>31 THEN LET PX=0: LET PY=PY+1
32 IF PX=25 AND PY=22 THEN GOTO 38
33 IF BF=1 THEN GOSUB 36 ELSE PAUSE 37
34 IF INKEY$="" THEN GOTO 29
35 IF PY=22 OR BF=1 OR B=0 THEN GOTO 29 ELSE LET BF=1: LET B=B-1: LET BX=PX+1+32*(PX=31): LET BY
=PY+1: CSR BX,BY: PRINT B$: CSR 10,0: PRINT B;"  ": GOTO 29
36 CSR BX,BY: PRINT " ": CSR BX,BY+1: IF SPK$<>" " THEN SOUND 0,1023,15: FOR Z=BY TO 22: CSR BX,
Z: PRINT " ": NEXT Z: LET BF=0: SOUND 0,0,0: RETURN
37 LET BY=BY+1: SOUND 0,50+BY,15: IF BY=22 THEN LET BF=0: SOUND 0,0,0: RETURN ELSE CSR BX,BY: PR
INT B$: RETURN
38 FOR X=2 TO 13: COLOUR 4,X: CRVS 5,1,X-2,X-2,36-2*X,28-2*X,32: COLOUR 2,X: CLS : NEXT X
39 VS 4: CSR 5,10: PRINT "SURVIVED:YOUR SCORE=";B*B: GOTO 41
40 FOR X=0 TO 15: PAUSE 15: COLOUR 4,X: NEXT X: COLOUR 4,6: CSR 10,10: PRINT "YOU'RE DEAD"
41 SOUND 0,0,0: SOUND 3,0,0: CSR 3,21: PRINT "PRESS ANY KEY TO PLAY AGAIN": PAUSE 100
42 IF INKEY$<>"" THEN GOTO 22 ELSE GOTO 42
```

and Disassembler

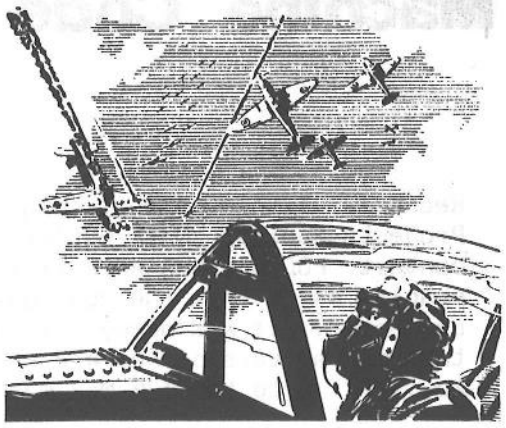
This listing is a Z80 disassembler which can cope with all of the published Z80 commands (i.e. all except the IX,IY extensions from MEMOPAD Vol. 2 No. 6)

```
1 DIM R$(8,4),RP$(4,2),CB$(11,3),I$(8,5),CO$(8,2),RST$(8,2),BL$(4,2)
5 LET H$="0123456789ABCDEF"
10 FOR X=1 TO 8: READ R$(X): NEXT
20 FOR X=1 TO 4: READ RP$(X): NEXT
30 FOR X=1 TO 11: READ CB$(X): NEXT
35 FOR X=1 TO 8: READ I$(X): NEXT
36 FOR X=1 TO 8: READ CO$(X): NEXT
38 FOR X=1 TO 8: READ RST$(X): NEXT
39 FOR X=1 TO 4: READ BL$(X): NEXT
40 FOR A=0 TO 8191
50 LET Y=PEEK(A): LET A$="": LET TA=A
60 IF Y=203 THEN GOSUB 140: GOTO 110
```

```
70 IF Y=237 THEN GOSUB 710: GOTO 110
80 IF Y=221 THEN GOSUB 1100: GOTO 110
90 IF Y=253 THEN GOSUB 1090: GOTO 110
100 GOSUB 190
110 LET Q=TA: GOSUB 1000: PRINT B$;" ";A$
120 NEXT A
130 STOP
140 LET A=A+1: LET Y=PEEK(A)
150 LET Y1=INT(Y/8)
160 IF Y1<8 THEN LET A$=CB$(Y1+1)+" "+R$(Y-8*Y1+1): RETURN
170 LET Y2=INT(Y1/8)
180 LET A$=CB$(Y2+8)+STR$((Y1-8)-8*INT((Y1-8)/8))+","+R$(Y-8*Y1+1): RETURN
190 IF Y<64 OR Y>127 THEN GOTO 210
200 LET A$="LD "+R$(INT((Y-56)/8))+","+R$(Y+1-8*INT(Y/8)): RETURN
210 IF Y<64 OR Y>191 THEN GOTO 230
220 LET A$=I$(INT((Y-120)/8))+","+R$(Y+1-8*INT(Y/8)): RETURN
230 IF Y>191 THEN GOTO 490
240 IF Y=0 THEN LET A$="NOP": RETURN
250 IF Y=1 OR Y=17 OR Y=33 OR Y=49 THEN LET A$="LD "+RP$(INT(Y/16)+1)+",": LET Q=PEEK(A+1)+256*P
EEK(A+2): LET A=A+2: GOSUB 1000: LET A$=A$+B$: RETURN
260 IF INT((Y-4)/8)=(Y-4)/8 THEN LET A$="INC "+R$(INT(Y+4)/8): RETURN
270 IF INT((Y-5)/8)=(Y-5)/8 THEN LET A$="DEC "+R$(INT(Y+5)/8): RETURN
280 IF INT((Y-6)/8)=(Y-6)/8 THEN LET A$="LD "+R$(INT(Y+6)/8)+",": LET Q=PEEK(A+1): LET A=A+1: LE
T B$="£": GOSUB 1020: LET A$=A$+B$: RETURN
290 IF Y=3 OR Y=19 OR Y=35 OR Y=51 THEN LET A$="INC "+RP$(INT((Y+13)/16)): RETURN
300 IF Y=11 OR Y=27 OR Y=43 OR Y=59 THEN LET A$="DEC "+RP$(INT((Y+5)/16)): RETURN
310 IF Y=9 OR Y=25 OR Y=41 OR Y=57 THEN LET A$="ADD "+RP$(3)+","+RP$(INT((Y+10)/16)): RETURN
320 IF Y=8 THEN LET A$="EX AF,AF'": RETURN
330 IF Y=2 THEN LET A$="LD (BC),A": RETURN
340 IF Y=18 THEN LET A$="LD (DE),A": RETURN
350 IF Y=7 OR Y=15 OR Y=23 OR Y=31 THEN LET A$=CB$(INT(Y÷9)/8)+"A": RETURN
360 IF Y=16 THEN LET A$="DJNZ ": GOSUB 1040: LET A$=A$+B$: RETURN
370 IF Y=10 THEN LET A$="LD A,(BC)": RETURN
380 IF Y=26 THEN LET A$="LD A,(DE)": RETURN
390 IF Y=24 THEN LET A$="JR ": GOSUB 1040: LET A$=A$+B$: RETURN
400 IF Y=32 OR Y=40 OR Y=48 OR Y=56 THEN LET A$="JR "+CO$(INT(Y-24)/8): GOSUB 1040: LET A$=A$+",
"+B$: RETURN
410 IF Y=34 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="LD ("+B$+"),"+RP$
(3): RETURN
420 IF Y=42 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="LD "+RP$(3)+",("+
B$+")": RETURN
430 IF Y=50 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="LD ("+B$+"),A": R
ETURN
440 IF Y=58 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="LD A,("+B$+")": R
ETURN
450 IF Y=39 THEN LET A$="DAA": RETURN
460 IF Y=47 THEN LET A$="CPL": RETURN
470 IF Y=55 THEN LET A$="SCF": RETURN
480 IF Y=63 THEN LET A$="CCF": RETURN
490 IF INT((Y-192)/8)=(Y-192)/8 THEN LET A$="RET "+CO$(INT((Y-184)/8)): RETURN
500 IF Y=241 THEN LET A$="POP AF": RETURN
510 IF Y=245 THEN LET A$="PUSH AF": RETURN
520 IF INT((Y-193)/16)=(Y-193)/16 THEN LET A$="POP "+RP$(INT((Y-177)/16)): RETURN
530 IF INT((Y-197)/16)=(Y-197)/16 THEN LET A$="PUSH "+RP$(INT((Y-181)/16)): RETURN
540 IF INT((Y-194)/8)=(Y-194)/8 THEN LET A$="JP "+CO$(INT((Y-186)/8)): LET Q=PEEK(A+1)+256*PEEK(
A+2): LET A=A+2: GOSUB 1000: LET A$=A$+",+B$: RETURN
550 IF INT((Y-196)/8)=(Y-196)/8 THEN LET A$="CALL "+CO$(INT((Y-188)/8)): LET Q=PEEK(A+1)+256*PEE
K(A+2): LET A=A+2: GOSUB 1000: LET A$=A$+B$: RETURN
560 IF Y=195 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="JP "+B$: RETURN
570 IF Y=201 THEN LET A$="RET": RETURN
580 IF INT((Y-199)/8)=(Y-199)/8 THEN LET A$="RST "+RST$(INT((Y-191)/8)): RETURN
590 IF INT((Y-198)/8)=(Y-198)/8 THEN LET A$=I$(INT((Y-190)/8)): LET B$="£": LET Q=PEEK(A+1): LET
 A=A+1: GOSUB 1020: LET A$=A$+B$: RETURN
600 IF Y=205 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="CALL "+B$: RETUR
N
```

```
610 IF Y=211 THEN LET Q=PEEK(A+1): LET B$="£": GOSUB 1020: LET A=A+1: LET A$="OUT ("+B$+"),A": R
ETURN
620 IF Y=219 THEN LET Q=PEEK(A+1): LET B$="£": GOSUB 1020: LET A=A+1: LET A$="IN A,("+B$+")": RE
TURN
630 IF Y=217 THEN LET A$="EXX": RETURN
640 IF Y=227 THEN LET A$="EX (SP),"+RP$(3): RETURN
650 IF Y=235 THEN LET A$="EX DE,"+RP$(3): RETURN
660 IF Y=233 THEN LET A$="JP ("+RP$(3)+")": RETURN
670 IF Y=243 THEN LET A$="DI": RETURN
680 IF Y=251 THEN LET A$="EI": RETURN
690 IF Y=249 THEN LET A$="LD SP,"+RP$(3): RETURN
700 RETURN
710 LET A=A+1: LET Y=PEEK(A)
711 IF Y=112 OR Y=113 THEN RETURN
715 IF Y>127 THEN GOTO 920
720 IF INT((Y-56)/8)=(Y-56)/8 THEN LET A$="IN "+R$((Y-56)/8)+",(C)": RETURN
730 IF INT((Y-57)/8)=(Y-57)/8 THEN LET A$="OUT (C),"+R$((Y-57)/8): RETURN
740 IF Y=66 OR Y=82 OR Y=98 OR Y=114 THEN LET A$="SBC "+RP$(3)+","+RP$((Y-40)/16): RETURN
750 IF Y=74 OR Y=90 OR Y=106 OR Y=120 THEN LET A$="ADC "+RP$(3)+","+RP$((Y-40)/16): RETURN
760 IF Y=68 THEN LET A$="NEG": RETURN
770 IF Y=69 THEN LET A$="RETN": RETURN
780 IF Y=70 THEN LET A$="IM 0": RETURN
790 IF Y=71 THEN LET A$="LD I,A": RETURN
800 IF Y=67 OR Y=83 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="LD ("+B$+
"),"+RP$(2+(Y=67)): RETURN
810 IF Y=75 OR Y=91 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="LD "+RP$(
2+(Y=67))+",("+B$+")": RETURN
820 IF Y=77 THEN LET A$="RETI": RETURN
830 IF Y=79 THEN LET A$="LD R,A": RETURN
840 IF Y=86 THEN LET A$="IM 1": RETURN
850 IF Y=87 THEN LET A$="LD A,I": RETURN
860 IF Y=94 THEN LET A$="IM 2": RETURN
870 IF Y=95 THEN LET A$="LD A,R": RETURN
880 IF Y=103 THEN LET A$="RRD": RETURN
890 IF Y=111 THEN LET A$="RLD": RETURN
900 IF Y=115 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="LD ("+B$+"),SP":
 RETURN
910 IF Y=123 THEN LET Q=PEEK(A+1)+256*PEEK(A+2): LET A=A+2: GOSUB 1000: LET A$="LD SP,("+B$+")":
 RETURN
920 IF Y<160 OR Y>191 THEN RETURN
930 IF INT((Y-160)/8)=(Y-160)/8 THEN LET A$="LD"+BL$((Y-152)/8): RETURN
940 IF INT((Y-161)/8)=(Y-161)/8 THEN LET A$="CP"+BL$((Y-153)/8): RETURN
950 IF INT((Y-162)/8)=(Y-162)/8 THEN LET A$="IN"+BL$((Y-154)/8): RETURN
960 IF Y=163 OR Y=171 THEN LET A$="OUT"+BL$((Y-155)/8): RETURN
970 IF Y=179 OR Y=187 THEN LET A$="OT"+BL$((Y-155)/8): RETURN
999 RETURN
1000 LET B$="£"+H$(INT(Q/4096)+1): LET Q=Q-4096*INT(Q/4096)
1010 LET B$=B$+H$(INT(Q/256)+1): LET Q=Q-256*INT(Q/256)
1020 LET B$=B$+H$(INT(Q/16)+1): LET Q=Q-16*INT(Q/16)
1030 LET B$=B$+H$(Q+1): RETURN
1040 LET Q=A+2+PEEK(A+1)+256*(PEEK(A+1)>127): LET A=A+1: GOTO 1000
1090 LET A=A+1: LET Y=PEEK(A): LET RP$(3)="IY": LET R$(7)="(IY)": GOTO 1110
1100 LET A=A+1: LET Y=PEEK(A): LET RP$(3)="IX": LET R$(7)="(IX)"
1110 IF Y=203 THEN GOTO 2000
1120 IF Y=237 THEN GOSUB 710: LET RP$(3)="HL": RETURN
1130 GOSUB 190: LET RP$(3)="HL": LET R$(7)="(HL)"
1140 IF Y=52 OR Y=53 THEN GOTO 2020
1160 IF Y=54 THEN LET Q=PEEK(A+1): LET A=A+1: LET B$="£": GOSUB 1020: LET A$="LD ("+RP$(3)+"+"+B
$+"),": LET Q=PEEK(A+1): LET B$="£": GOSUB 1020: LET A$=A$+B$: RETURN
1170 IF Y>63 OR Y<192 THEN GOTO 2010
1180 RETURN
2000 GOSUB 140: LET RP$(3)="HL": LET R$(7)="(HL)"
2010 IF INT((Y-6)/8)<>(Y-6)/8 THEN RETURN
2020 LET A$=LEFT$(A$,LEN(A$)-1)
2030 LET Q=PEEK(A+1): LET A=A+1: LET B$="£": GOSUB 1020
```

```
2040 LET A$=A$+"+"+B$+")": RETURN
60000 DATA B,C,D,E,H,L,(HL),A
60010 DATA BC,DE,HL,SP
60020 DATA RLC,RRC,RLB,RR,SLA,SRA, ,SRL,BIT,RES,SET
60030 DATA ADD A,ADC A,SUB,SBC A,AND ,XOR,OR,CP
60040 DATA NZ,Z,NC,C,PO,PE,P,M
60050 DATA 0,8,10,18,20,28,30,38
60060 DATA I,D,IR,DR
```



Spitfire

```
10 VS 4: COLOUR 4,1: COLOUR 2,1: COLOUR 0,1: CLS : INK 12
20 PLOT 93,110: ANGLE -1.57: DRAW 11: ANGLE -PI: DRAW 2: ANGLE -1.57: DRAW 5: ANGLE 0: DRAW 7: A
NGLE 1.57: DRAW 5
30 ANGLE -PI/2: DRAW 8: PHI PI/2: DRAW 10: PHI 1.4: DRAW 9: PHI 1.2: ARC 12,1: ANGLE 3.25: DRAW
40: ARC 8,-1.3: DRAW 12: ARC 11,2.4: ARC 22,.5
40 INK 7: ARC 9,1.2: DRAW 56: ANGLE .9: ARC 21,-.7: ARC 18,-1.4: ARC 4,-1.4: ANGLE -PI: DRAW 30
50 ANGLE 0: DRAW 5: ANGLE -1.5: DRAW 4: PHI 1.5: DRAW 15: PHI 1.3: DRAW 4
60 ANGLE 0: DRAW 2: ANGLE -1.2: ARC 11,1.2: ANGLE 1.57: DRAW 1: ANGLE PI: ARC 8,-1.1
66 ANGLE 0: DRAW 1: ANGLE -1.2: ARC 9,1.2: ANGLE 1.57: DRAW 1: ANGLE PI: ARC 7,-1.1
70 ANGLE PI: DRAW .5: ANGLE 0: ARC 21,.4: ARC 11,.6: ANGLE 1.57: DRAW 9: ANGLE 0: ARC 7,-.7: ARC
 5,-1.7: ARC 7,-.7: ANGLE 1.57: DRAW 9
80 INK 12: ANGLE 2.9: ARC 38,.21: ANGLE PI+.3: DRAW 7: ANGLE .3: DRAW 7: ANGLE 2.3: DRAW 7: ANGL
E 2: DRAW 1
90 INK 12
100 FOR X=37 TO 161
110 INK 12: LET Y=90+5*(X>100): LET F=0: IF X>151 THEN INK 10
120 LET A$=GR$(X,Y,1)
130 IF A$<>CHR$(0) THEN LET F=1
140 IF A$=CHR$(0) AND F=1 THEN GOTO 200
150 PLOT X,Y
160 LET Y=Y+1
170 GOTO 120
200 LET Y=89+5*(X>100): LET F=0
210 LET A$=GR$(X,Y,1)
220 IF A$<>CHR$(0) THEN LET F=1: IF Y>70 AND X>95 AND X<135 THEN LET F=0: INK 7: ATTR 3,1
225 IF F=1 AND Y>68 AND X>103 AND X<120 THEN LET F=0: INK 7: ATTR 3,1
230 IF A$=CHR$(0) AND F=1 THEN GOTO 300
240 PLOT X,Y
245 ATTR 3,0
250 LET Y=Y-1
255 IF (X<105 AND Y<90-X/8) OR (X>104 AND Y<90-(200-X)/8 AND X<152) THEN INK 7
260 GOTO 210
300 NEXT X
310 PLOT 152,110: PLOT 153,108: PLOT 153,107: PLOT 152,55: PLOT 153,57: PLOT 153,58
320 INK 12: LINE 24,115,34,115: LINE 22,100,30,100
330 INK 7: LINE 22,82,32,82: LINE 85,66,95,66
340 INK 15: CSR 6,19: PRINT "THE SPITFIRE"
```

Issue 3 & 4 special

memopad vol 0011

Machine Code Conversion Routines

by K. Hook

Recently I had a real 'pig' of a conversion to finish. Our intrepid Software Projects Manager, Mark Butler, signed a contract with a company for Syntax to convert FOOTBALL MANAGER from Amstrad to Msx. On the surface this presented no problem as we knew the code was all written in Basic, and as far as we were concerned, it was money for 'old rope'. However, when we counted up the bytes, the Amstrad version was 41k which meant that there was no way the original code would fit into the 32k Msx machine.

"No problem !", I declared. "I'll re-write it in machine code." I thought that this was the sensible alternative - it would make the very slow program run a lot faster and leave more room to improve the match-stick graphics. Oh! How I wish I had never uttered those brave words. Nearly every other line contained code on the lines ...

```
2155 FOR I = B TO B+20:IF V(I,B) = N(I,3)*V(I) THEN CV(I,B) = 45000/2+V(I)
```

Very nice, and easy to write in Basic but putting it into machine code was another matter. Anyway, I couldn't find any 24/32 bit maths routines in my archives so I had to start from scratch. While I was writing the routines I realised that other programmers must have been in the same sort of predicament and may even have put some off finishing that 'cupboard masterpiece'. So here are my first drafts. I have managed to speed two of them up significantly from the routines I present here, but I thought that we might offer a prize to the best 24/32 bit maths routine sent in by users..... let's have your thoughts down on paper, and please, because of the complexity of the subject, comment as much as possible.

One of the first problems I encountered was how to accept a Single Precision number from the keyboard in Ascii and convert it into a binary value that could be worked on by the program.

Ascii numbers are in the region 30H (0) to 39H (9) so stripping the number into its binary equivalent is just a matter of subtracting 30H. However, a number in the region of 145,789 bears no relationship to its binary equivalent. This number would be entered from the keyboard and accepted into a buffer as:

31H:34H:35H:37H:38H:39H:

If we strip 30H, the number that is left, even though it is in binary format is not correct because 145,789 = 2397DH in binary. The formula used for converting it to a binary number is :

GET NUMBER * 10 : GET NEXT NUMBER ADD IT AND THEN * 10 REPEAT UNTIL THE LAST DIGIT IS ENCOUNTERED WHICH IS THEN ADDED.

The routine that follows first multiplies zero by 10 - this saves having to check for last digit just to add it. Input, from the keyboard, is terminated with a Carriage Return <CR> or 0DH, and each digit is put into a buffer **ASCBF**. The program stores its data in a work area named **WRKA** which is 4 bytes in length. To multiply by 10, in machine code, you have to carry out the following procedure:

Say number in HL registers.

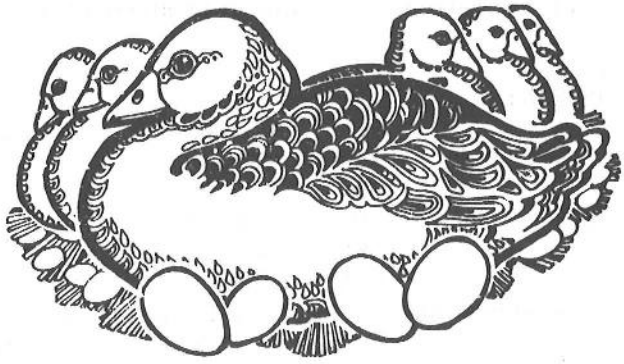
```
                ADD     HL,HL       ;*2
                PUSH    HL
                ADD     HL,HL       ;*4
                ADD     HL,HL       ;*8
                POP     DE          ;*2 BACK
                ADD     HL,DE       ;*10
```

Very long winded when you have to loop through the routine a few times the following code uses a 3 register shift left which is the same as multiplying by two.

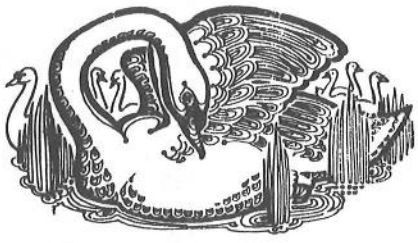
ASCII TO SINGLE PRECISION BINARY IN 24 BITS.

```
;Entry === ASCBUF contains Ascii input terminated with a CR.
;Exit ==== 24 bit binary number left in WRKA with LSB in WRKA, NSB in WRKA+1
;MSB in WRKA+2.
;All registers used and corrupted on exit.
;
;
ASCBIN:
                LD      HL,WRKA+3       ;Point to end of Work area
                XOR     A               ;Clear A reg
                LD      B,4
ZBF:            LD      (HL),A          ;Zero work area
                DEC     HL
                DJNZ    ZBF
;The above makes the four bytes in WRKA zero
                LD      HL,ASCBF        ;GET ASCII BUFFER INTO HL
                LD      A,(HL)          ;GET FIRST CHAR
                CP      0DH             ;IS IT <CR>
                JR      Z,ACFIN         ;YES SO FINISH
                SUB     30H             ;GET RID OF ASCII
                PUSH    HL              ;SAVE ASCBF
                PUSH    AF              ;SAVE FIRST DIGIT
                LD      HL,WRKA         ;EMPTY A PRESENT
                CALL    SPLD            ;PUT IT INTO C/DE
ACLP:           SLA     E               ;DON'T WANT TO INCLUDE CARRY HERE
                RL      D               ;NOW WE INCLUDE CARRY FROM E
                RL      C               ;INCLUDE CARRY FROM D
                                        ;NOW *2
                LD      HL,WRKA         ;GET BUFFER WORK AREA
                CALL    DWNLD           ;SAVE *2 IN WRKA
                SLA     E
                RL      D
                RL      C               ;*4
                SLA     E
                RL      D
                RL      C               ;*8
                LD      HL,WRKA         ;GET *2 BACK
                CALL    REGAD           ;ADD WRKA TO BECOME *10
                POP     AF              ;GET DIGIT BACK
                CALL    REGAD1          ;ADD IT TO VALE IN C/DE
                POP     HL              ;GET ASCBF BACK
                INC     HL              ;POINT TO NEXT CHARACTER
                LD      A,(HL)
                CP      0DH             ;IS IT CARRIAGE RETURN
                JR      Z,ACFIN         ;
```

```
                SUB     30H             ;NOW BINARY
                PUSH    HL              ;SAVE ASCBF
                PUSH    AF              ;SAVE DIGIT
                JR      ACLP            ;DO IT AGAIN SAM!
ACFIN:          LD      HL,WRKA
                CALL    DWNLD           ;SAVE BINARY IN WRKA
                RET                     ;RETURN TO CALLER
WRKA:           DS      4
ASCBF:          DS      7               ;SIX DIGITS ALLOWED + A <cr>
;
REGAD:

;Entry  HL points to memory area
;adds C/DE with (HL)
;EXIT   HL -> end of buffer
;C/DE holds value

                LD      A,(HL)
                ADD     A,E
                LD      E,A
                INC     HL
                LD      A,(HL)
                ADC     A,D
                LD      D,A
                INC     HL
                LD      A,(HL)
                ADC     A,C
                LD      C,A
                RET
```



```
REGAD1:

;Adds value in A reg to C/DE
;
                ADD     A,E
                LD      E,A
                LD      A,0             ;sO WE CAN ADD CARRY TO D & C
                ADC     A,D
                LD      D,A
                LD      A,0
                ADC     A,C
                LD      C,A
                RET

SPLD:

;ENTRY HL points to start of buffer where values are stored
;EXIT Buffer untouched
;C/DE hold value
;
                LD      E,(HL)
                INC     HL
                LD      D,(HL)
                INC     HL
                LD      C,(HL)
                INC     HL
                RET
```



```
DWNLD:

;ENTRY  HL = memory area
;C/DE = Values
;EXIT   C/DE  untouched
;       MEM = values
              LD      (HL),E
              INC     HL
              LD      (HL),D
              INC     HL
              LD      (HL),C
              RET
```

** You will ,more than likely, by now, have spotted how I shortened the main routine but I'll say no more.

```
    S.P BINARY TO ASCII

    ;SP NO IN BC/DE
    ;DE = DIVISION TABLE
    ;ASCII STORED IN ASCBF TERMINATED FF

    BINASC:     LD      DE,DIVTAB       ;TABLE OF CONSTANTS
                LD      HL,ASCCBF
                XOR     A
    CONLP:      CCF                     ;CARRY=1ST TIME SWITCH
                PUSH    AF              ;DIVISION LP (C)
                PUSH    HL
                PUSH    DE
                LD      HL,WRKA         ;PUT
                CALL    SPLD            ;PUT SP INTO BC/DE
                POP     HL              ;HL=DIVTAB
                LD      B,2FH           ;ASCII(30H-1) = "0"-1
    INKASC:     INC     B               ;INCREMENT ASCII CHARACTER
                LD      A,E             ;LSB
                SUB     (HL)
                LD      E,A
                INC     HL
                LD      A,D             ;NSB
                SBC     A,(HL)
                LD      D,A
                INC     HL
                LD      A,C
                SBC     A,(HL)
                LD      C,A
                DEC     HL
                DEC     HL              ;REALIGN TO 100,000 ETC
                JR      NC,INKASC       ;LOOP UNTIL <100,000
                CALL    REGAD           ;ADD 100,000 TO MAKE POSITIVE
                INC     HL              ;BUMP TO NEXT -> 10,000
                CALL    PUTBK           ;PUT REMAINDER BACK FOR CURRENT VALUE
                EX      DE,HL           ;DE=DIVTAB        ADDRESS
                POP     HL              ;HL=PBUF=ASCBF
```

```
                LD      (HL),B          ;SAVE ASCII DIGIT
                INC     HL              ;NEXT PRINT AREA
                POP     AF              ;GET CARRY SWITCH
                JR      C,CONLP
                INC     DE              ;WHEN ROUTINE FALLS THROUGH
                INC     DE              ;WE HAVE DIVIDED BY 10000,000,10,000
                                        ;SO BUMP TO 10,000 WHICH IS
                                        ;INTEGER DIVISION
                LD      A,04            ;NO OF DIGITS
                JR      NXTASC

    ;THIS ENTRY TO BE USED FOR INTEGER TO ASCII

INTEG:          LD      DE,DIVIT2       ;TABLE OF INTEGER CONSTANTS
                LD      A,5             ;NO OF ASCII DIGITS
NXTASC:         PUSH    AF              ;SAVE NO OF DIGITS
                PUSH    HL              ;SAVE PRINT BUFFER
                EX      DE,HL           ;HL = DIVT2
                LD      C,(HL)          ;PWR 10 IN BC
                INC     HL
                LD      B,(HL)          ;MSB    OF PWR
                PUSH    BC              ;SAVE IT
                INC     HL              ;NEXT VALUE IN PWR TABLE
                EX      (SP),HL         ;HL = VALUE IN BC
                EX      DE,HL           ;DE = VALUE
                LD      HL,(WRKA)       ;HL = CURRENT VALUE (INTEGER)
                LD      B,2FH           ;
INKAS2:         INC     B               ;DIVIDE CURRENT VALUE
                LD      A,L             ;BY POWERS OF TEN STARTING
                SUB     E               ; AT 10,000. REMAINDER FROM
                LD      L,A             ;EACH DIVISION IS ADDED
                LD      A,H             ;TO THE DIVISION AND SUM
                SBC     A,D             ; = DIVIDEND FOR NEXT DIVISION
                LD      H,A             ;COMPOUND SUBTRACTION
                JR      NC,INKAS2       ;IN OTHER WORDS!
                ADD     HL,DE
                LD      (WRKA),HL
                POP     DE              ;NXT PWR 10
                POP     HL              ;HL = ASCBF
                LD      (HL),B
                INC     HL
                POP     AF
                DEC     A
                JR      NZ,NXTASC
                LD      A,0FFH
                LD      (HL),A          ;TERMINATOR
                RET                     ;RETURN TO CALLER
ASCBF:          DS      7               ;SIX ASCII CHARS 999,999 + <CR>
DIVTAB:         DB      A0H,86H,01,10H,27H,00
DIVT2:          DB      10H,27H,0E8H,03,64H,00
                DB      0AH,00,01,00
WRKA:           DS      4
PUTBK:          EX      DE,HL           ;HL = DE : DE = DIVTAB
                LD      (WRKA),HL       ;LSB
                LD      H,B
                LD      L,C
                LD      (WRKA+3),HL     ;MSB
                EX      DE,HL
                RET
```

The above routine is complementary to the last one - it converts a Single Precision binary number to ASCII in order that the value can be printed to the screen.

The routine uses a lookup table in order to divide the binary number by powers of ten. Because we are dealing with 24-bit numbers, which can be in the range 0 - 999,999, the division table starts at 100,000 :-

```
100,000
10,000
10000    ;integer part
1000
100
10
1
```

The registers are divided by these constants until they become negative. The constant is then added back to the remainder and this becomes the current value and the next, lower, value in the look-up table is used and the process continues. Division is accomplished by successive subtraction, although I have used a compound subtraction algorithm in the integer portion of the conversion.

When the conversion is first entered we are dealing with 24 bits which are stored in registers C/DE (MSB to LSB). Once the routine falls through to INTEG we are then dealing with an integer value and this is handled in a different manner to the first part of the routine. In fact, you can use the routine from INTEG for 16 bit conversions.

The number is converted to ASCII by keeping a count in the B registers. The B reg is first aligned to 2FH which is one less than the ASCII value of 30H ("0"). Every time the subtraction is completed without a carry the B reg is incremented. When a carry is detected the INKASC loop is terminated and the B register holds the current ASCII value which is stored in the print buffer ASCBUF. The B reg is then decremented to 2FH and the whole process is repeated.

The print buffer ASCBUF is finally loaded with OFFH which denotes the end of number which can be used by a print routine to detect the 'end of buffer marker'.

Next time we will look at some Binary Coded Decimal routines and how easy it is to use this type of number in calculations and printing.

HAVE FUN!★

Screen Dump Routine by G. Carter

Memotech Basic provides us with a routine to take 8 pixels of screen memory and pass this to the printer [GR$(x, y, 8)]. This enables us to write a program in basic to dump the Graphic screen (VS 4) to our printer. The drawback to using this method is the time taken, usually 4 to 7 minutes.

The machine code routine here does the same job but much faster. The graphic screen (VS 4) can be dumped to the printer in two sizes, using the same routine. The first size prints a single dot for each pixel on the screen. The second size prints 4 dots per single pixel on the screen, two dots across and two dots down making the printed picture twice as wide and twice as deep.

The basis of the program is to first transfer the screen memory to user memory using the code in line 40, then the routine at line 20 will adjust the user memory to meet the printer requirements.

To set the routine to print single or double size, a flag routine is used by poking memory #C1F7 HEX (49655 decimal) with a 0 for single size or 1 for double. If only the top part of the screen is wanted then line 980 can be altered to the number of lines (of 8 pixels) down. i.e. half of the screen needs the following line:

```
980 FOR N=1 TO 12: GOSUB 20:NEXT
```

The printer control codes given are for the MANNESMAN TALLY, but are mostly compatible with other printers. For those with different printers the following details give the requirements for printer control codes.

Line 960. Set line spacing to 8/72".:- LPRINT CHR$(27);"A";CHR$(8);
Line 970. SET PRINTER TO UNIDIRECTION PRINT.:- LPRINT CHR$(27);"U";
CHR$(1);
SET PRINTER TO GRAPHIC MODE (640 DOTS PER LINE MAX):- LPRINT CHR(27);
CHR$(#4B HEX);CHR$(0);CHR$(N);...Where N=1 for single size and N=2 for double size. This last routine is incorporated in code line 20 as subroutine DTS (and DDTS) line #8053 to #8072.

To use normal printing after the graphics the printer control codes must be reset using the following code:- LPRINT CHR$(27);"@"; (RESETS ALL SPECIAL MODES) NOTE: this code resets top of form to power up state, and you may need to reset the top of form.

The screen dump can be used from any part of a program being called up by an INKEY$ call or from a menu as shown. If called by the inkey$ a simple routine such as follows will work well.

```
LINE 1000 GOSUB 40
LINE 1010 POKE 49655,0 :REM (or 1 for double size)
LINE 1020 POKE 49656,0:POKE 49657,194
LINE 1040 LPRINT CHR$(27);"A";CHR$(8);
LINE 1050 LPRINT CHR$(27);"U";CHR$(1);
LINE 1060 FOR N=1 TO 24:GOSUB 20:NEXT
```

LINE 1070 RETURN (TO REST OF PROGRAMME OR MENU) This routine will print a complete screen in single size in approx. 32 seconds and in double size in approx. 105 seconds. The routine at lines 600-770 enables a picture to be loaded into user memory from tape or saved to tape from user memory.

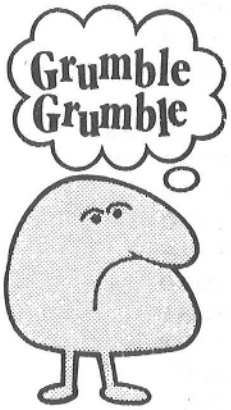
NOTE:If a mismatch occurs whilst verifying the program will jump to BASIC. From VS 4 in your own program GOTO line 400 to save the screen to user memory ready for printing or saving to tape.

```
10 GOTO 500
20 CODE

8010            CALL DTS
8013            CALL LINE
8016            CALL CRT
8019            LD A,(#C1F7)
801C            CP 0
801E            JR NZ,DB
8020            LD HL,(#C1F8)
8023            INC H
8024            LD (#C1F8),HL
8027            RET
8028 DB:        LD HL,(#C1F8)
802B            LD DE,#0004
802E            ADC HL,DE
8030            LD (#C1F8),HL
8033            CALL DTS
8036            CALL LINE
8039            CALL CRT
803C            LD HL,(#C1F8)
803F            LD DE,#00FC
8042            ADC HL,DE
8044            LD (#C1F8),HL
8047            RET
8048 CRT:       LD B,13
804A            CALL #0CE0
804D            LD B,10
804F            CALL #0CE0
8052            RET
8053 DTS:       LD B,27
8055            CALL #0CE0
8058            LD B,#4B
805A            CALL #0CE0
805D            LD B,0
805F            CALL #0CE0
8062            LD A,(#C1F7)
8065            CP 0
8067            JR NZ,DDTS
8069            LD B,1
806B            JR HR
806D DDTS:      LD B,2
806F HR:        CALL #0CE0
8072            RET
8073 LINE:      LD HL,(#C1F8)
8076            LD C,0
8078 L3:        LD B,8
807A L2:        PUSH BC
807B            PUSH HL
807C            CALL ADJ
807F            POP HL
8080            POP BC
8081            DEC C
8082            RET Z
8083            DJNZ L2
8085            LD DE,#0008
8088            ADC HL,DE
808A            JR L3
808C ADJ:       LD A,(#C1F7)
808F            CP 0
8091            JR NZ,DADJ
8093            LD BC,#0800
8096 LP:        BIT 7,(HL)
8098            JR Z,NX
809A            SET 7,C
809C NX:        RLC (HL)
809E            RLC C
80A0            INC HL
80A1            DJNZ LP
80A3            LD B,C
80A4            CALL #0CE0
80A7            RET
80A8 DADJ:      LD BC,#0400
80AB LPD:       BIT 7,(HL)
80AD            JR Z,DNX
80AF            SET 7,C
80B1            SET 6,C
80B3 DNX:       RLC (HL)
80B5            INC HL
80B6            RLC C
80B8            RLC C
80BA            DJNZ LPD
80BC            LD B,C
80BD            CALL #0CE0
80C0            CALL #0CE0
80C3            RET

Symbols:
DTS     8053    LINE    8073
CRT     8048    L3      8078
L2      807A    ADJ     808C
LP      8096    NX      809C
DB      8028    DADJ    80A8
LPD     80AB    DNX     80B3
DDTS    806D    HR      806F


25 RETURN
30 CODE

8235            LD HL,#C200
8238            LD DE,#1800
823B            CALL #AAE
823E            RET

Symbols:


35 RETURN
40 CODE

827F            DI
8280            XOR A
8281            OUT (2),A
8283            LD A,0
8285            OUT (2),A
8287            NOP
8288            NOP
```

```
8289          LD HL,#C200
828C          LD BC,#1800
828F LP:      IN A,(1)
8291          LD (HL),A
8292          INC HL
8293          DEC BC
8294          LD A,B
8295          OR C
8296          JR NZ,LP
8298          EI
8299          RET
```

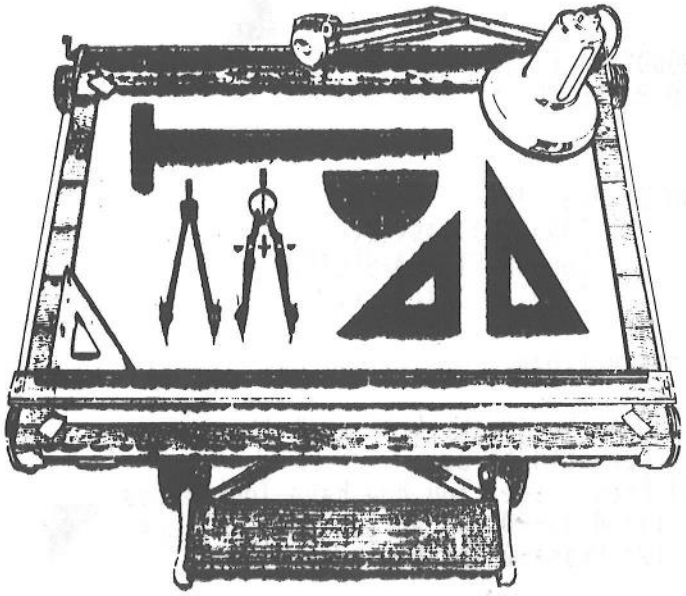
```
Symbols:
LP       828F
```

```
45 RETURN
200 REM DTS IN CODE 20 SETS DOT GRAPHICS TO 640 DOTS PER LINE WITH A TOTAL OF
256 DOTS
220 REM DDTS SETS DOTS TO MAX OF 512 DOTS
300 LET I$=INKEY$: IF I$="" THEN  GOTO 300
310 RETURN
340 PRINT : PRINT "IS THE CASSETTE READY ?"
350 GOSUB 300: IF I$="Y" OR I$="y" THEN  RETURN  ELSE  GOTO 350
400 GOSUB 40: REM SAVE VS 4 TO MEMORY AT #C200 HEX
500 VS 5: CLS
510 PRINT "1. PRINT SINGLE"
520 PRINT "2. PRINT DOUBLE"
530 PRINT "3. LOAD PICTURE FROM TAPE"
540 PRINT "4. SAVE PICTURE TO TAPE"
550 PRINT "5. VERIFY SAVED PICTURE"
570 GOSUB 300: LET I=VAL(I$): IF I>5 OR I<1 THEN  GOTO 570
590 ON I GOTO 500,900,800,600,700,750
599 REM LOAD A PICTURE FROM TAPE
600 POKE 64872,1: POKE 64871,0
610 GOSUB 340
630 GOSUB 30: GOTO 500
699 REM SAVE A PICTURE TO TAPE
700 GOSUB 40
710 POKE 64872,0: POKE 64871,0
720 GOSUB 340
730 GOSUB 30: GOTO 500
749 REM VERIFY PICTURE
750 POKE 64872,1: POKE 64871,1
760 GOSUB 340
770 GOSUB 30: GOTO 500
800 REM SET FLAG TO DOUBLE PRINT (@C1F7 HEX)
820 POKE 49655,1
850 GOTO 910
890 REM SET FLAG FOR SINGLE PRINT SIZE (@ C1F7 HEX)
900 POKE 49655,0
910 PRINT "IS THE PRINTER READY?[Y]"
920 GOSUB 350
940 REM POKE START OF PICTURE INTO BUFFER (@ C1F8 HEX)
950 POKE 49656,0: POKE 49657,194
955 REM SET LINE SPACE ON PRINTER TO 8/72 INCH
960 LPRINT CHR$(27);"A";CHR$(8);
965 REM SET PRINTER TO UNIDIRECTION PRINTING
970 LPRINT CHR$(27);"U";CHR$(1);
975 REM CALL MACHINE CODE FOR EACH LINE OF SCREEN (24 TIMES)
980 FOR N=1 TO 24: GOSUB 20: NEXT
990 GOTO 500
```

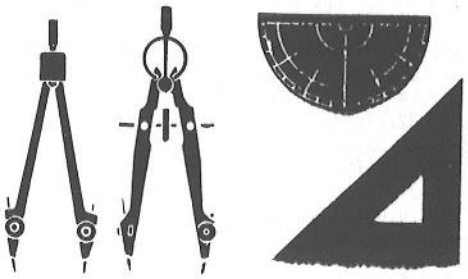


Colour Etch-A-Sketch

```
10 GOSUB 650
20 REM INITIALISE
30 CRVS 6,1,0,0,32,24,32
40 VS 6: COLOUR 0,14: COLOUR 1,1: COLOUR 2,15: COLOUR 3,1: COLOUR 4,14: CLS
50 FOR T=19 TO 22: FOR U=0 TO 31 STEP 4: CSR U,T: PRINT "    ": NEXT : NEXT
60 CSR 0,20: PRINT "INK "
70 CSR 0,21: PRINT "PAPER": CSR 24,20: PRINT "MODE :"
80 FOR T=1 TO 15: COLOUR 0,T: CSR T+6,20: PRINT " ";: CSR T+6,21: PRINT " ",: NEXT : COLOUR 0,14
90 COLOUR 2,14: LINE 55,16,176,16
100 LINE 55,32,176,32
110 LINE 55,16,55,32
120 LINE 176,16,176,32
130 LINE 55,24,176,24
140 GENPAT 3,0,0,127,65,93,85,93,65,127
150 CTLSPR 2,1
160 CTLSPR 6,0
170 CTLSPR 4,1
180 SPRITE 1,0,0,0,0,0,1
190 REM COLOUR CHANGE ROUTINE
200 LET L=1
210 CSR 6+APER,21: GOSUB 910: PRINT CHR$(147)
220 CSR 6+NK,20: GOSUB 890: PRINT CHR$(147)
230 COLOUR 0,14: COLOUR 1,1: CSR 24,21: PRINT "COLOURS"
240 IF L=1 THEN  GOTO 330
250 LET ZA=ASC(INKEY$): IF ZA=10 THEN  LET L=1: GOTO 330
260 IF ZA=-1 THEN  GOTO 250
270 COLOUR 0,NK: CSR NK+6,20: PRINT " "
280 IF ZA=8 THEN  LET NK=NK-1: IF NK<1 THEN  LET NK=15
290 IF ZA=25 THEN  LET NK=NK+1: IF NK>15 THEN  LET NK=1
300 GOSUB 890: CSR NK+6,20: PRINT CHR$(147)
310 IF ZA=26 THEN  LET ZA=-1: PAUSE 500: GOTO 420
320 GOTO 250
330 LET ZA=ASC(INKEY$): IF ZA=11 THEN  LET L=0: GOTO 250
340 IF ZA=-1 THEN  GOTO 330
350 COLOUR 0,APER: CSR APER+6,21: PRINT " "
360 IF ZA=8 THEN  LET APER=APER-1: IF APER<1 THEN  LET APER=15
370 IF ZA=25 THEN  LET APER=APER+1: IF APER>15 THEN  LET APER=1
380 GOSUB 910: CSR APER+6,21: PRINT CHR$(147)
390 IF ZA=26 THEN  LET ZA=-1: PAUSE 500: GOTO 420
400 GOTO 330
410 REM MAIN LOOP STARTS HERE
420 COLOUR 0,14: COLOUR 1,1: CSR 24,21: PRINT P$(ODE): ATTR 2,PQ(ODE,1): ATTR 3,PQ(ODE,2)
430 COLOUR 2,APER: COLOUR 3,NK
440 LET ZA=ASC(INKEY$)
450 IF ZA=-1 THEN  GOTO 600
460 IF ZA=128 THEN  GOTO 620
470 IF ZA=26 THEN  LET ZA=-1: PAUSE 500: GOTO 230
480 IF ZA=8 AND X>8 THEN  LET X=X-1
490 IF ZA=25 AND X<247 THEN  LET X=X+1
500 IF ZA=11 AND Y<183 THEN  LET Y=Y+1
510 IF ZA=10 AND Y>48 THEN  LET Y=Y-1
520 PLOT X,Y
530 IF INKEY$="E" THEN  COLOUR 0,15: FOR T=1 TO 17: CSR 1,T: PRINT "                                        ": NEXT
540 ADJSPR 1,1,INT(RND*14*2)
550 LET ZZ$=CHR$(ZA): IF ZZ$<"1" OR ZZ$>"4" THEN  GOTO 600
560 IF ZZ$="1" THEN  LET ODE=1: GOTO 420
570 IF ZZ$="2" THEN  LET ODE=2: GOTO 420
580 IF ZZ$="3" THEN  LET ODE=3: GOTO 420
590 IF ZZ$="4" THEN  LET ODE=4: GOTO 420
600 GOTO 440
610 REM HARD COPY
620 LPRINT CHR$(27);"P";CHR$(0);CHR$(27);"3";CHR$(24);
630 FOR PY=183 TO 48 STEP -8: LET PP$="": FOR PX=8 TO 247: LET PP$=PP$+GR$(PX,PY,8): NEXT PX: LPRINT CHR$(27);"K";CHR$(240);CHR$(0);
PP$: NEXT PY: GOTO 440
640 REM INSTRUCTIONS
650 LET NK=1: LET APER=15: DIM PP$(256)
660 LET X=8: LET Y=48
670 GENPAT 1,147,0,0,60,60,60,0,0,255
680 CTLSPR 4,0
690 GENPAT 0,95,0,0,0,0,0,0,0,255
700 GENPAT 0,92,0,0,0,0,0,0,0,0
710 PLOD "INST"
720 DIM P$(4,7)
730 LET P$(1)="DRAW   "
```



```
740 LET P$(2)="ERASE  "
750 LET P$(3)="OVER   "
760 LET P$(4)="MOVE   "
770 DIM PQ(4,2)
780 LET PQ(1,1)=0
790 LET PQ(1,2)=0
800 LET PQ(2,1)=1
810 LET PQ(2,2)=0
820 LET PQ(3,1)=0
830 LET PQ(3,2)=1
840 LET PQ(4,1)=1
850 LET PQ(4,2)=1
860 LET ODE=4
870 RETURN
880 REM COLOURS "CURSOR"
890 LET ZZ=NK
900 GOTO 920
910 LET ZZ=APER
920 IF ZZ=1 OR ZZ=2 OR ZZ=4 OR ZZ=5 OR ZZ=6 OR ZZ=8 OR ZZ=12 OR ZZ=13 THEN  LET ZX=15 ELSE  LET ZX=1
930 LET ZY=ZZ+16*ZX
940 GENPAT 2,147,ZY,ZY,ZY,ZY,ZY,ZY,ZY,17
950 RETURN
960 STOP
970 CLEAR
980 CLS
990 USER  SAVE "ETCHONE.BAS"
1000 RUN
```



```
NODDY PAGE 'INST'
*D P. *E *R
```

```
NODDY PAGE 'P'
      >>>>>>>>>**<<<<<<<<<
      COLOUR ETCH-A-SKETCH
      >>>>>>>>>**<<<<<<<<<
```

Instructions.

1) Use the joystick to pick the colours you wish to use.
2) Press fire. You now have four modes
   (1) Draw--Draws an ink line on paper
   (2) Erase-Draws paper on paper
   (3) Over--Changes ink dots to paper and paper to ink
   (4) Move--Leaves no line
3) Press fife at any time to goto 1)
4)(E) Will erase the picture
5)(f1)Will print out a hard copy to a DMX80 in Black and White

Press Return.

NB. The Joystick should be in port R

Danger Island by D. A. Mansell

```
5 SAVE "DANGER ISLAND"
10 REM DANGER ISLAND
20 REM A MINI-ADVENTURE FOR THE ORIC            (16 & 48K) BY S.W. LUCAS
30 REM ADAPTED FOR THE MTX 500 BY                    D.A. MANSELL
40 PRINT CHR$(27);"B1": PAPER 6: PLOD "INSTRUCTIONS"
45 LET AA=0: LET AB=0: LET AC=0: LET AZ=0: LET RF=0
50 DIM Q$(30,160),G$(30,50),V$(4,50),S(30,4),B(30,1),N$(30,30),M$(10,64),N(30)
51 LET BLANK$="                                                "   <- 49 SPACES
55 LET M$(1)="Even Tarzan couldnt carry any more! and I'm not that strong!"
56 LET M$(2)="I can't do that here.": LET M$(3)="I can't go in that direction."
57 LET M$(4)="I FELL OUT OF THE TREE AND BROKE MY    NECK!": LET M$(5)="I'VE POISONED MYSELF!"
58 LET M$(6)="IT HAS SAVAGELY ATTACKED AND TORN MY   THROAT OUT!"
60 LET A$="": LET P=1: GOSUB 11000: RESTORE 10000: GOSUB 12000
90 CLS
100 PRINT : PRINT "I am ";: PRINT Q$(P): LET A$=""
110 IF S(P,1)>0 THEN  LET A$="NORTH"
120 GOSUB 11000
130 IF S(P,2)>0 AND LEN (A$)>0 THEN  LET A$=A$+",SOUTH" ELSE  IF S(P,2)>0 THEN  LET A$="SOUTH"
140 IF S(P,3)>0 AND LEN (A$)>0 THEN  LET A$=A$+",EAST" ELSE  IF S(P,3)>0 THEN  LET A$="EAST"
150 IF S(P,4)>0 AND LEN (A$)>0 THEN  LET A$=A$+",WEST" ELSE  IF S(P,4)>0 THEN  LET A$="WEST"
155 IF A$="" THEN  LET A$="Nowhere at all."
160 IF A=8 THEN  PLOD "end": STOP
190 PRINT : PRINT "I can go :-";A$
```

```
200 LET E=0: FOR T=1 TO 20: LET PP=0: IF B(T,1)=P THEN  LET PP=1
210 IF PP=1 THEN  GOTO 230
220 NEXT : GOTO 250
230 IF E=0 AND G$(T)<>"" THEN  PRINT : PRINT "I can see :-"
240 PRINT G$(T): LET E=E+1: GOTO 220
250 PRINT : INPUT "What do I do now ";Z$
260 LET B$=LEFT$(Z$,2): LET C$=LEFT$(Z$,3)
271 IF Z$="GO IN" OR B$="IN" THEN  IF P=20 THEN  LET P=21: GOTO 90
272 IF Z$="GO OUT" OR B$="OU" THEN  IF P=21 THEN  LET P=20: GOTO 90
273 IF Z$="GO IN" OR C$="IN" THEN  CLS : PRINT M$(2): GOTO 100
274 IF Z$="GO OUT" OR B$="OU" THEN  CLS : PRINT M$(2): GOTO 100
280 IF B$="SC" THEN  GOSUB 11000: CLS : PRINT "YOU HAVE MADE #";A;" AND NEED #8": GOTO 100
290 IF B$="N" AND S(P,1)<>0 THEN  LET P=S(P,1): GOTO 90
300 IF B$="S" AND S(P,2)<>0 THEN  LET P=S(P,2): GOTO 90
310 IF B$="E" AND P=17 AND AB<>1 THEN  CLS : PRINT "The dog won't let me go anywhere!": GOTO 100
311 IF B$="E" AND S(P,3)<>0 THEN  LET P=S(P,3): GOTO 90
320 IF B$="W" AND S(P,4)<>0 THEN  LET P=S(P,4): GOTO 90
322 IF B$="N" OR B$="S" OR B$="W" OR B$="E" THEN  CLS : PRINT M$(3): GOTO 100
325 IF B$="FU" OR B$="PI" OR C$="BUG" THEN  GOSUB 15000
330 IF B$="GE" OR B$="TA" OR B$="GR" THEN  GOTO 13000
340 IF C$="DRO" OR B$="PU" THEN  GOTO 14000
350 IF C$="INV" THEN  GOTO 14700
360 IF C$="SWI" THEN  GOTO 17000
362 IF (C$="CLI" OR B$="UP") AND P=10 THEN  LET P=22: GOTO 90
364 IF (C$="CLI" OR C$="DOW") AND RF<>1 AND P=22 THEN  CLS : PRINT "HOW?": GOTO 100
365 IF (C$="CLI" OR C$="DOW") AND P=22 THEN  CLS : PRINT "I've climbed down the tree with the aid of the rope.": LET P=10: GOTO 100
368 IF C$="CLI" THEN  CLS : PRINT "There's nothing here to climb!": GOTO 100
370 IF C$="DOW" OR B$="UP" THEN  CLS : PRINT M$(2)
374 IF C$="JUM" AND P=22 THEN  LET DEAD=4: GOTO 17500
376 IF C$="JUM" THEN  CLS : PRINT "O.K.": PRINT "I don't get very far!": GOTO 100
380 IF C$="LOO" THEN  GOTO 90
385 IF C$="THR" THEN  GOTO 17600
390 IF C$="EAT" THEN  CLS : PRINT "I'm not very hungry thank you!": GOTO 100
391 IF C$="DRI" AND P=2 THEN  PRINT "O.K. it's a bit brackish - in fact I feel sick": PAUSE 1000: PRINT "Oh dear! I think it's pollu
ted": PAUSE 1000
392 IF C$="DRI" AND P=2 THEN  LET DEAD=5: GOTO 17500
393 IF C$="DRI" AND P>10 AND P<12 THEN  CLS : PRINT "A bit on the cold side but very refreshing!": GOTO 100
395 IF C$="DRI" THEN  CLS : PRINT "There's nothing here to drink!": GOTO 100
400 IF C$="HEL" THEN  PLOD "help": PAPER 6: GOTO 90
401 IF C$="EXA" THEN  CLS : PRINT " I can't see anything special!": GOTO 100
402 IF C$="QUI" THEN  CLS : STOP
405 IF C$="SEA" THEN  GOTO 18000
410 IF C$="DIG" AND P=23 THEN  GOTO 19000 ELSE  IF C$="DIG" THEN  CLS : PRINT M$(2): GOTO 100
415 IF C$="GIV" THEN  GOTO 19500
450 PRINT : PRINT "SORRY! I don't understand you.": PAUSE 1000: GOTO 90
460 STOP
2999 STOP
10000 DATA on a stony footpath with trees on either side. I can hear birds singing  merrily in the undergrowth and the sun is shinin
g brightly.,0,0,2,0
10010 DATA on the shores of a large lake. Thelake is shrouded in the early morning  mist and I can't see how far it        stretches
.,0,3,0,1
10020 DATA on a wide mountain track winding  slowly up the side of the hill.,2,4,0,0
10030 DATA at the top of a mountain. I can  see for miles...in the distance is a  small village and behind me a lake witha small i
sland.,3,0,5,6
10040 DATA by a small waterfall which is fed by a mountain stream. Large clouds of  spray are being thrown up and the sun  is creati
ng a rainbow.,0,10,0,4
10050 DATA on a narrow mountain track. It is very misty here making the stones      slippery and treacherous.,0,7,4,0
10060 DATA by a huge rockfall. The area is  covered with a litter of debris        including several large boulders.,6,8,0,0
10070 DATA by a cave entrance. The opening isdraped with moss and from within I can here the sound of dripping water.,7,0,9,0
```

```
10080 DATA in a vast cavern. Stalagtites hangfrom the roof of the cave and water    drips incessantly from their pointed   tips.,0,0
,11,8
10090 DATA by the waterfall. There is a tall cypress tree growing just by the path  and a cuckoo is sitting in it watching me contem
platively.,5,11,0,0
10100 DATA standing in the waterfall and it'sbloody cold! Through the freezing sprayI can see a cave entrance.,10,0,12,9
10110 DATA at the other side of the waterfall. A small stream trickles by my feet.  I think I can hear a cuckoo!,0,13,0,11
10120 DATA on a footpath which is wending itsway gently down the side of the       mountain.,12,14,0,0
10130 DATA at the bottom of the mountain.    There's a small village here basking inthe midday sun. All is quiet except fora dog bar
king nearby.,13,16,15,17
10140 DATA on the main road through the      village. There is nobody about - must  be siesta time.,18,0,0,14
10150 DATA by the village blacksmiths. It is closed for the day. A note pinned to   the door says 'BEWARE OF THE DOG'.,14,0,0,0
10160 DATA by some small cottages. There's a little old lady here dozing in a       rocking chair.,0,0,14,0
10170 DATA on a road bridge over a small     cutting. It's a bit smelly - I think itmust be a sewer.,0,15,19,0
10180 DATA back on the main road.,20,0,0,18
10190 DATA by a shop. Above the door are     three large brass balls.,0,19,0,0
10200 DATA inside the pawnbrokers shop.      The pawnbroker is sitting behind an oldoak counter.,0,0,0,0
10210 DATA at the top of a very tall cypress tree. I'm glad I don't suffer from     vertigo. The cuckoo has flown away.,0,0,0,0
10220 DATA on a small sandy island in the    middle of the lake. The only vegatationis a few tufts of grass.,0,0,0,0
10230 DATA an old empty beer can,12,some algae,11,a small boy,4
10240 DATA ,7,a tiny #GOLD# nugget,9,a #HORSESHOE#,16,a #TEAPOT#,17
10250 DATA ,23,A #WEDDING RING#,23
10260 DATA A VALUABLE CHINESE #KITE#,22
10280 DATA pebbles,11,stones,7,a squirrel,22,nuts,1,a shovel,23
10290 DATA a hammer,16,a dog,17,a cheese sandwich,1,a large boulder,8,a coil of rope,18
10300 DATA BEER,1,CAN,1,ALGAE,2,BOY,3,SWISS,3,BADGE,4,ROLLS,4,GOLD,5,NUGGET,5
10310 DATA HORSESHOE,6,SHOE,6,TEAPOT,7,SILVER,8,NEEDLE,8,WEDDING,9,RING,9
10320 DATA CHINESE,10,KITE,10,PEBBLES,11,STONES,12,SQUIRREL,13,NUTS,14,SHOVEL
10330 DATA 15,HAMMER,16,DOG,17,SANDWICH,18,BOULDER,19,COIL,20,ROPE,20
11000 LET A=0
11030 IF B(3,1)=21 THEN  LET A=A+1
11040 IF B(4,1)=21 THEN  LET A=A+1
11050 IF B(5,1)=21 THEN  LET A=A+1
11060 IF B(6,1)=21 THEN  LET A=A+1
11070 IF B(7,1)=21 THEN  LET A=A+1
11072 IF B(8,1)=21 THEN  LET A=A+1
11075 IF B(9,1)=21 THEN  LET A=A+1
11080 IF B(10,1)=21 THEN  LET A=A+1
11090 RETURN
12000 FOR H=1 TO 23: READ Q$(H): FOR D=1 TO 4: READ S(H,D): NEXT D: NEXT H
12010 FOR H=1 TO 20: READ G$(H),B(H,1): NEXT
12040 FOR H=1 TO 29: READ N$(H),N(H): NEXT
12050 RETURN
12999 REM GET ROUTINE
13000 GOSUB 13500: IF L=1 THEN  GOTO 13020
13010 CLS : PRINT "I CAN'T SEE A ";L$: GOTO 100
13020 LET E=0: FOR H=1 TO 20: IF B(H,1)=P AND B(N(R),1)=P THEN  LET E=1
13030 NEXT
13040 IF E=0 THEN  GOTO 90
13041 IF R=23 THEN  LET AA=AA+1
13042 IF (R=4 OR R=5) AND AC()1 THEN  CLS : PRINT "I CAN'T DO THAT DUMMY !": GOTO 100
13043 IF R=25 THEN  CLS : LET DEAD=6: GOTO 17500
13045 IF R=28 OR R=29 THEN  LET RF=RF+1
13450 LET E=0: FOR D=1 TO 3: IF ASC(V$(D))<33 THEN  LET V$(D)=G$(N(R)): LET E=1: LET D=5
13460 NEXT
13462 IF E=0 THEN  CLS : PRINT M$(1): GOTO 100
13480 LET B(N(R),1)=0: GOTO 90
13499 REM ROUTINE TO MAKE SENSE OF RESPONSE
13500 LET L$="": FOR H=1 TO LEN (Z$)
13510 IF MID$(Z$,H,1)=" " THEN  LET L$=RIGHT$(Z$,(LEN (Z$)-H)): LET H=H+40
```

```
13520 NEXT
13530 LET R=0
13540 LET L=0: IF LEN (L$)<2 THEN  RETURN
13550 FOR H=1 TO 29: IF LEFT$(N$(H),LEN (L$))=L$ THEN  LET L=1: LET R=H
13560 NEXT
13570 RETURN
13999 REM DROP ROUTINE
14000 GOSUB 13500
14010 IF L=1 THEN  GOTO 14030
14020 CLS : PRINT M(3): GOTO 100
14030 LET E=0
14040 FOR D=1 TO 3: IF V$(D)=G$(N(R)) THEN  LET V$(D)=BLANK$: LET E=1
14050 NEXT
14060 IF E=1 THEN  GOTO 14080
14070 CLS : PRINT ;"I'VE NOT GOT IT YOU IDIOT!!!!": GOTO 100
14080 LET B(N(R),1)=P: CLS
14090 IF R=23 THEN  LET AA=0
14095 IF R=28 OR R=29 THEN  LET RF=0
14100 IF P=17 AND R=26 THEN  LET AB=1: PRINT "IT IS BUSY EATING. I THINK I CAN ESCAPE"
14500 GOTO 100
14699 REM INVENTORY ROUTINE
14700 CLS : PRINT ;"I HAVE :- "
14705 LET F=0
14710 FOR H=1 TO 3: IF ASC(V$(H))<>32 THEN  PRINT ;V$(H): LET F=1
14720 NEXT
14730 IF F=0 THEN  PRINT "BUGGER ALL"
14735 PRINT
14740 GOTO 100
15000 CLS : PRINT : PRINT : PRINT : PRINT "HOW DARE YOU TALK TO ME LIKE THAT?"
15010 PRINT : PRINT : PRINT "WHAT HAVE YOU GOT TO SAY TO  ME NOW?"
15020 INPUT Z$: IF Z$<>"SORRY" THEN  GOTO 15020 ELSE  RETURN
16999 REM SWIM ROUTINE
17000 CLS : IF P=2 OR P=23 THEN  GOTO 17020
17010 PRINT "I CAN'T SWIM HERE! MORON!": GOTO 100
17020 PRINT "O.K."
17030 IF P=2 THEN  LET P=23 ELSE  IF P=23 THEN  LET P=2
17040 GOTO 100
17499 REM DEATH ROUTINE
17500 CLS : PAPER 7: INK 1: PRINT M$(DEAD)
17510 PRINT : PRINT : PRINT "I AM DEAD"
17520 PRINT : PRINT : PRINT "DO YOU WANT TO PLAY AGAIN?"
17530 LET A$=INKEY$: IF A$<>"Y" AND A$<>"N" THEN  GOTO 17530
17540 IF A$="N" THEN  STOP  ELSE  INK 15: CLS : GOTO 60
17599 REM THROW ROUTINE
17600 CLS : GOSUB 13500
17610 IF L=1 THEN  GOTO 17630
17620 PRINT : PRINT "I CAN'T  SEE A ";L$: GOTO 100
17630 LET E=0
17640 FOR D=1 TO 3: IF V$(D)=G$(N(R)) THEN  LET V$(D)="": LET E=1
17650 NEXT
17660 IF E=1 THEN  GOTO 17680
17670 PRINT "I HAVEN'T GOT IT !!! DUMMY!!": GOTO 100
17680 IF P<23 THEN  LET B(N(R),1)=P+1 ELSE  LET B(N(R),1)=P
17690 PRINT "O.K."
17800 GOTO 100
17999 REM SEARCH ROUTINE
18000 CLS : IF P=7 THEN  GOTO 18010 ELSE  PRINT "I CAN'T FIND ANYTHING": GOTO 100
18010 LET G$(4)="A #BADGE# FROM A.ROLLS  ROYCE CAR"
18015 IF AZ<>0 THEN  PRINT "THERE'S NOTHING ELSE HERE!": GOTO 100
18020 PRINT "I HAVE FOUND SOMETHING!!!": LET AZ=1: GOTO 100
```

```
19000 CLS : IF AA()1 THEN  PRINT "I DON'T HAVE ANYTHING TO DIG WITH": GOTO 100
19010 PRINT "I HAVE FOUND SOMETHING": LET G$(8)="A #SILVER# NEEDLE": LET AA=2: GOTO 100
19500 CLS : GOSUB 13500: IF L=1 THEN  GOTO 19515
19510 GOTO 100
19515 IF P=4 AND R=22 THEN  PRINT "HE TAKES A FEW AND OFFERS ME SOMETHING"
19520 IF P=4 AND R=22 THEN  PRINT "IN EXCHANGE. HE THEN RUNS OFF SINGING"
19530 IF P=4 AND R=22 THEN  LET G$(3)="A #SWISS ARMY KNIFE#": LET N$(4)="KNIFE": LET AC=1: GOTO 100
19540 IF P=22 AND R=22 THEN  PRINT "THE SQUIRREL DOESN'T LIKE THEM!": GOTO 100
19550 PRINT " I'M NOT QUITE SURE I  KNOW WHAT YOU MEAN": GOTO 100
```

NODDY PAGES FOR DANGER ISLAND

INSTRUCTIONS

```
*D P1. *P *P
*P *P *P *P
*P *P *P *P
*P *P *P *P
*R
```

P1

You are a student fallen on hard times. You have gone on a long needed holiday, but have run out of money. Your task is to find sufficient items of value and take them to the Pawnbrokers to exchange for cash. You need £8 for your fare home and each item is worth £1.

GOOD LUCK!

help

```
*D HELP.  *E  *R
```

HELP

I understand :-

N S E W	TAKE
[INV]ENTORY	DROP
[SC]ORE	PUT
GRAB	OUT
SWIM	IN
JUMP	THROW
DOWN	DRINK
UP	HELP
CLIMB	DIG
GO IN	GET
GO OUT	GIVE
LOOK	EXAMINE
EAT	SEARCH
QUIT	

```
end

  *D END. *R
```

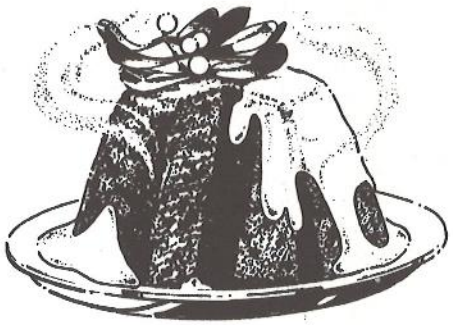
END

Well done! You have found all of the eight items of treasure and exchanged them for your fare home. Next time I suggest that you go somewhere less dangerous for your holiday!

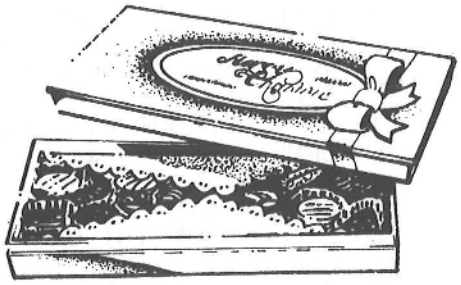
GOODBYE

The Towers of Hanoi by E. Roy

```
100 REM ********************************
110 REM ****  THE TOWERS OF HANOI  ****
120 REM ****   MEMOTECH 500,512.   ****
130 REM ****   by E. Roy. Jan.85   ****
140 REM ********************************
150 REM
160 DIM DISC$(8,17),DP(3),POST(3,8): LET MOVES=45000
170 LET TITLE$="--- THE TOWERS OF HANOI ---"
180 VS 5: CLS : CSR 6,0: PRINT TITLE$
190 POKE 64145,132
200 GOSUB 1690: GOSUB 1890
210 REM-----------------------------------
220 REM  Main Program Loop.
230 REM-----------------------------------
240 VS 4: COLOUR 0,1: COLOUR 2,1: COLOUR 4,1: CLS
250 CRVS 4,1,1,0,31,24,32
260 VS 4: CLS
270 COLOUR 0,5: COLOUR 1,15
280 CSR 0,0: PRINT BLK$+BLK$+BLK$;
290 CSR 0,21: PRINT BLK$+BLK$+BLK$;
300 CSR 2,1: PRINT TITLE$
310 GOSUB 1420
320 CLOCK "000000"
330 REM
340 COLOUR 0,5: COLOUR 1,15: CSR 0,22: PRINT BLK$(1,20)
350 CSR 2,22: PRINT "FROM.."
360 GOSUB 1220
370 IF K$>"0" AND K$<"4" THEN  LET FP=VAL(K$) ELSE  GOTO 360
380 SOUND 1,194,10: PAUSE 100: SOUND 1,0,0
390 REM
400 CSR 8,22: PRINT K$+" TO.."
410 GOSUB 1220
420 IF K$>"0" AND K$<"4" THEN  LET TP=VAL(K$) ELSE  GOTO 410
430 CSR 14,22: PRINT K$
440 SOUND 1,194,10: PAUSE 100: SOUND 1,0,0
450 REM
460 IF TP=FP THEN  GOSUB 1320: GOTO 340
470 IF DP(FP)=0 THEN  GOSUB 1320: GOTO 340
480 IF DP(FP)>0 AND DP(TP)>0 THEN  IF POST(FP,DP(FP))>POST(TP,DP(TP)) THEN  GOSU
B 1320: GOTO 340
490 GOSUB 760
500 IF DP(1)>0 OR DP(2)>0 AND DP(3)>0 THEN  GOTO 340
510 REM-----------------------------------
520 REM  Finished, Play Again, Replay.
530 REM-----------------------------------
540 LET T$=TIME$
550 COLOUR 0,5: COLOUR 1,15
560 CSR 0,1: PRINT BLK$: CSR 11,1: PRINT "FINISHED"
570 CSR 1,22: PRINT "TIME TAKEN FOR";N;" DISCS"
580 CSR 25,22: PRINT T$(3,2)+":"+T$(5,2)
590 COLOUR 0,15: COLOUR 1,5
600 FOR L=3 TO 6: CSR 0,L: PRINT BLK$: NEXT L
610 CSR 0,3
620 PRINT " Number of Moves.....";COUNT
630 PRINT " Illegal Moves.....";ILLEGAL
640 PRINT " Minimum Number of Moves"
650 PRINT " for";N;" Discs is.....";2^N-1
660 CSR 0,7: PRINT BLK$: PAUSE 1000
670 CSR 0,7: PRINT " Another Game  Y/N, Replay  ";
680 LET K$=INKEY$: IF K$="" THEN  GOTO 680
690 IF K$="Y" THEN  GOTO 260
```



```
700 IF K$="R" THEN  GOSUB 1040: COLOUR 0,15: COLOUR 1,5: GOTO 660
710 IF K$="M" THEN  POKE 64145,160: STOP
720 GOTO 680
730 REM-----------------------------------------
740 REM  Move Disc From Post-To Post.
750 REM-----------------------------------------
760 LET BYTE=FP+(TP*16)
770 POKE (MOVES+COUNT),BYTE: LET COUNT=COUNT+1
780 LET DN=POST(FP,DP(FP))
790 LET DX=(FP-1)*10: LET P=0
800 FOR DY=19-DP(FP) TO 9 STEP -1
810 SOUND 1,DY*6,10
820 CSR DX,DY: PRINT DISC$(DN)
830 IF P>0 THEN  CSR DX,DY+1: PRINT POST$
840 LET P=P+1: NEXT DY
850 SOUND 1,0,0
860 CSR DX,DY+1: PRINT POST$
870 LET F=(FP-1)*10: LET T=(TP-1)*10: LET S=SGN(TP-FP)
880 FOR DX=F TO T STEP S
890 CSR DX,8: PRINT DISC$(DN)
900 NEXT DX
910 CSR T,8: PRINT BLK$(1,10)
920 FOR DY=9 TO 18-DP(TP)
930 SOUND 1,DY*6,10
940 CSR T,DY: PRINT DISC$(DN)
950 IF DY>9 THEN  CSR T,DY-1: PRINT POST$
960 NEXT DY
970 LET DP(FP)=DP(FP)-1
980 LET DP(TP)=DP(TP)+1
990 LET POST(TP,DP(TP))=DN
1000 SOUND 1,0,0: RETURN
1010 REM-----------------------------------------
1020 REM  Replay Moves Made.
1030 REM-----------------------------------------
1040 CSR 0,22: FOR L=1 TO 23: LET S$(L)=SPK$: NEXT
1050 LET TF=FP: LET TT=TP
1060 COLOUR 0,5: COLOUR 1,15
1070 CSR 0,22: PRINT BLK$(1,23)
1080 CSR 12,22: PRINT "REPLAY"
1090 GOSUB 1500
1100 FOR R=MOVES TO (MOVES+COUNT)-1
1110 LET BYTE=PEEK(R)
1120 LET FP=MOD(BYTE,16)
1130 LET TP=INT(BYTE/16)
1140 GOSUB 780
1150 NEXT R
1160 LET FP=TF: LET TP=TT
1170 COLOUR 0,5: COLOUR 1,15: CSR 0,22: PRINT S$
1180 RETURN
1190 REM-----------------------------------------
1200 REM  Update time, Get Key Pressed.
1210 REM-----------------------------------------
1220 LET K$=""
1230 LET T$=TIME$: CSR 25,22: PRINT T$(3,2)+":"+T$(5,2)
1240 IF INKEY$="" THEN  GOTO 1230
1250 LET K$=INKEY$
1260 IF INKEY$<>"" THEN  GOTO 1260
1270 IF K$="R" AND COUNT>0 THEN  GOSUB 1040  GOTO 1220
1280 RETURN
1290 REM-----------------------------------------
1300 REM  Illegal Move.
1310 REM-----------------------------------------
1320 LET ILLEGAL=ILLEGAL+1
1330 CSR 0,22: PRINT BLK$(1,20): ATTR 1,1
```



```
1340 FOR L=1 TO 20: SOUND 1,88,10
1350 CSR 12,22: PRINT "ILLEGAL": PAUSE 120
1360 SOUND 1,212,10
1370 NEXT L: ATTR 1,0: SOUND 1,0,0
1380 RETURN
1390 REM------------------------------------------
1400 REM  Input Number of Discs.
1410 REM------------------------------------------
1420 CSR 1,22: PRINT "ENTER NUMBER OF DISCS  2-8 ";
1430 LET K$=INKEY$: IF K$="" THEN  GOTO 1430
1440 IF K$<"2" OR K$>"8" THEN  GOTO 1430
1450 PRINT K$: PAUSE 500: CSR 0,22: PRINT BLK$
1460 LET N=VAL(K$): LET ILLEGAL=0: LET COUNT=0
1470 REM------------------------------------------
1480 REM  Display Posts, Discs.
1490 REM------------------------------------------
1500 LET DP(1)=N: LET DP(2)=0: LET DP(3)=0
1510 CSR 0,19: COLOUR 0,15: COLOUR 1,5: PRINT BASE$
1520 FOR L=18 TO 9 STEP -1
1530 CSR 0,L: PRINT POST$+POST$+POST$
1540 NEXT L
1550 LET D=N
1560 FOR L=18 TO 19-N STEP -1
1570 CSR 0,L: PRINT DISC$(D)
1580 SOUND 1,L*3,10: PAUSE 100
1590 LET D=D-1: NEXT L
1600 SOUND 1,0,0
1610 LET D=N
1620 FOR L=1 TO N
1630 LET POST(1,L)=D
1640 LET D=D-1: NEXT L
1650 RETURN
1660 REM------------------------------------------
1670 REM  Init. Characters, Strings.
1680 REM------------------------------------------
1690 GENPAT 0,64,255,255,255,255,255,255,255,255
1700 GENPAT 0,91,15,15,15,15,15,15,15,15
1710 GENPAT 0,93,240,240,240,240,240,240,240,240
1720 GENPAT 0,95,0,0,0,0,0,0,0,0
1730 LET COL$=CHR$(16)+CHR$(0)+CHR$(1)+CHR$(16)+CHR$(1)
1740 LET POST$=COL$+CHR$(15)+"_____@_____"
1750 LET DISC$(1)=COL$+CHR$(06)+"____[@]____"
1760 LET DISC$(2)=COL$+CHR$(10)+"____@@@____"
1770 LET DISC$(3)=COL$+CHR$(02)+"___[@@@]___"
1780 LET DISC$(4)=COL$+CHR$(04)+"___@@@@@___"
1790 LET DISC$(5)=COL$+CHR$(12)+"__[@@@@@]__"
1800 LET DISC$(6)=COL$+CHR$(14)+"__@@@@@@@__"
1810 LET DISC$(7)=COL$+CHR$(13)+"_[@@@@@@@]_"
1820 LET DISC$(8)=COL$+CHR$(07)+"_@@@@@@@@@_"
1830 LET BLK$="_____________________________";: REM  31 Chars.
1840 LET BASE$="___No.1_____No.2_____No.3____"
1850 RETURN
1860 REM------------------------------------------
1870 REM  Instructions  Y / N.
1880 REM------------------------------------------
1890 CSR 8,8: PRINT "DO YOU WISH INSTRUCTIONS"
1900 CSR 9,10: PRINT "PRESS KEY - Y or N. ";
1910 LET K$=INKEY$
1920 IF K$="" THEN  GOTO 1910
1930 IF K$="Y" THEN  PRINT "YES": PAUSE 1000: GOSUB 2000: GOTO 1960
1940 IF K$="N" THEN  PRINT "NO": PAUSE 1000: GOTO 1960
1950 GOTO 1910
1960 CLS : RETURN
1970 REM------------------------------------------
1980 REM  Display Instructions.
```

```
1990 REM-----------------------------------------
2000 CSR 2,8: PRINT BLK$: CSR 2,10: PRINT BLK$: CSR 1,2
2010 PRINT "The  Towers of Hanoi  Puzzle  involves","moving a number of discs  f
rom one Post","to  another,  in  the  least  Possible","number of moves."
2020 PRINT : PRINT " Rules of Play.....A larSe disc cannot","be  Placed on a  s
mall disc,  and the","FROM & TO  Posts must not be the same."
2030 PRINT : PRINT " You will be timed during Play, so any","illegal  move  or
hesitation  on your","Part will result in an increase in the","overall time."
2040 PRINT : PRINT " Use either the keyboard or keypad keys","1, 2 & 3 to move d
iscs. Press key 'R'","at any time to REPLAY  all moves from","the start."
2050 PRINT : PRINT "        PRESS ANY KEY TO PLAY."
2060 LET K$=INKEY$: IF K$="" THEN  GOTO 2060
2070 RETURN
2080 REM-----------------------------------------
2090 REM  Save to Tape.
2100 REM-----------------------------------------
2110 SAVE "HANOI": RUN
```

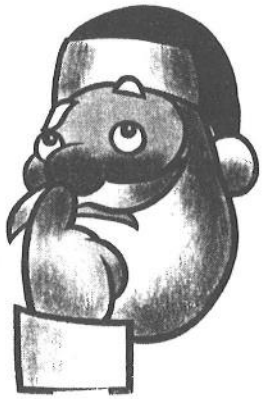
Wallpaper For The Mind by B.L. Houghton

"Wallpaper for the Mind" is the delightful subtitle of an article by A.K. Dewdney in 'Scientific American', September 1986, in the series 'Computer Recreations' which normally turn out to be very complex 'recreations', but this time describes a set of graphic algorithms ranging from 'easy' to 'childishly easy'.

The functions used have sort of family resemblance to fractals, except that whereas the latter are iterative functions applied to complex numbers, these all involve ordinary real numbers, and are corresponding simpler.

The following short BASIC version of one of Dr. Dewdney's algorithms shows just how simple they mostly are. It's not very fast, but it is accessible to everyone, and may stimulate readers to get a reprint of the article. For results that you can actually sit down and watch develop, it really should be written in something like PASCAL or FORTRAN. CP/M users should put the display on a Type 1 screen, as the resolution of the 80-column screen is not up to the complexity and detail of the patterns. The outputs are really quite extraordinary, with an uncanny resemblance to plant and animal structures as seen under a microscope.

```
10 INPUT"ITERATIONS: ";NUM:LET  NUM
     =ABS(NUM):IF  NUM<10000   THEN
     LET NUM=NUM+10000
20 INPUT"A,B,C: ";A,B,C
30 INPUT"SCALE:  ";SC:IF SC=0  THEN
     LET SC=1
40 A=A*SC:B=B*SC:C=C*SC
50 VS 4:COLOUR 0,1:COLOUR 2,1:
     COLOUR 4,1:COLOUR 1,15:
     COLOUR 3,15:CLS
60 LET X=0:LET Y=0
70 FOR I=1 TO NUM
80 PLOT X+128,Y+95
90 XX=SGN(X)*SQR(ABS(B*X-C))
100 YY=A-X:X=XX:Y=YY
110 NEXT
```



(F) CP/M-80 SOFTWARE

PRODUCT	BY	TYPE	VERSION	RRP (£)
ACCESS MANAGER	Digital Research	Programming tool	1.1	270
ANIMATOR (CIS)	MicroFocus	Programming tool	1.0	225
ANIMATOR LVII	**MicroFocus**	**Programming tool**	**2.1**	**325**
ASCOM	DMA	Communications	2.29	170
ASSEMBLER + TOOLS	Digital Research	Programming tool	1.0	185
BACK-UP	Xitan	Utility	1.1	75
BASIC COMPILER	Microsoft	Language	5.30	345
BASIC INTERP.	Microsoft	Language	5.21	295
BSTAM	Byrom Software	Communications	4.6	150
BSTMS	Byrom Software	Communications	1.2	150
CALCSTAR	MicroPro	Financial planng	1.45	99
CBASIC	Digital Research	Language	2.8	130
CBASIC COMPILER	Digital Research	Language	2.0	445
CIS COBOL	**MicroFocus**	**Language**	**4.5.5**	**350**
COBOL	Microsoft	Language	4.68	560
CP/M	Digital Research	Operating System	2.2	135
CP/M PLUS	Digital Research	Operating System	3.0	315
DATASTAR	MicroPro	Database	1.41	175
DATAPLOT+	Grafox	Graphics		195
DBASE II	Ashton-Tate	Database	2.43*	395
DISPLAY MANAGER	Digital Research	Programming tool	1.1	350
DR DRAW	Digital Research	Graphics Req GSX	1.0	265
DR GRAPH	Digital Research	Graphics Req GSX	1.0	265
ECO-C	EcoSoft	Language	3.30	150
FILESHARE (CIS)	**MicroFocus**	**Utility**	**3.0.6**	**225**
FILESTAR	MicroSec	Disk reformatter	10	110
FORMS 2	**MicroFocus**	**Utility**	**1.3.5**	**160**
FORTRAN	Microsoft	Language	3.44	400
GSX	Digital Research	Graphics	1.1	70
INFOSTAR	MicroPro	Database/Reportr	1.02	245
LEVEL II COBOL	**MicroFocus**	**Language**	**2.1.3**	**550**
M2CBASIC	Digital Research	Translator	1.4	150
MACRO (M80)	Microsoft	Programming tool	3.44	195
MAILMERGE	MicroPro	Word Processing	3.30	145
MICROSTAT	Ecosoft	Statistics	4.1.11	329
MP/M II	Digital Research	Operating System	2.1	400
MULTIPLAN	Microsoft	Financial Planng	1.06	159
PASCAL/MT	Digital Research	Language	5.6.1	295
PERTMASTER	Abtex	Critical Path	4.6	650
PL/I	Digital Research	Language	1.4	495
PROFESSIONAL OPT.	MicroPro	Word proc add-on	3.30	245
PRO-FORTRAN	Prospero	Language	1.25	220
PRO-PASCAL	Prospero	Language	2.18	220
RM-COBOL Compiler	Ryan McFarland	Language	2.0C	625
RM-COBOL RTS	Ryan McFarland	Language Runtime	2.0C	210
SDISK	Xitan	Silicon disk	2.0	200
SENSIBLE SOLN.s/u	O'Hanlon	Database	2.0CA	P.O.R.
SENSIBLE SOLN.m/u	O'Hanlon	Database	2.0CB	P.O.R.
SENSIBLE RTS s/u	O'Hanlon	Database	2.0CA	P.O.R.
SENSIBLE RTS m/u	O'Hanlon	Database	2.0CB	P.O.R.
SORT	Microsoft	Sort	1.05	155
SOURCEWRITER	**MicroFocus**	**Programming tool**	**1.4**	**700**
SPELLSTAR	MicroPro	Word Processing	3.30	145
SPP (for Pascal)	Digital Research	Programming tool	5.5	195
STARINDEX	MicroPro	Word Processing	1.01	110
SUPERCALC	Sorcim/IUS	Financial planng	1.12H	126
SUPERCALC2	Sorcim/IUS	Financial Planng	1.0	195
SUPERSORT	MicroPro	Sort Utility	1.60	145
WORDMASTER	MicroPro	Word Processing	1.07	90
WORDSTAR	MicroPro	Word Processing	3.30	295
WS PROFESSIONAL	**MicroPro**	**Word Processing**	**3.30**	**399**
WP WORKSHOP(W/S)	MAC	Training	3.3	75
WP WORKSHOP(M/M)	MAC	Training	3.3	75
XASM04 (6804)	Avocet	Cross Assembler	1.01	225
XASM05 (6805)	Avocet	Cross Assembler	1.07	180
XASM09 (6809)	Avocet	Cross Assembler	1.09	180
XASM18 (1802/05)	Avocet	Cross Assembler	1.53	180
XASM48 (8048/41)	Avocet	Cross Assembler	1.64	180
XASM51 (8051)	Avocet	Cross Assembler	1.10	180
XASM65 (6502)	Avocet	Cross Assembler	2.03	180
XASM68 (6800/01)	Avocet	Cross Assembler	2.13	180
XASMF8 (F8/3870)	Avocet	Cross Assembler	1.05	180
XASMZ8 (Z8)	Avocet	Cross Assembler	1.07	180
XASMZ80 (Z80)	Avocet	Cross Assembler	1.03	225
XASM85 (8085/Z80)	Avocet	Cross Assembler	1.04	225
XASM400 (COP400)	Avocet	Cross Assembler	1.04	180
XASM75 (NEC7500)	Avocet	Cross Assembler	1.19	450
XMAC68K (68000)	Avocet	Cross Assembler	5.0	680
XBASIC	Xitan	Language	4.63	185

Issue 3 & 4 special

memopad vol 0011

Hardware

TYPE CODES

U UTILITY
L LANGUAGE
B BUSINESS
E EDUCATIONAL
G GAME
J JOYSTICK COMPATIBLE

PRICE SUBJECT TO CHANGE WITHOUT NOTICE AND ALSO SUBJECT TO POST AND PACKING.

STOCK NO.	MANUFACT. REFERENCE	TITLE	TYPE	RETAIL PRICE INCL.VAT	MEMBERS PRICE INCL.VAT
6000	MTX 500	NO LONGER AVAILABLE			
6001	MTX 512	64K COMPUTER		79.95	69.95
6003		MEMORY 32K UPGRADE		39.95	37.95
6004		MEMORY 64K UPGRADE		49.95	47.45
6005		MEMORY 128K UPGRADE		79.95	75.95
6011		NEWWORD ON ROM		39.95	37.95
6012		PASCAL ON ROM		39.95	37.95
6013		RS232 BOARD		39.95	37.95
6021	SDX PACK 1	500K DRIVE/INTERFACE/80 COL. BOARD/CPM		399.00	359.10
6022	SDX PACK 2	*** DITTO *** BUT WITH 1MB DRIVE		449.00	404.10
6023	SDX PACK 3	AS PACK 1 PLUS MTX 512		499.00	449.10
6024	SDX PACK 4	*** DITTO *** BUT WITH 1MB DRIVE		549.00	494.10
6031	SDX	2ND DRIVE 500K UPGRADE		189.95	170.96
6032	SDX	2ND DRIVE 1MB UPGRADE		229.95	206.96
6036		80 COL PCB CP/M NW+SC		199.95	184.95
6037		80 COL PCB RS232 KIT		27.00	27.00
6042	SDX	SDX 500K DRIVE & INTERFACE		249.95	224.96
6043	SDX	SDX 1 MB DRIVE & INTERFACE		299.95	269.96
6050	FDX	TWIN 500K (REQUIRES RS232 BOARD)		649.00	584.10
6051	FDX	TWIN 1 MB (REQUIRES RS232 BOARD)		825.00	742.50
6055		SILICON DISC 1MB		179.00	161.10
6056		SILICON DISC 2MB		338.00	304.20
6057		SILICON DISC 4MB		636.00	572.40
		FACTORY FITTED FOR ABOVE 3 ITEMS		25.50	23.00
6065	FDX	2ND DRIVE 500K UPGRADE		159.00	143.10
6066	FDX	2ND DRIVE 1MB UPGRADE		189.00	170.10
6071	SDX 3.5 PACK 1	1 MB 3.5" DRIVE PLUS INTERFACE		166.00	149.40
6072	SDX 3.5 PACK 2	******* DITTO ******* PLUS SILICON DISC, 80 COLUMN CARD AND CPM		264.00	237.60
6073		80 COLUMN CARD, CPM AND SILICON DISC		129.00	116.10
		ADDON FOR SDX 3.5			
6080	DMX 80	PRINTER		192.00	172.80
6085		PRINTER CABLE 2 METRE		14.95	13.46
6090	KX M110	DMX 80 PRINTER RIBBON		9.95	8.96
6095		FDX TO MTX CONNECTOR		25.30	22.77
6097	TM110 V23	TANDATA MODEM FOR MEMOTECH 1200/75 BAUD		119.95	107.96
6100		DUST COVER FOR MTX		5.99	4.95
6105	DS QD	5.25" BLANK DISCS (BOX OF 10)		27.95	19.95
6110		5.25" DISC CASES (HOLDS 10 DISCS)		2.95	2.65
6120		FLOPPICLEAN KIT FOR 5.25"		19.10	17.20
6125		ANTISTATIC WIPES (PACK OF 10)		1.67	1.50

Manuals & Advertising

6501		CRIB CARD		2.95	2.12
6502		MTX ROM LISTING		45.00	32.40
6503		SDX CONTROLLER LISTING		25.00	18.75
6504		ROM CALLS INFORMATION SHEET		0.50	0.50
6505		RST10 CALLS INFORMATION SHEET		0.50	0.50
6506		INTERRUPTS INFORMATION SHEET		0.80	0.80
6507		MTX SERVICE MANUAL		9.95	9.95
6508		NEW USER MANUAL FOR MTX		8.95	8.95
6509		VDP MANUAL		7.95	7.95
6510		DDT MANUAL		2.50	2.50
		ADVERTISING IN THE MEMOPAD			
		==========			
6950		SMALL		5.00	2.50
6951		1/8 PAGE		27.50	27.50
6952		1/4 PAGE		45.00	45.00
6953		1/2 PAGE		80.00	80.00
6954		FULL PAGE		125.00	125.00

memopad
vol 0011

Issue 3 & 4
special

Software

TYPE CODES

U UTILITY
L LANGUAGE
B BUSINESS
E EDUCATIONAL
G GAME
J JOYSTICK COMPATIBLE

PRICE SUBJECT TO CHANGE WITHOUT NOTICE AND ALSO SUBJECT TO POST AND PACKING.

STOCK NO.	MANUFACT. REFERENCE	TITLE	TYPE	RETAIL PRICE INCL.VAT	MEMBERS PRICE INCL.VAT
7000	00106	26X26 SPREAD SHEET	U	8.83	7.95
7001	00057	3D TACHYON FIGHTER	G+J	7.72	6.95
7003	00062	ADVENTURE QUEST	G	9.95	8.96
7004	00033	AGROVATOR	G	6.61	5.95
7009	00071	ALICE IN WONDERLAND	G	7.72	6.95
7011	00141	ASSEM LANG COURSE	E	9.94	8.95
7012	00008	ASTROMILON	G+J	7.72	6.95
7013	00047	ASTROPAC	G+J	7.72	6.95
7014		ATTACK OF KILLER TOMATOES	G	8.83	7.95
7033	00058	BACKGAMMON	G	8.83	7.95
7034	00041	BASIC BUSINESS	B	7.72	6.95
7035	00043	BLOBBO	G+J	7.72	6.95
7036	00073	BOUNCING BILL	G+J	6.61	5.95
7037	00074	BRIDGE	G	7.72	6.95
7058	00077	CANVAS	U	8.83	7.95
7059	00085	CAVES OF ORB	G	6.61	5.95
7060		CAVES OF DOOM	G+J	6.61	5.95
7061	00094	CHAMBEROIDS	G+J	7.72	6.95
7062	00059	CHESS	G	9.94	8.95
7063	00053	COBRA	G	7.72	6.95
7064	00025	COLOSSAL ADVENTURE	G	9.95	8.96
7065	00098	COMBAT	G	4.39	3.95
7067	00046	CONT RAIDERS	G+J	7.72	6.95
7068	00099	CRIBBAGE (TEMPORARY NOT AVAILABLE)	G	4.39	3.95
7069	00110	CRYSTAL	G	7.72	6.95
7089	00050	DEN.GOES BANANAS	G+J	3.83	3.45
7090	00011	DENNIS & CHICKEN	G+J	3.83	3.45
7091	00103	DENNIS AND CIRCUS	G+J	3.83	3.45
7092		DISASM	U	8.83	7.95
7093	00068	DOODLEBUG	G+J	6.61	5.95
7094	00108	DOWNSTREAM DANGER	G+J	7.72	6.95
7095	00096	DR. FRANKIE	G+J	6.61	5.95
7096	00056	DRAUGHTS	G	7.72	6.95
7097	00111	DRIVE THE CEE 5	G+J	7.72	6.95
7098	00063	DUNGEON ADVENTURE	G	9.95	8.96
7116	00067	EDASM	U	8.83	7.95
7117	00067D	EDASM SDX (DISC)	U	9.94	8.95
7118	00066	EMERALD ISLE	G	7.72	6.95
7119	00038	ESCAPE FROM ZARKOS	G+J	7.72	6.95
7120	00081	EXTENDED BASIC	U	8.28	7.45
7121		F1 SIMULATOR	G+J	5.50	4.95
7140	00082	FATHOMS DEEP	G+J	7.72	6.95
7141	00090	FIG FORTH	L	17.50	15.75
7142	00090D	FIG FORTH SDX (DISC)	L	17.50	15.75
7143	00055	FIREHOUSE FREDDIE	G+J	7.72	6.95
7144	00021	FIRST LETTERS 1	E	9.94	8.95
7146	00037	FLUMMOX	G+J	7.72	6.95
7166	00052	GAUNTLET	G	7.72	6.95
7167	00102	GHOSTLY CASTLE	G	3.83	3.45
7168	00031	GOLDMINE	G+J	7.72	6.95
7169	00069	GRAPHICS	U	6.61	5.95
7190	00072	HAWKWARS	G+J	6.61	5.95
7191	00065	HELI-MATHS	E	8.28	7.45
7192	00139	HIGHWAY ENCOUNTER	G+J	8.83	7.95
7193	00034	HUNCHY	G+J	6.61	5.95
7213	00083	ICEBERG	G+J	6.61	5.95
7214		INTO OBLIVION	G+J	6.61	5.95
7233	00105	JET SET WILLY (TEMPORARY NOT AVAILABLE)	G	9.94	8.95
7235	00097	JUMPING JACK FLASH	G+J	6.61	5.95
7255	00115	KARATE KING	G+J	7.72	6.95
7256	00016	KEY TO TIME	G	7.72	6.95
7257	00042	KILOPEDE	G+J	7.72	6.95
7258	00019	KNUCKLES	G+J	8.83	7.95

Issue 3 & 4 special

memopad vol 0011

Software

PRICE SUBJECT TO CHANGE WITHOUT NOTICE AND ALSO SUBJECT TO POST AND PACKING.

STOCK NO.	MANUFACT. REFERENCE	TITLE	TYPE	RETAIL PRICE INCL.VAT	MEMBERS PRICE INCL.VAT
7279	00032	LITTLE DEVILS	G+J	6.61	5.95
7280	00024	LORDS OF TIME	G	9.95	8.96
7300	00035	MISSILE COMMAND & ARCADIANS	G+J	6.61	5.95
7301	00070	MAN FROM GRANNY	G	6.61	5.95
7302	00104	MANIC MINER (TEMPORARY NOT AVAILABLE)	G	7.72	6.95
7305	00022	MATHS 1	E	9.94	8.95
7306	00013	MAXIMA	G+J	7.72	6.95
7307	00086	MEMOCHEQUE	U	7.72	6.95
7308	00075	MEMOSKETCH	U+J	8.83	7.95
7309	00075D	MEMOSKETCH SDX (DISC)	U+J	9.94	8.95
7310	00089	MINER DICK	G+J	7.72	6.95
7311	00044	MISSION ALPHATRON	G+J	6.61	5.95
7312	00030	MISSION OMEGA	G+J	6.61	5.95
7313	00054	MURDER AT MANOR	G	8.28	7.45
7314	00010	MUSIC PAD	U	7.72	6.95
7333	00003	NEMO	G+J	7.72	6.95
7334	00131	NETWORK LOADER	U	9.94	8.95
7354	00112	OBLITERATION ZONE	G+J	7.72	6.95
7355	00045	OBLOIDS	G+J	7.72	6.95
7356		ONE MAN AND HIS DROID	G+J	5.50	4.95
7375	00129	PAINTBOX	U	6.61	5.95
7376	00001	PAYROLL	B	23.61	21.25
7377	00005	PHAID	G+J	7.72	6.95
7378	00061	PHYSICS 1	E	9.94	8.95
7380	00012	PONTOON & BLACKJACK	G	7.72	6.95
7381	00009	POT HOLE PETE	G+J	7.72	6.95
7382	00040	PURCHASE LEDGER	B	14.17	12.75
7402	00048	QOGO	G+J	7.72	6.95
7403	00076	QOGO 2	G+J	7.72	6.95
7404	00095	QUANTUM	G+J	6.61	5.95
7405	00109	QUAZZIA	G+J	7.72	6.95
7427	00064	RETURN TO EDEN	G	9.95	8.96
7428	00020	REVERSI	G	8.83	7.95
7429	00114	ROLLA BEARING	G+J	7.72	6.95
7430	00100	RUTHLESS B.	G	3.83	3.45
7450	00002	SALES LEDGER	B	17.50	15.75
7451	00029	SALTY SAM	G+J	6.61	5.95
7452	00113	SEPULCRI SCELERATI	G	7.72	6.95
7453	00101	SLOOPY'S CHRISTMAS	G	3.83	3.45
7454	00116	SMG	G+J	7.72	6.95
7455	00049	SNAPPO	G+J	7.72	6.95
7456	00140	TOURNAMENT SNOOKER	G	8.83	7.95
7457	00023	SNOWBALL	G	9.95	8.96
7458	00036	SON OF PETE	G+J	7.72	6.95
7459		SOUL OF A ROBOT	G+J	5.50	4.95
7460		SPELLBOUND	G+J	5.50	4.95
7461	00026	SPELLI-COPTER	G	7.72	6.95
7462	00017	STAR COMMAND	G	7.72	6.95
7463	00014	SUPA CODER	U	8.83	7.95
7464	00084	SUPER BIKE	G+J	6.61	5.95
7465	00004	SUPER MINEFIELD	G	7.72	6.95
7466	00093	SURFACE SCANNER	G+J	7.72	6.95
7490	00039	TAPE TO DISC	U	7.72	6.95
7491	00007	TAPEWORM	G+J	7.72	6.95
7492	00088	TARGET ZONE	G+J	7.72	6.95
7493	00118	THE DESIGNER	U	9.95	8.96
7494	00128	THE WALL	G+J	6.61	5.95
7495	00051	THE ZOO GAME	G	7.72	6.95
7497	00006	TOADO	G+J	7.72	6.95
7500	00018	TURBO	G+J	7.72	6.95
7520	00117	USER BASIC	U	9.95	8.96
7521	00117D	USER BASIC SDX (DISC)	U	11.05	9.95
7522	00079	USER EXTEND	U	8.83	7.95
7523	00027	UTILITIES 1	U	7.72	6.95
7524	00027D	UTILITIES SDX (DISC)	U	11.05	9.95
7542	00091	VERNON & VAMPIRES	G	6.61	5.95
7566	00060	WORD & PICTURE	E	9.94	8.95

Special Offers

STOCK NO.	MEMOTECH SOFTWARE (CASSETTE)	NORMAL	SPECIAL OFFER
	Alice in Wonderland	£ 6.95	£ 4.95
	Astromilon	£ 6.95	£ 4.95
	Astropac	£ 6.95	£ 4.95
	Backgammon	£ 7.95	£ 5.95
	Caves of Orb	£ 5.95	£ 3.95
	Cobra	£ 6.95	£ 4.95
	Combat	£ 3.95	£ 2.75
	Crystal	£ 6.95	£ 4.95
	Dennis goes Bananas	£ 3.45	£ 2.75
	Dennis and the Chicken	£ 3.45	£ 2.75
	Dennis and the Circus	£ 3.45	£ 2.75
	Downstream Danger	£ 6.95	£ 4.95
	Draughts	£ 6.95	£ 4.95
	Drive the Cee 5	£ 6.95	£ 4.95
	First Letters 1	£ 8.95	£ 6.95
	Gauntlet	£ 6.95	£ 4.95
	Ghostly Castle	£ 3.45	£ 2.75
	Graphics	£ 5.95	£ 3.95
	Memosketch	£ 7.95	£ 5.95
	Miner Dick	£ 6.95	£ 4.95
	Nemo	£ 6.95	£ 4.95
	Obliteration Zone	£ 6.95	£ 4.95
	Obloids	£ 6.95	£ 4.95
	Paintbox	£ 5.95	£ 3.95
	Payroll	£21.25	£16.95
	Phaid	£ 6.95	£ 4.95
	Pontoon & Blackjack	£ 6.95	£ 4.95
	Purchase Ledger	£12.75	£10.75
	Quazzia	£ 6.95	£ 4.95
	Reversi	£ 7.95	£ 5.95
	Sepulcri Scelerati	£ 6.95	£ 4.95
	Tournament Snooker	£ 7.95	£ 5.95
	Soul of a Robot	£ 4.95	£ 3.75
	Star Command	£ 6.95	£ 4.95
	Tapeworm	£ 6.95	£ 4.95
	Target Zone	£ 6.95	£ 4.95
	Toado	£ 6.95	£ 4.95
	User Basic	£ 8.96	£ 6.95
	Utilities 1	£ 6.95	£ 4.95
	5.25" Empty Library Cases		£ 1.00
	3½" Empty Library Cases Holds 20 Discs		£13.95
	Holds 40 Discs		£19.95
	3½" DD/DS Box of 10 in plastic library cases		£31.75