
MEMOPAD
Volume 3
Issue 788

Volume 011 memopad Memotech Computer User Group Number 7

Editorial

Hello Readers,

I am sure you will all be pleased to know that the ranks of members are swelling, the past few months has seen an influx of new users and it would seem that the Memotech has seen a new lease of life, let's hope that the software houses take note of this and consider developing some new software.

As I mentioned in my last editorial M.C.L. have replaced the DMX 80 printer with the new DMX 100, which is selling well.

I have been requested by Mrs Exton in our Sales Office to ask anyone ordering add-on's and disc based software to state clearly which machine they have and in the case of discs the exact specifications of the disc drive. As you can no doubt imagine there are now a number of different systems and drives and this can cause great confusion when dealing with a large quantity of orders. If everyone states clearly which system they have then we could avoid unnecessary delays and make Mrs Extons life a lot easier, for which she will be eternally grateful.

Sales of "THE SOURCE" have exceeded all expectations and we would like to apologise to all members who have had their cheques returned. We have twisted Keith's arm up his back and he has agreed to allow us to print a further five hundred. We did make a note of those people who were dissapointed and if you return your cheques you will get priority.

Many thanks to those of you who responded to the appeals from members struggling with Microcosm, it is refreshing to realise that our members want to help each other, but I must add that one or two of you gave a little too much away and since there is so much at stake we have found it neccesary to edit some of the replies.

The sales of Memotech computers were revitalised last month when a London company with Middle East connections bought up all existing bancrupt stock and over seven thousand machines were sold during May.

It is now certain that Football manager will be released on the Memotech and will be available from the end of the month at £6.95. This is the 100% machine code version from the Addictive Games best selling catalogue.

Sue

Number 7

memopad
Memotech Computer User Group

Volume 011

CONTENTS

What's In It For You!

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

COVER PRICE #1.35

AVAILABLE BY SUBSCRIPTION ONLY.

Volume 011 memopad Memotech Computer User Group Number 7

VIEW Letters TO THE EDITOR POINT

CP/M TIPS

Most readers who have ever experimented with system programming in CP/M assembler, or in a language such as Pascal or C which lets you do that sort of thing at a high level have had occasional accidents which have corrupted directory entries so that the files affected still take up disc space, but their names are now unreadable rubbish which cannot be got at by commands transmitted by the CCP.

While restoring some sort of normality to these files is a fairly major project, there is a very simple way to get rid of them but leave the rest of the disc contents undamaged.

The SDX/FDX system disc contains a file called ERAQ.COM which behaves like the ERA command except that it gives you an opportunity to type YES or NO with each call of the function, aborting to the next matching file if you type 'N'.

What the manual doesn't say is that since it bypasses the CCP, ERAQ *.* will display and erase the corrupted files, and may save a lot of irritating reformatting and copying.

Readers who have used CP/M 3.X systems such as the one on the Amstrad machines may know of the file type 'PROFILE.SUB' which allows you to auto-start your discs. If you invoke 'STARTUP.COM' with a command string such as STARTUP FDXB7 TOADO.BAS', this string will be passed to the CCP every time CP/M is booted from the disc. To change or abort the auto-start sequence, change the disc from within CP/M with the C command and re-invoke STARTUP."STARTUP DIR STAT" will display directory and free space on your disc and the modified CP/M can be transferred in the usual ways to new discs.

DR. BRIAN HOUGHTON.

Dear Sir,

While reading one of the 'Popular Computing' magazine I came across an advertisment for discs that I thought may be of interest to other 250K SDX disc owners. The unusual thing about these discs are that they have two write notches and two index holes, so the disc can be flipped over and the other side used . I bought a pack of ten and to date have had no problems with them. The name of the discs is, believe it or not. BANANA Reversible Mini Discs, and they cost #8.86 + vat for a box of ten. They can be bought from FREEPOST DISCING INTERNATIONAL, 1 Royal Parade, HINDHEAD, Surrey. SU26 6TD

Still on the subject of the SDX, have you ever accidently erased a program or file, come on own up, I have. There's no need to jump up and down cussin' and a holl'erin any more, here's a very easy way to recover them. First a little explanation. On a true CP/m system the user has the choice of saving files into a number of different areas on the disc called USER AREA'S numbered 0 - 15. The area the file belongs to is stored in the disc directory along with the file name and the sectors that contain the file data. As the SDX is a single user disc all files are stored in area 0. Now on to the good part...

When a file is erased all that happens is that the area number in the directory is changed to #E5 = 229. The sdx does not have a command to allow you to change the user areas, however, the BDOS code to change areas is there, and normally you would have to write the appropriate assembly call to BDOS. A more brutal approach (and the only one that will work in this case) is to poke the number into memory directly so .. POKE 55873,229 then enter USER DIR all the erased files will be displayed. To recover a program al you have to do is :-

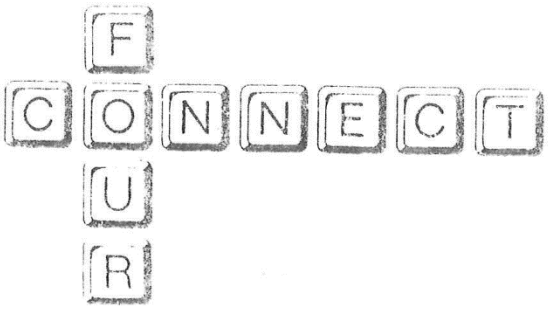
```
POKE   55873,229      (select area 229)
USER   LOAD "program.typ"
POKE   55873.0
USER   SAVE "program.typ"
```

This method should work for all basic and inary files where the start and end addresses are known, so that you can resave the file again using the appropriate command. Trying to recover text and data files may be a problem as the length of the file is not known. You could try loading the program that created the file, insert POKE 55873.229, read in the file or data, then POKE 55873,0 and write the file/data back to disc.

A word of warning, do not write any further data to the disc after a file has been accidently erased, save anything you need to on another disc, and then try to recover it.

Yours sincerely

Eric Roy



This month we are going to install two routines. One will set up the VDP and the other will be used as a screen handler for all our printing routines. Once again we are keeping the routines universal so that they can be easily transported to other machines utilising the Texas VDP.

To start typing this section of code first load your program into the computer.

```
TYPE .. ASSEM 10    <RET>
TYPE    I #4171     <RET>       512 OWNERS
TYPE    I #8171     <RET>       500 OWNERS
```

You should now be in the correct position. That is, all the code you now type will push the Label START higher up in memory, and insert all this new code in front of it.

One point to mention is: If you have included any of the comments, your memory addresses may vary because the MTX in-line assembler increments memory addresses past the comments. This doesn't matter as long as the difference in memory addresses is not due to mistakes in the actual typing of the code. Start by typing the following:

```
I #START
```

G2INIT initialises the VDP to our own specification. The Pattern Table (screen) is set at 3C00H. The Pattern Generator Table is set to 0000H in Vram, of course.

On initialisation the screen is set to a bit mapped mode. In order to utilise this screen we have to place a count of 0 - 255 in each of the 256 screen locations - we do this three times .... 3 * 256 = 768 screen locations. Why this has to be done has been mentioned on numerous occasions in past issues of Memopad. In order to preserve the character set, it is placed in locations starting at 1900H in Vram. When we want to print a character on the screen it is simply pulled from the correct location, bit by bit, and placed in the correct location in the generator table. We don't have to worry about this - KSUB1 takes care of all the hassle !

We shall be using a universal print routine that will operate in tandem with KSUB1 and I will supply the code for this in the next edition.

One very useful point about KSUB1 is the way it allows us to update the cursor position. IX+00 holds the x position and IX+01 holds the y position. To move the cursor to the first screen position all we need to do is:

```
LD      (IX+00H),0
LD      (IX+01H),0
```

The next time that the printing routine is called it will automatically update the position of the cursor to location 0,0.

How these routines work will become a lot clearer when I give you the interfacing routines in the next edition.

To utilise these routines on other machines only the port addresses need to be changed. Cp/m owners who are lucky enough to use M80 can equate the addresses at the very start of the program.

```
VDP     EQU     02
VRM     EQU     01
```

This makes it easier to change when transferring to another machine.

```
                        ;
                        ;G2INIT.MAC
                        ;
                        ;
                        .COMMENT/
                        This subroutine sets up the VDP in a graphic mode 2 bit-mapped
                        screen.  It loads all the pattern name table with numbers from
                        0=255 THREE times for each third of the screen.  Characters are
                        then displayed by loading their bit patterns into the generator
                        table and setting the relevant colour byte.

                        /
4171                    G2INIT:
4171  21 3C00                   LD      HL,3C00H        ;START OF PATTERN TABLE
4174  CD 429C                   CALL    ADDOUT
4177  01 0000                   LD      BC,0000H        ;INITIALISE BC FOR ZERO
417A                    G2LP:
417A  79                        LD      A,C             ;1 2 3 4 ... ETC
417B  D3 01                     OUT     (01),A
417D  03                        INC     BC
417E  7B                        LD      A,B
417F  FE 03                     CP      3               ;HAVE WE DONE IT ALL 3 TIMES ?
4181  20 F7                     JR      NZ,G2LP

                        ;NOW SEND CHARACTER SET DETAILS

4183  21 1900                   LD      HL,1900H        ;Set start actual start = 1800H
4186  CD 429C                   CALL    ADDOUT
4189  21 43BC                   LD      HL,CHARX        ;Character set details
418C  01 02E0                   LD      BC,2E0H         ;Char count
418F  7E                LOPA:   LD      A,(HL)          ;Get a value
4190  D3 01                     OUT     (01),A          ;Send it
4192  23                        INC     HL
4193  0B                        DEC     BC
4194  79                        LD      A,C
4195  B0                        OR      B
4196  20 F7                     JR      NZ,LOPA
4198  C9                        RET
```

```
                        ;ROUTINE TO LOAD CHARACTER INTO VRAM AND SET COLOUR BYTE
                        ;
                        ;New routine to load character into Vram and set the colour byte
                        ;Routine to be called from main loop.
                        ;
                        ;************************
1900                    TEXT    EQU     1900H
                        ;************************
                        ;
                        ;Assumes that IX register set up as follows::::
                        ;IX+00H points to X co-ordinate
                        ;IX+01H points to Y co-ordinate
                        ;IX+02H holds current character value
                        ;IX+03H holds current ink  colour value
                        ;IX+04H holds paper colour
                        ;Preserves HL,BC,DE registers
                        ;
                        ;************************
                        ;
4199                    KSUB1:
4199    C5                      PUSH    BC
419A    D5                      PUSH    DE
419B    E5                      PUSH    HL
419C    DD 21 4297              LD      IX,XCORD        ;ALIGN IX FOR ABOVE TABLE
41A0    DD 77 02                LD      (IX+02H),A      ;SAVE CURRENT CHARACTER
41A3    FE 20                   CP      32              ;IS IT A CONTROL CHARACTER?
41A5    DA 421D                 JP      C,CONTROL       ;LESS THAN 32 YES IT IS A CONTROL.
41A8    D6 20                   SUB     32              ;GET RID OF ALL CONTROL CODES
41AA    16 00                   LD      D,0             ;WE CAN NOW ALIGN TO RIGHT POSITION
41AC    5F                      LD      E,A             ;DO ALIGNMENT
41AD    CB 23                   SLA     E               ;
41AF    CB 12                   RL      D               ;MAKE SURE NO FALLOUT FROM E ESCAPES!
41B1    CB 23                   SLA     E
41B3    CB 12                   RL      D
41B5    CB 23                   SLA     E
41B7    CB 12                   RL      D               ;NOW CHAR*8 = CORRECT POSITION.
41B9    21 1900                 LD      HL,TEXT         ;POINT TO START OF TEXT IN VRAM = EQU****
41BC    19                      ADD     HL,DE           ;ALIGNED TO CORRECT CHARACTER
41BD    EB                      EX      DE,HL           ;DE=CHARACTER IN VRAM
41BE    D5                      PUSH    DE              ;WE WILL NEED IT AGAIN SO SAVE IT
                        ;
                        .COMMENT ?
                        **
                        The following routine converts the X,Y co-ordinates to Vram position which
                        in the case of a Bit Mapped screen is set up from pattern generator table
                        located at 0000H through to 17FFH in Vram.

                        Formula for calculations is based on 32 characters per line & 8 bytes per
                        character.

                        Position in Vram = No lines in Ycord * 32 * 8 + (Number of chars in Xcord*8)

                        ?
```

Connect Four Assembler version for magazines only (c) K. Hook 1987 MACRO-80 3.44 09-Dec-81 PAGE 1-5

```
41BF    26 00                   LD      H,0
41C1    DD 6E 01                LD      L,(IX+01H)      ;Y CO-ORDINATE
41C4    06 08                   LD      B,08            ;FOR * 256 TRICK
41C6                    LPX:
41C6    29                      ADD     HL,HL           ;*2 *4  *8 ETC....
41C7    10 FD                   DJNZ    LPX
41C9    16 00                   LD      D,0             ;HL=Y*32 * 8 = NO COLS PER LINE *
                                                        ;NUMBER OF BYTES PER CHARACTER
41CB    DD 5E 00                LD      E,(IX+00H)      ;GET X CO-ORD
41CE    CB 23                   SLA     E
41D0    CB 23                   SLA     E               ;*8 FOR NUMBER OF BYTES PER CHARACTER
41D2    CB 23                   SLA     E
41D4    19                      ADD     HL,DE           ;HL-> TO CORRECT PLACE IN VRAM
41D5    D1                      POP     DE              ;DE STILL -> TO CHARACTER IN VRAM
41D6    CD 41F8                 CALL    SENDX           ;GO AND SEND IT
                        ;
                        ;
41D9    DD 7E 00                LD      A,(IX+00H)      ;X CO-ORD
41DC    3C                      INC     A               ;INCREMENT X CO-ORD
41DD    DD 77 00                LD      (IX+00H),A
41E0    FE 20                   CP      32
41E2    20 10                   JR      NZ,BACK         ;IF NOT MAX GO BACK
41E4    AF                      XOR     A               ;ZERO A REG MAX SCREEN WIDTH REACHED
41E5    DD 77 00                LD      (IX+00H),A      ;RESET X CO-ORD
41EB                    LNFED:
41E8    DD 7E 01                LD      A,(IX+01H)      ;GET Y CO-ORD
41EB    3C                      INC     A
41EC    FE 18                   CP      24              ;MAX DEPTH ?
41EE    20 04                   JR      NZ,BACK
41F0    AF                      XOR     A
41F1    DD 77 01                LD      (IX+01H),A
                        ;
41F4                    BACK:
41F4    E1                      POP     HL
41F5    D1                      POP     DE
41F6    C1                      POP     BC
41F7    C9                      RET

                        .COMMENT *
                        HL NOW -> TO VRAM POSITION FOR BIT-MAPPEED MODE WHICH IS IN FACT THE PATTERN
                        GENERATOR TABLE STARTING AT 0000H IN VRAM.

                        *
41F8                    SENDX:
41F8    E5                      PUSH    HL      ;SAVE PRINT POSITION
                        ;
                        ;
                        .COMMENT *
                        The following routine takes the screen position pointed to by HL
                        the character bytes (in Vram) pointed to by DE.
                        Reads them from Vram and prints them at the screen position
                        held in HL. It then sets the CORRECT COLOUR BYTES from the location
                        pointed to by (IX+03H)
                        *
                        ;
```

```
Connect Four Assembler version for magazines only  (c) K. Hook 1987     MACRO-80 3.44   09-Dec-81       PAGE    1-6


41F9    06 08                   LD      B,08H
41FB                    LPA:
41FB    EB                      EX      DE,HL           ;HL= PATTERN ADDRESS:DE = SCREEN POS
41FC    CD 42A7                 CALL    ADDIN           ;SEND DATA
41FF    DB 01                   IN      A,(01)          ;GET BYTE OF CHARACTER
4201    EB                      EX      DE,HL           ;AS WE WERE!
4202    CD 429C                 CALL    ADDOUT          ;NOW SEND SCREEN ADDRESS
4205    D3 01                   OUT     (01),A          ;SEND DATA
4207    23                      INC     HL              ;INC SCREEN
4208    13                      INC     DE              ;INC CHARACTER POINTER
4209    10 F0                   DJNZ    LPA             ;CONTINUE FOR 8 BYTES
                        ;
                        ;COLOUR ROUTINE
                        ;
420B    E1                      POP     HL              ;GET ORIGINAL SCREEN POSITION
420C    CB EC                   SET     5,H             ;ALIGN TO COLOUR TABLE
420E    DD 7E 03                LD      A,(IX+03H)      ;GET COLOUR INFORMATION
4211    06 08                   LD      B,08
4213    CD 429C                 CALL    ADDOUT
4216    07              LPB:    RLCA
4217    0F                      RRCA                    ;HANG ABOUT A BIT
4218    D3 01                   OUT     (01),A          ;01 for the MTX change for another machine
421A    10 FA                   DJNZ    LPB
421C    C9                      RET
                        ;
                        .COMMENT £
                        Routine to handle control characters within data
                        It assumes that:-
                        3 = CSR x,y
                        4 = PAPER n
                        6 = INK n
                        8 = Backspace
                        10 = Linefeed


                        £
421D                    CONTROL:
421D    21 4296                 LD      HL,CONTRL
4220    FE 03                   CP      3               ;IS IT CSR X,Y ?
4222    20 17                   JR      NZ,CONTA        ;NO BUT CHECK IF LAST BYTE WAS
4224    CB 7E                   BIT     7,(HL)
4226    20 3C                   JR      NZ,CONTD
4228    CB 66                   BIT     4,(HL)
422A    C2 427F                 JP      NZ,CONTF        ;MUST BE PAPER COLOUR
422D    CB 46                   BIT     0,(HL)
422F    20 0A                   JR      NZ,CONTA
4231    CB 4E                   BIT     1,(HL)
4233    20 06                   JR      NZ,CONTA

4235    CB C6                   SET     0,(HL)          ;FLAG THAT X CO-ORD EXPECTED
4237    CB CE                   SET     1,(HL)          ;AND Y CO-ORD ALSO EXPECTED
4239    18 B9                   JR      BACK

423B                    CONTA:
423B    CB 46                   BIT     0,(HL)          ;IS THIS BYTE X COORD
423D    28 08                   JR      Z,CONTB         ;NO GO CHECK IF ITS Y COORD
```

```
Connect Four Assembler version for magazines only  (c) K. Hook 1987     MACRO-80 3.44   09-Dec-81       PAGE    1-7


423F   DD 77 00                 LD      (IX+00H),A      ;O.K ITS X COORD
4242   CB 86                    RES     0,(HL)          ;RESET X FLAG
4244   C3 41F4                  JP      BACK

4247                        CONTB:
4247   CB 4E                    BIT     1,(HL)          ;IS IT A Y COORD
4249   28 08                    JR      Z,CONTC         ;NO SO CHECK ANOTHER CONTROL
424B   DD 77 01                 LD      (IX+01H),A      ;SAVE Y COORD
424E   CB 8E                    RES     1,(HL)          ;RESET FLAG
4250   C3 41F4                  JP      BACK

4253                        CONTC:
4253   FE 06                    CP      6
4255   20 0D                    JR      NZ,CONTD        ;NOT INK CONTROL BUT CHECK IF LAST WAS
4257   CB 7E                    BIT     7,(HL)          ;
4259   20 09                    JR      NZ,CONTD
425B   CB 66                    BIT     4,(HL)          ;MAKE SURE PAPER ISN'T EXPECTED
425D   20 05                    JR      NZ,CONTD
425F   CB FE                    SET     7,(HL)          ;SET FLAG
4261   C3 41F4                  JP      BACK

4264                        CONTD:
4264   CB 7E                    BIT     7,(HL)          ;IS FLAG SET TO EXPECT THIS TO BE
4266   28 10                    JR      Z,CONTE         ;INK COLOUR. IF NOT CHK NXT CONTRL
4268   A7                       AND     A
4269   07                       RLCA
426A   07                       RLCA
426B   07                       RLCA
426C   07                       RLCA
426D   DD B6 04                 OR      (IX+04H)        ;PAPER BYTE MUST BE DEFINED FOR EACH PROGRAM
4270   DD 77 03                 LD      (IX+03H),A      ;SAVE IT
4273   CB BE                    RES     7,(HL)          ;RESET FLAG
4275   C3 41F4                  JP      BACK

4278                        CONTE:
4278   FE 0A                    CP      10              ;IS IT A LINE FEED
427A   20 03                    JR      NZ,CONTF
427C   C3 41E8                  JP      LNFED           ;GO AND DO IT BEFORE RETURNING

427F                        CONTF:
427F   FE 04                    CP      4
4281   C2 4289                  JP      NZ,CONTG
4284   CB E6                    SET     4,(HL)
4286   C3 41F4                  JP      BACK            ;UNTIL WE'VE FOUND WHAT'S GOING ON

4289                        CONTG:
4289   CB 66                    BIT     4,(HL)
428B   CA 41F4                  JP      Z,BACK
428E   DD 77 04                 LD      (IX+04H),A
4291   CB A6                    RES     4,(HL)
4293   C3 41F4                  JP      BACK
                            ;************************
                            ;
4296   00                   CONTRL: DB      00H
4297   00                   XCORD:  DB      00H
4298   00                   YCORD:  DB      00H
```

SLVCODE

As you know the Memotech does not allow you to save or load assembly code from memory directly to tape. OK!! I know that there are short assembly programs that will enable you to do just that, and they work alright, but if you do not keep an accurate record of there start address, length and position on tape then you can get into quite a muddle trying to load it back into memory.

The following utility will save, load and verify code to tape but this time it is preceding with a header, similar to basic save, that enable you to specify a filename it's start address, length and reload address. The three new USER commands are..

USER SCODE "filename", start addr, length (,entry)

The filename, start address and length of code must be specified, the entry parameter is optional, but if included the code will auto run from this address when reloaded. All numeric parameters can be in decimal or hex if prefixed with *EG. #A000. The only restriction on the code being saved is that must be in memory page 0.

USER LCODE "filename"(,reload addr)

Both the filename and reload address are optional eg. USER LCODE "" `RET" would load the next code file from tape. The reload address if specified will load the code to that address, so that you can save from one address and reload to another.

USER VCODE "filename"

Again the filename is optional, if ommited then the next code file on tape will be used for verifing. E.g. USER VCODE "" `RET".

Basic programs and code files can be saved on the same tape as a basic header starts with *FF byte while the code header starts with an *FE byte, so there is no danger of accidently LOADING a code file.

The utility can be used by both tape and SDX disc owners, the SYSTEM routine setting up the USER jump address as required.

```
100 REM
110 REM
120 REM
130 REM
140 REM
150 REM
160 REM ------------------------
170 REM  PROGRAM NAME - SLVCODE.BAS
180 REM
190 REM  THREE NEW USER COMMANDS THAT
200 REM  ALLOW MEMORY TO BE SAVED TO
210 REM  TAPE WITH HEADER INFORMATION.
220 REM
230 REM
240 REM
250 CODE

41B2 SYSTEM:  LD HL,(#FABA)       ;VECTOR USER
41B5          LD BC,#F5B3         ;WILL WORK ON TAPE
41B8          AND A       ;OR SDX SYSTEM
41B9          SBC HL,BC
41BB          JR NZ,SYST
41BD SYSD:    LD HL,NEWCOM        ;SDX ROUTINE
41C0          LD (#F5B4),HL
41C3          LD A#C3
41C5          LD (#F5B3),A
41C8          LD HL,#F586
41CB          JR SYSVEC
41CD SYST:    LD A,07  ;TAPE ROUTINE
41CF          LD (#FA88),A
41D2          LD A,#C9
41D4          LD (#FA87),A
41D7          LD HL,NEWCOM
41DA          LD (#FA8A),HL
41DD          LD A,#C3
41DF          LD (#FA89),A
41E2          LD HL,UNDEF
41E5 SYSVEC:  LD (#FA8D),HL       ;JUMP TO UNDEFINED
41E8          LD (#FA8C),A        ;OR SDX ENTRY
41EB          RET
41EC NEWCOM:  LD (SNAME),DE
41F0          LD A,(DE)
41F1          CP "S"      ;SAVE CODE
41F3          JP Z,SAVECOM
41F6          CP "L"      ;LOAD
41F8          JP Z,LOADCOM
41FB          CP "V"      ;VERIFY
41FD          JP Z,VERICOM
4200 NAMERR:  LD A,(#FAD2)        ;EXIT IF NO MATCH
4203          LD DE,(SNAME)
4207          JP #FA8C
420A SAVECOM: CALL FILEN ;GET FILENAME
420D          JP NZ,NAMERR
4210          LD A,B      ;BC=0 IF NO FILENAME
4211          OR C
4212          JP Z,UNDER  ;ERROR IF NO NAME
4215          PUSH BC     ;SAVE LENGTH
4216          PUSH HL
4217          CALL PARAM  ;GET START ADDRESS
421A          LD (START),BC
421E          LD A,(DE)
421F          CP ","
4221          JP NZ,UNDEF ;ERROR IF NO CODE LENGTH
4224          CALL PARAM  ;GET LENGTH
4227          LD (LENGTH),BC
422B          LD BC,0000  ;PRE-LOAD BC WITH
422E          LD A,(DE)   ;NO ENTRY ADDRESS
422F          CP ","
4231          JR NZ,S1    ;NO ENTRY PARAM FOUND
4233          CALL PARAM  ;GET ENTRY ADDRESS
4236 S1:      LD (ENTRY),BC
423A          POP HL
423B          POP BC
423C          PUSH DE
423D          XOR A       ;SIGNAL SAVE
423E          LD (#FD68),A
4241          CALL #0BEF  ;PADS OUT FILENAME
4244          PUSH DE     ;WITH SPACES
4245          LD DE,START
4248          EX DE,HL
4249          LD B,06     ;INSERT START LENGTH
424B          LDIR        ; and entry into header.
424D          RST 10
424E          DB #9C      ; Ready to save
424F          DB "Select record, press <RET>",#0D,#0A
426B S2:      CALL #0079
426E          CP 03       ; Break ?
4270          JP Z,#0250  ; Yes, ret to basic
4273          CP #0D
4275          JR NZ,S2    ; Wait for RET key
4277          POP HL
4278          LD DE,#16   ; Length of header
427B          CALL #0AAE  ; Save it
427E          LD DE,0000  ; Delay before saving
4281 S3:      DEC DE      ; code block.
4282          LD A,D
4283          OR E
4284          JR NZ,S3
4286          LD HL,(START)
4289          LD DE,(LENGTH)
428D          CALL #0AAE  ; Save code block
4290          POP DE
4291          RET         ; Back to basic
4292 VERICOM: CALL FILEN
4295          JP NZ,NAMERR
4298          PUSH DE
4299          LD A,01     ; Signal verifing
429B          JR L1
429D LOADCOM: CALL FILEN
42A0          JP NZ,NAMERR
42A3          LD BC,0000  ; No reload addr.
42A6          LD A,(DE)
42A7          CP ","
42A9          JR NZ,L0
42AB          CALL PARAM  ; Get reload addr.
42AE L0:      LD (RELOAD),BC
42B2          PUSH DE
42B3          XOR A       ; Signal loading
42B4 L1:      PUSH AF
42B5          XOR A
42B6          LD (#FD67),A
42B9          INC A
42BA          LD (#FD68),A
42BD          CALL #0BEF
42C0          PUSH HL
42C1          EXX
42C2          POP HL
42C3          LD DE,#16   ; Length of header
42C6 L2:      PUSH AF
42C7 L3:      PUSH DE
42C8          PUSH HL
42C9          CALL #0AAE  ; Load header
42CC          POP HL
42CD          POP DE
42CE          LD A,(HL)
42CF          CP #FE      ; Is it a code header
42D1          JR NZ,L3    ; No, get next header
42D3          CALL #0C23  ; Compare filenames
42D6          EX AF,AF'
42D7          POP AF
42D8          JR Z,L4     ; Match if empty string
42DA          EX AF,AF'   ; i.e. LOADE,""
42DB          JR NZ,L2    ; Names don't match
42DD L4:      ADD HL,DE   ; Get start, length from
42DE          DEC HL      ; header and store.
42DF          LD DE,ENTRY
42E2          INC DE
42E3          LD BC,06
42E6          LDIR
42E8          POP AF
42E9          PUSH AF
42EA          LD (#FD67),A    ; Load or Verify?
42ED          AND A
42EE          JR Z,L5     ; Load code
42F0          RST 10
42F1          DB #8F
42F2          DB "Verifing code",#0D,#0A
4301          JR L6
4303 L5:      RST 10
4304          DB #8E
4305          DB "Loading code",#0D,#0A
4313          LD HL,(RELOAD) ; Test if code
```

```
4316            LD A,H      ; to be reloaded to
4317            OR L        ; a new address.
4318            JR NZ,L7
431A L6:        LD HL,(START)       ; Now Load or Verify
431D L7:        LD DE,(LENGTH)
4321            CALL #0AAE
4324            POP AF      ; Load / Verify flag
4325            POP DE      ; Position in basic
4326            AND A
4327            RET NZ      ; Return if verifing
4328            LD HL,(ENTRY)       ; Test if code
432B            LD A,H      ; is to auto run
432C            OR L        ; from entry address
432D            RET Z       ; No
432E            JP (HL)     ; Jump to code entry
432F PARAM:     INC DE      ; Get paramaters
4330            CALL #1ACD ; Get hex or decimal
4333            JP NZ,UNDEF         ; Error
4336            RET         ; Param in BC
4337 FILEN:     LD HL,CNAME         ; HL->valid name
433A            LD B,(HL)   ; B=length
4338 F1:        INC HL
433C            INC DE
433D            LD A,(DE)
433E            CP (HL)     ; Compare
433F            RET NZ      ; No match
4340            DJNZ F1
```

```
4342            PUSH AF     ; Save Z (match) flag
4343            INC DE      ; Start to build header
4344            LD HL,(#FAB1)       ; HL->calc stack
4347            LD (HL),#FE         ; Code identifier byte
4349            INC HL
434A            LD (#FAB1),HL
434D            RST 28      ; EVALSE, get filename
434E            DB #88      ; and put in calc stack
434F            POP AF
4350            RET
4351 CNAME:     DB 04,"CODE"        ; Command name
4356 START:     DW 00       ; Store for variables
4358 LENGTH:    DW 00
435A ENTRY:     DW 00
435C RELOAD:    DW 00
435E SNAME:     DW 00
4360 UNDEF:     RST 28      ; Error
4361            DB #26
4362            RET
```

Symbols:

SYSTEM	41B2	SYST	41CD
NEWCOM	41EC	SYSD	41BD
SYSVEC	41E5	UNDEF	4360
SNAME	435E	SAVECOM	420A
LOADCOM	429D	VERICOM	4292
NAMERR	4200	FILEN	4337
PARAM	432F	START	4356
LENGTH	4358	S1	4236
ENTRY	435A	S2	426B
S3	4281	L1	42B4
L0	42AE	RELOAD	435C
L2	42C6	L3	42C7
L4	42DD	L5	4303
L6	431A	L7	431D
CNAME	4351	F1	4338

MTX TOKENS

MTX TOKENS

Have you ever wondered how the MTX's basic interpreter stores a basic line in memory? Well the MTX like the Sinclair ZX Specrtum and Commodore machines stores the basic keyword as tokens. The keywords are converted into a ASCII code between 128 and 255. If the user presses any of the function keys which represent ASCII codes 128 - 136 and press enter then the keyword is displayed, ie. F1=REM. This is a convenient starting point.Type in the basic line in fig 1. Now enter the front panel by typing PANEL (a full description of the front panel can be found in issue 4 of the Memopad). Now press D for display and enter 4000 (or 8000 on the 500, from now on the 500 owners should change any initial '4' with an '8'); at this point the memory location and its contents will be displayed at the bottom of the screen. At this point press the `BRK" key in order to avoid changing any contents.

FIG 1.

```
10    REM
```

From the location 4000 you will see the following 6 bytes of data :- 06 00 0A 00 80 FF.

0600 The number of bytes used to describe line 10.

0A00 The basic line number in hexadecimal. The line number is stored as LSB/MSB format. Note that the allocation of two bytes to represent a basic line number means that the MTX can have line numbers between 0 and 65535.

80 This equals 128 decimal and represents the keyword for REM. A list of the main keywords, tokens and their address in rom is given later.

FF This marker is used to indicate the end of this basic line.

Fig. 2 shows where the next basic line would go.

fig.2

```
10  REM
20  GOTO 1
```

Line 20 gave 07 00 14 00 96 31 FF

The first 5 bytes are similar to above ie. the length of memory used, basic line number and the token for GOTO. But what is #31. Well this equal to the number 1. If you look at the ASCII table in your MTX manual, appendix section, you will see that 31 hex or 39 dec equals the number 1. This means that if we were to goto line 100, for instance, then we would need 3 ASCII codes to represent this 3 digit number, ie. 96 31 30 30 = GOTO 100.

Already this insight into the comuters internal working s has given us a tip to save memory. Basic is generally greedy and inefficient as memory goes and this has telling affects when writting long programs or programs like word processors which need as much free memory as possible. Therefore if we have many GOSUB's or GOTO's in a program then it would be more economical to have them at the begining of a of a program, ie. at low line numbers, 0 - 99 and save a number of valuable bytes.

As you will soon see there are a number of ways to save memory by looking at the MTX's method of storing basic.

One of these savings is to do with the GENPAT command. Genpat stores the data after the command as ASCII codes, as did the GOTO command. This means that the maximum number of bytes after the token will be 10X3+9 = 39, the 9 representing the commas. If you have ever used the built in assembler then you will know that a DB command plus 10 data only uses up 10 bytes as the data is stored as a hex number and not as an ASCII code. Therefore we need an extra 29 bytes of data not including the memory for the length, line number and token. It would be easier to store genpat data in the assmbler format and peek the data when needed, or write a short machine code routine to do the same. All in all the savings are immense, eg. whereas 10 DB assembler commands use up about 100 bytes, a saving of 340 bytes.

Milti-statement lines

Take the example given in figure 3 an dthe resulting memory data. As you can see only the REM keyword is assigned a token number whereas th GOTO command is stored as ASCII codes. The number of bytes used to represent fig 3 equals the number used in fig 2. This is because the 5 initial bytes used at the strat of any basic line in this case is cancelled out by the 4 bytes for the GOTO and 1 for the colon. If we use Multi-statements with keywords less than 4 characters long, ie. PHI, CLS, REM, etc then we will be saving a few bytes per line. But, if the keyword is greater than 4 characters then we are eating memory up. In this case it would be better to use a separate line for keywords :- GOSUB, GENPAT and RESTORE which will save us 1.2 or 3 bytes respectively., ie. 1 (colon) + 7 (for keyword RESTORE) = 8 compared with only 5 opening bytes for new line.

Fig 3:

```
10 REM:GOTO 1

0D 00 0A 00 80 3A 47 4F 47 4F 20 31 FF
            REM :  G  O  T  O  sp 1 end   (sp = space)
```

As you can see, there are a number of ways to make memory savings by taking a closer look at the MTX. I have only discussed a few. By experimenting you will soon see how to write efficient basic programs.

Here is a sample of the main keywords their addresses in rom and their token. (Note that the MTX basic rom is on page 1 and this means that if we want to disassemble it we will will have to move the rom into ram on page zero to get into the front panel. See appendix 8, memory map in manual).

table 1:A table of the main keywords as used by MTX basic with their corresponding token numbers and addresses.

Token	Keyword	Address (decimal)
128	REM	10154
129	CLS	05525
130	ASSEM	10116
131	AUTO	02567
132	BAUD	03384
133	VS	05622
134	CONT	10119
135	USER	64137
136	CRVS	05610
137	CLEAR	10235
138	CLOCK	10966
139	ATTR	05602
140	COLOUR	05555
141	INK	05507
142	CSR	05507
143	DATA	10154
144	PRINT	11144
145	DIM	10295
146	ADJSPR	05563
147	EDIT	10310
148	NEXT	10852
149	FOR	10343
150	GOTO	10383
151	GOSUB	10396
152	INPUT	10460
153	IF	10593
154	MVSPR	05585
155	LIST	10621
156	LET	10714
157	LLIST	10561
158	LOAD	10990
159	LPRINT	10870
160	SPRITE	05572
161	CTLSPR	05531
162	NODE	10111
163	NEW	00517
164	PAPER	05514
165	NODDY	11076
166	ON	11087
167	OUT	02557
168	PLOD	11058
169	PANEL	11256
170	GENPAT	05539
171	PAUSE	11236
172	PHI	05642
173	POKE	11266
174	RAND	11273
175	RETURN	11281
176	READ	11380
177	VIEW	05594
178	RESTORE	11404
179	ROM	00158
180	RUN	11439
181	SAVE	11011

Token	Keyword	Address (decimal)
182	SOUND	11532
183	EDITOR	10437
184	DSI	05500
185	PLOT	05744
186	STOP	11605
187	ANGLE	05628
188	SBUF	10163
189	VERIFY	11034
190	DRAW	05767
191	ARC	05838
192	CIRCLE	06159
193	LINE	05492

From codes 194-255 I don't have any addresses for them so I have decided not to include them above.

Written by:-

Alan Wilson

NODDY COMMANDS

Recently, when I was lost for something to do I st down and re-read all the previous issues of Memopad. It was not long before I was asking Myself the question, "why has nobody written a new NODDY command?". To date there has plenty of new Panel, USER and NODE commands and the system variables at #FAA1 (USERNOD) has been provided for use in the same way as the others. So now I have something to do, and here is what I found.

I wrote a short test program and set up the USERNOD locations so that it would jump to the test routine which just printed OK on the screen. When run the OK message would be displayed, but on return it always produced an error. Out came the disassembler, and with it I set out to find out how the PLOD command worked. (Can't use the Panel as the PLOD code is in the other ROM).

The PLOD command entry point is #2B32, and first it puts the page name on the calc stack, selects VS 5, saves DE and (#FAD3) in AF
to stack. A call is then made to #2EBB to find the page and execute any * commands. When a * command is found the register A is loaded letter in A is valid, and this is where the fault is. The first instruction in the routine at "2F1A is a CALL to USERNOD location, so any routine may have patched in will work, but on return the registers that scan the PLOD page and A will be corrupt, so the test on the letter in A will always fail.

One solution to this problem is shown in the two listings below, which tests the contents of A and if not a new command returns immediately. When a new command letter is found the return address to the test routine (#2F1D) is removed from the stack with POP AF. The register pair HL is used to scan the page, and at this point will point to the first letter of the new command.

Listing one does not create a new command, it just alters the way the *DISPLAY command works. Listing two does create a new command *C ink paper ie. *C5F. The two hex numbers following *C are used to set the ink (5=BLUE) and paper (F=15=WHITE) colours for the text and input screens (5+7) and should be used before the screen is displayed. *C5F *D PAGENAME. *E *R

When writing Noddy pages you can get an idea of paper and ink colours by pressing CTRL D then key A to O for paper, and CTRL F then key A to O for ink, (A=1 - @=15).

I am sure that you can think of more commands, eg. an alternative *LIST to list Noddy pages to printers that cannot change the print width. HAVE FUN!

```
100 REM ******************************
110 REM ****  NODDY DISPLAY PAGES ****
120 REM ****                        ****
130 REM ****  by Eric Roy Nov 86    ****
140 REM ******************************
150 REM
160 REM-----------------------------------
170 REM Program name  NODDISP.BAS
180 REM
190 REM Alternative Noddy page display
200 REM-----------------------------------
210 REM
220 CODE

4163 VECNOD: LD HL,NEWNOD        ; Vector Noddy * commands
4166         LD (#FAA2),HL
4169         LD A,#C3
416B         LD (#FAA1),A
416E         RET
416F NEWNOD: CP "D"         ; Is it *D command
4171         RET NZ         ; No
4172         POP AF         ; Remove return addr.
4173         RST 28         ; Noddy Display codde
4174         DB #A5         ; = GETSPACE
4175         CALL #3009 ; GETCHAR / PUTSTACK
4178         PUSH DE
4179         PUSH HL
417A         LD A,(#FAD3)
417D         PUSH AF
417E         CALL #2E39 ; Find page
4181         JP NZ,#2F99          ; Not found
4184         RST 10         ; Alternative dispay code
4185         DB #45         ; CLS VS 5
4186         PUSH BC
4187         LD B,24        ; Lines on screen
4189 D1:     PUSH BC
418A         RST 10         ; Insert blank line
418B         DB #85,3,0,0,27,"I"
4191         POP BC
4192         DJNZ D1        ; Scrolls text down
4194         POP BC
4195         RST 10         ; Cursor on bottom line
4196         DB #83,3,0,23
419A         JP #2F85       ; Display Noddy page
419D         RET
```

```
Symbols:
NEWNOD  416F    VECNOD  4163
D1      4189

230 PLOD "COLOUR"



100 REM ******************************
110 REM **** COLOURED NODDY PAGES ****
120 REM ****                      ****
130 REM ****  by Eric Roy Nov 86  ****
140 REM ******************************
150 REM
160 REM-----------------------------------
170 REM Program name  NODCOLOR.BAS
180 REM
190 REM New Noddy *C command enables
200 REM ink and paper colours to be
210 REM changed on each page.
220 REM
230 REM New *command is *Cink paper i.e
240 REM *C5F INK 5=blue PAPER 15=white
250 REM-----------------------------------
260 CODE

41EB RELOC:  LD HL,NEWNOD        ; Relocate to common ram
41EE         LD DE,#F618         ; = 63000
41F1         LD BC,#0015
41F4         LDIR
41F6 VECNOD: LD HL,#F618         ; Vector Noddy * commands
41F9         LD (#FAA2),HL
41FC         LD A,#C3
41FE         LD (#FAA1),A
4201         RET
4202 NEWNOD: CP "C"      ; Is it *C command
4204         RET NZ      ; No
4205         POP AF      ; Remove return addr.
4206         INC HL      ; HL-> ink paper numbers
4207         PUSH DE
4208         EX DE,HL    ; Now in DE
4209         CALL #1AE0  ; Convert to hex number
420C         LD A,L      ; Returned in HL A=ink paper
420D         LD (#FFB1),A        ; Set VS 5 ink paper
4210         LD (#FFCF),A        ; Set VS 7 ink paper
4213         EX DE,HL    ; HL->byte after *C?? command
4214         POP DE      ; Restore DE
4215         LD A,(HL)   ; Get byte after command
4216         RET         ; Return to command loop
4217         RET

Symbols:
NEWNOD  4202    VECNOD  41F6
RELOC   41EB

270 PLOD "COLOUR"
280 CLS : PRINT : PRINT
290 INPUT " Print Noddy pages Y/N ";A$
300 IF A$<>"y" AND A$<>"Y" THEN  STOP
310 PRINT : PRINT
320 INPUT " ENSURE PRINTER IS ON, HIT RET ";A$
330 LPRINT CHR$(27)+"Q"+CHR$(39)
340 PLOD "LLIST"
350 LPRINT CHR$(27)+"@"
```

DRAWING  ROUTINE

```
10 VS 4: CLS
20 DIM N(26),A(26,12,2)
30 FOR N=1 TO 26
40 READ N(N)
50 FOR M=1 TO N(N)
60 READ A(N,M,1),A(N,M,2)
70 NEXT M: NEXT N
80 CLS : INPUT "ENTER A STRING ";A$
90 INPUT "ENTER X-FACTOR ";X
100 INPUT "ENTER Y-FACTOR ";Y
110 CLS : FOR N=1 TO LEN(A$)
120 LET T$=MID$(A$,N,1): IF T$<"A" OR T$>"Z" THEN NEXT N: GOTO 80
130 LET J=(10*(N-1)*X)+X*A((ASC(T$)-64),1,1): LET K=10+Y*A(ASC(T$)-64,1,2)
140 FOR M=2 TO N(ASC(T$)-64)
150 LET F=X*A((ASC(T$)-64),M,1)
160 LET G=Y*A((ASC(T$)-64),M,2)
170 LINE J,K,J+F,G+K
180 LET J=J+F: LET K=K+G
190 NEXT M: NEXT N
200 IF INKEY$="" THEN GOTO 200
210 GOTO 80
1000 DATA 8,0,0,0,5,1,1,4,0,1,-1,0,-5,0,3,-6,0
1010 DATA  12,0,0,0,6,5,0,1,-1,0,-1,-1,-1,-5,0,5,0,1,-1,0,-1,-1,-1,-5,0
1020 DATA 8,6,1,-1,-1,-4,0,-1,1,0,4,1,1,4,0,1,-1
1030 DATA 7,0,0,0,6,4,0,2,-2,0,-2,-2,-2,-4,0
1040 DATA  7,6,0,-6,0,0,6,6,0,-6,0,0,-3,5,0
1050 DATA 6,0,0,0,6,6,0,-6,0,0,-3,5,0
1060 DATA  10,5,2,1,0,0,-1,-1,-1,-4,0,-1,1,0,4,1,1,4,0,1,-1
1070 DATA 6,0,0,0,6,0,-3,6,0,0,3,0,-6
1080 DATA 6,0,0,6,0,-3,0,0,6,-3,0,6,0
1090 DATA 5,0,1,1,-1,4,0,1,1,0,5
1100 DATA 6,0,0,0,6,0,-3,6,3,-6,-3,6,-3
1110 DATA 3,6,0,-6,0,0,6
1120 DATA  5,0,0,0,6,3,-3,3,3,0,-6
1130 DATA    4,0,0,0,6,6,-6,0,6
1140 DATA 9,1,0,-1,1,0,4,1,1,4,0,1,-1,0,-4,-1,-1,-4,0
1150 DATA 7,0,0,0,6,5,0,1,-1,0,-1,-1,-1,-5,0
1160 DATA 9,4,2,2,-2,-5,0,-1,1,0,4,1,1,4,0,1,-1,0,-5
1170 DATA 9,0,0,0,6,5,0,1,-1,0,-1,-1,-1,-5,0,4,0,2,-3
1180 DATA 12,0,1,1,-1,4,0,1,1,0,1,-1,1,-4,0,-1,1,0,1,1,1,4,0,1,-1
1190 DATA 4,3,0,0,6,-3,0,6,0
1200 DATA 6,0,6,0,-5,1,-1,4,0,1,1,0,5
1210 DATA 3,0,6,3,-6,3,6
1220 DATA 5,0,6,1,-6,2,3,2,-3,1,6
1230 DATA 5,0,0,6,6,-3,-3,-3,3,6,-6
1240 DATA 5,3,0,0,3,-3,3,3,-3,3,3
1250 DATA  4,6,0,-6,0,6,6,-6,0
```

LEAST SQUARES

```
10 LET A$="INKEY$"
100 DIM Z$(3)
110 CSR 1,0: INPUT "HOW MANY DATA POINTS DO YOU HAVE ";H
115 LET B=H
120 IF B<>INT(B) THEN PRINT "ONLY INTEGER VALUES ALLOWED ": PAUSE 1000: GOTO 160
130 IF B<4 THEN PRINT "NOT ENOUGH DATA POINTS FOR LEAST SQUARES ": PAUSE 1000: GOTO 910
140 DIM X(B): DIM Y(B)
150 READ CORR,SX,SY,SX2,SY2,SXY,ERROR,P
160 DATA  1,0,0,0,0,0,0,0
170 CLS
180 CSR 1,0: PRINT "METHOD OF DATA ENTRY"
190 CSR 1,3: PRINT "INPUT X-VALUE <RET>;THEN Y-VALUE <RET>"
200 CSR 1,6: PRINT "CORRECTION OF DATA -SEE LATER"
210 FOR I=1 TO B: CSR 1,8: INPUT "X-VALUE ";X(I): INPUT "Y-VALUE ";Y(I): NEXT I
220 CLS : LET D=1
230 CSR 5,0: PRINT "LINE": CSR 13,0: PRINT "X-VALUE": CSR 21,0: PRINT "Y-VALUE"
240 FOR I=1 TO B: CSR 5,D: PRINT I: CSR 13,D: PRINT X(I): CSR 21,D: PRINT Y(I): LET D=D+1: NEXT
I
260 CSR 1,20: INPUT "DO YOU REQUIRE TO MAKE ANY CHANGES? (Y/N) ";A$
270 IF A$="Y" THEN GOTO 300
280 IF A$="N" THEN GOTO 370 ELSE GOTO 260
300 CLS
310 CSR 1,0: INPUT "LINE NUMBER TO BE CHANGED? ";J
320 LET J=INT(J): IF J<1 OR J>B THEN PRINT "THIS IS NOT IN THE RANGE 1 TO ";B: PAUSE 1000: GOTO
300
330 CSR 1,3: INPUT "TYPE THE CORRECT DATA FOR LINE :X-VALUE ";X(J): INPUT "Y-VALUE ";Y(J)
340 CSR 1,7: PRINT "DATA FOR LINE ";J,X(J),Y(J)
344 CSR 1,9: INPUT "PRESS C TO CONTINUE ";A$
345 IF A$="C" THEN GOTO 220 ELSE GOTO 344
370 FOR I=1 TO B: LET SX=SX+X(I): LET SY=SY+Y(I): NEXT I
380 LET AX=SX/B: LET AY=SY/B
390 FOR I=1 TO B
400 LET X9=X(I)-AX: LET Y9=Y(I)-AY
410 LET SX2=SX2+X9*X9: LET SY2=SY2+Y9*Y9
420 LET SXY=SXY+X9*Y9
430 NEXT I
440 IF SX2=0 THEN PRINT "ALL X-COORDINATES ARE THE SAME-CALCULATIONS TERMINATED": PAUSE 1000: GO
TO 910
450 LET SLOPE=SXY/SX2
460 LET INTERCEPT=(SX2*SY-SX*SXY)/(B*SX2)
470 IF SY2=0 THEN GOTO 490
480 LET CORR=SXY/SQR(SX2*SY2)
490 FOR I=1 TO B
500 LET E=(Y(I)-AY)-SLOPE*(X(I)-AX)
510 LET ERROR=ERROR+E*E: NEXT I
520 IF B<30 THEN LET DEV=SQR(ERROR/(B-2)): GOTO 540
530 LET DEV=SQR(ERROR/B)
540 CLS
550 LET YY=0: LET XS=1: LET YS=2
560 CSR 1,0: PRINT "RESULTS OF CALCULATIONS"
570 CSR 1,2: PRINT "SLOPE                        =";SLOPE
575 CSR 1,4: PRINT "CORRELATION COEFFICIENT =";CORR
580 CSR 1,6: PRINT "Y-AXIS INTERCEPT         =";INTERCEPT
590 CSR 1,8: PRINT "STANDARD DEVIATION OF                    POINTS FROM LINE         =";DEV
600 CSR 1,12: PRINT "STANDARD ERROR OF SLOPE=";DEV/SQR(SX2)
610 CSR 1,14: PRINT "STANDARD ERROR                          OF INTERCEPT             =";DEV*SQR((1
/B)+AX*AX/SX2)
```

```
620 CSR 1,20: INPUT "WOULD YOU LIKE TO SEE A PLOT OF THE   DATA? (Y/N) ";A$
630 IF A$="Y" THEN GOTO 660
640 IF A$="N" THEN GOTO 910 ELSE GOTO 620
660 CLS
710 LET XMAX=X(1): LET XMIN=X(1): LET YMAX=Y(1): LET YMIN=Y(1)
720 FOR I=2 TO B
730 IF XMAX<X(I) THEN LET XMAX=X(I)
740 IF XMIN>X(I) THEN LET XMIN=X(I)
750 IF YMAX<Y(I) THEN LET YMAX=Y(I)
760 IF YMIN>Y(I) THEN LET YMIN=Y(I)
770 NEXT I
780 LET XRANGE=XMAX-XMIN: LET YRANGE=YMAX-YMIN
790 LET XS=185/XRANGE: LET YS=110/YRANGE
800 GOSUB 1000
820 FOR I=1 TO B: LET PX=65+(X(I)-XMIN)*XS: LET PY=65+(Y(I)-YMIN)*YS
830 CIRCLE PX,PY+0,2: NEXT I
840 LET Y1=SLOPE*XMIN+INTERCEPT
845 IF SLOPE>0 THEN GOTO 857
850 LINE 64,(Y1-YMIN)*YS+65,252,64
855 GOTO 860
857 LINE 64,(Y1-YMIN)*YS+65,252,176
860 CSR 1,21: PRINT "XMIN=";XMIN: CSR 17,21: PRINT "XMAX=";XMAX
865 CSR 1,22: PRINT "YMIN=";YMIN: CSR 17,22: PRINT "YMAX=";YMAX
870 CSR 1,0: INPUT "PRESS E TO END ";A$
880 IF A$="E" THEN GOTO 910 ELSE GOTO 870
910 VS 5: CLS
920 CSR 1,0: PRINT "COPYRIGHT A.F.WILSON 1984"
930 CSR 1,4: PRINT "LEAST SQUARES PROGRAM"
940 STOP
1000 VS 4: PAPER 1: COLOUR 4,1: INK 15: CLS
1010 COLOUR 3,15: LINE 32,62,255,62: LINE 32,63,255,63: LINE 62,33,62,176: LINE 63,33,63,176
1020 COLOUR 3,2: LET T1=65+110*.75: LET T2=65+110*.5: LET T3=65+110*.25
1030 LINE 59,176,255,176: LINE 59,T1,255,T1: LINE 59,T2,255,T2: LINE 59,T3,255,T3
1040 CSR 3,2: PRINT "YMAX": CSR 3,15: PRINT "YMIN"
1050 COLOUR 3,2: LET T4=64+191*.75: LET T5=64+191*.5: LET T6=64+191*.25
1060 LINE T6,65,T6,176: LINE T5,65,T5,176: LINE T4,65,T4,176: LINE 255,65,255,176
1070 CSR 3,17: PRINT "XMIN": CSR 28,17: PRINT "XMAX"
1080 CSR 1,21: PRINT "INPUT X-AXIS TITLE AT CURSOR(MAX 12)"
1090 CRVS 0,1,14,19,12,1,32
1100 EDITOR B$
1110 VS 4: CSR 1,21: PRINT "
1115 CSR 1,21: PRINT "INPUT Y-AXIS TITLE(MAX 12)"
1120 CRVS 0,1,1,2,1,12,32
1130 EDITOR B$
1140 VS 4: CSR 1,21: PRINT "
1150 RETURN
```

MEMOTEXT

The following program will be of interest to most users. It is entitled "MEMOTEXT" and it is best described as a scrolling version of the MTX's Direct Screen Input Mode. The program is initially set up to allow text to be entered onto an 80 column by 56 row area although this can be varied from within the program. The 40 column by 24 line screen can be thought of as a window onto the text area. The bottom three rows of the screen are used as a message area similar to the Edasm program. Text can be entered with the aid of the functions listed below.

1. Csr right.
2. Csr left.
3. Csr up.
4. Csr down.
5. Home.
6. Cls. (Clears all 80 column * 56 rows)
7. Tab.
8. F2. (Back Tab - used in preference to Ctrl W)
9. Insert.
10. Delete.
11. Eol.
12. Esc I.
13. Esc J.
14. Esc K.
15. Return.
16. F3. (Opposite of Return - Csr moves to end of line above)

In addition the following commands have been added.

17. Esc P. (Sends text to printer - 80 cols * 56 rows only)
18. Esc X. (Exit from program into the Front Panel)
19. Esc C. (Change colours - as per MTX manual)
20. Esc Page. (This command allows different size text areas to be defined subject to the parameters below)

Maximum number of columns	= 240
Maximum number of rows	= 240
Minimum number of columns	= 41
Minimum number of rows	= 21
Maximum text area	= 240 cols * 80 rows or 80 cols * 240 rows or combinations in between.

Values outside of these parameters will not be allowed.

21. Esc Csr right, left, up or down. (This command passes control to a joystick routine which allows extremely fast scrolling. Only five keys are used in this mode i.e. Csr right, left, up, down and home. The home key returns control back to the main routine. No text can be entered during fast scrolling.

23. BRK. (It should be noted that the break key is disabled during the running of the program and cannot be used to exit. Rather, the break key is used to abort all commands (except Esc P - Print). Also, aborting Esc C and Esc Page will cause the default settings to be reinstated ie. 80 columns by 56 rows and white ink on blue paper.

There are two limitations to the program. Firstly the tabulation routines will not work if the number of columns is less than eighty or is not divisible by eight. Secondly, screen wrap around has bee ommited as I consider this feature to be undesirable. (However, this feature can easily be added).

The following information should help anyone contemplating adding to the program.

DE holds current data position.

HL holds current screen position. (The screen starts at hex 1000 in VRAM).

(CHR) holds the ASCII of the last key pressed.

(INS) Bit 0 is insert toggle. Bit 1 is fast scroll toggle.

(TEMP1) holds the data postion of the character displayed in the top left hand corner of the screen.

(TEMP2) temporary storage for HL ie. screen position.

(TEMP3) temporary storage for DE ie. data postion.

(TEMP4) holds data position of the character in column one of the line displayed at the top of the screen (NOt to be confused with TEMP1).

(ROW) and (COL) are self explanatory.

(STOREND) holds the end of the text area plus two blank rows plus one character.

Some useful subroutines are as follows.

CALL LRAM sends screen position held in HL to VDP.

CALL MULT is an 8 bit multiplications routine. (Multiplier is placed in (MPR), multiplicand in (MPD). Result is stored in (RES).

CALL LINE returns with HL holding data position of the character in column one of the current csr line.

CALL XPOS returns with both A and C holding current column position.

CALL YPOS returns with both A and C holding current row position.

CALL DIV is an 8 bit division routine. Divisor is placed in (MPR), dividend in (MPD). Quotient is stored in (ACC1), remainder is (RES1).

CALL DEC allows a decimal number up to 65535 to be input. This is converted to a two byte hex value which is stored at (RES).

CALL SCR3 is used to display messages. All messages should be twenty bytes long.

```
ie. PUSH AF
    PUSH BC
    PUSH HL        - SAVE SCREEN POSITION
    LD BC,MES?     - ? = MESSAGE NUMBER
    CALL SCR3      - DISPLAY MESSAGE
    POP HL         - RETRIEVE SCREEN POSITION
    POP BC
    POP AF
    CALL LRAM      - SEND SCREEN POSITION TO VDP.
```

If any routine is added which affects the main display area it would be wise to observe the following proceedure.

```
ie.  LD (TEMP2),HL  - SAVE SCREEN POSITION.
     LD (TEMP3),DE  - SAVE DATA POSITION.

YOUR ROUTINE

     LD,DE (TEMP3) - RETRIEVE DATA POSITION
     LD,HL(TEMP2)  - RETRIEVE SCREEN POSITION.
     CALL SCROLL   - UPDATE MAIN DISPLAY AREA.
     JP UPDATE     - UPDATE MESSAGE AREA.
```

Lastly it should be noted that every time the program arrives at GETCHR all pushes and pops, calls and returns are balanced; the stack is therefore cleared. AF and BC are free but DE and HL are not. Also, the alternative register is set, in order that they may be freely available.

Finally, hidden away at hex F007 you will find some panel extentions.

Press P to obtain print out.
Press F4 to clear registers.

Decimal/Hexdecimal/Binary covnversion as follows.

Press F1 to enter decimal number.
Press F2 to enter hexdecimal number.
Press F3 to enter binary number.

Values up to 65535 may be entered. Press ESC to continue.

As the stack limit has been lowered the extentions will be saved alongside the main program or, with any other program that is subsequently entered. They will be lost if the reset keys are pressed but will remain intact if only new,d. However, the stack limit will be reset to its original values and so it must be lowered again prior to saving. This is easily accomplished by typing poke 64146,7 and poke 64147,240 as a direct command.

STEVE SCOTT

```
5 CODE

4007            LD SP,£FD48
400A            LD A,32
400C            LD (£FA91),A
400F            LD HL,BREAK
4012            LD (£FD55),HL
4015            LD A,£C3
4017            LD (£FD54),A
401A            LD A,4
401C            LD (£FD5E),A
401F            JP CHECK
4022 X:         DS 1
4023 Y:         DS 1
4024 XX:        DS 1
4025 YY:        DS 1
4026 CHR:       DS 1
4027 INS:       DS 1
4028 TEMP1:     DS 2
402A TEMP2:     DS 2
402C TEMP3:     DS 2
402E TEMP4:     DS 2
4030 STOREND:   DS 2
4032 LASTROW:   DS 2
4034 XTRAROW:   DS 2
4036 ROW:       DB 56
4037 COL:       DB 80
4038 CLEAR:     LD (TEMP3),DE
403C            LD (TEMP2),HL
403F            LD BC,MES2
4042            CALL SCR3
4045            LD A,32
4047            LD (£FA91),A
404A CLEAR1:    CALL £79
404D            JR Z,CLEAR1
404F            CP "y"
4051            JR Z,CLEAR5
4053            CP "n"
4055            JR Z,CLEAR5
4057            JR CLEAR1
4059 CLEAR2:    LD HL,STORE
405C CLEAR3:    LD (HL),32
405E            INC HL
405F            DEC BC
4060            LD A,B
4061            OR C
4062            JR NZ,CLEAR3
4064 CLEAR4:    LD BC,MES7
4067            CALL SCR3
406A            CALL CLS
406D            JR RESET
406F CLEAR5:    LD (CHR),A
4072            LD BC,MES7
4075            CALL SCR3
4078            LD BC,£4CE0
407B            LD A,(CHR)
407E            CP "y"
4080            JR Z,CLEAR2
4082            LD DE,(TEMP3)
4086            LD HL,(TEMP2)
4089            CALL LRAM
408C            JP GETCHR
408F BREAK:     LD SP,£FD48
4092            LD BC,MES23
4095            CALL SCR2
4098            LD BC,MES24
409B            CALL SCR3
409E            CALL DELAY
40A1 BRK1:      CALL £79
40A4            JR Z,BRK1
40A6            JR RESET
40A8 BRK2:      LD A,80
40AA            LD (COL),A
40AD            LD A,56
40AF            LD (ROW),A
40B2            LD A,£F5
40B4            OUT (2),A
40B6            LD A,7
40B8            OR £80
40BA            OUT (2),A
40BC            LD SP,£FD48
40BF            JR RESET
40C1 BRK3:      LD (TEMP2),HL
40C4            LD (TEMP3),DE
40C8 BRK4:      LD BC,MES7
40CB            CALL SCR3
40CE            LD HL,(TEMP2)
40D1            LD DE,(TEMP3)
40D5            CALL LRAM
40D8            LD SP,£FD48
40DB            JP UPDATE
40DE RESET:     LD DE,£1C27
40E1            LD (TEMP2),DE
40E5            LD DE,STORE
40E8            LD (TEMP1),DE
40EC            LD (TEMP3),DE
40F0            LD (TEMP4),DE
40F4            LD A,1
40F6            LD (X),A
40F9            LD (Y),A
40FC            LD (XX),A
40FF            LD (YY),A
4102            LD A,3
4104            LD (INS),A
4107            LD HL,£1C00
410A            CALL LRAM
410D            CALL SCROLL
4110            LD BC,MES7
4113            CALL SCR3
4116            JP UPDATE
4119 GETCHR:    LD A,(INS)
411C            BIT 1,A
411E            JP Z,JOYS
4121            XOR A
4122            CALL £79
4125            JR Z,GETCHR
4127            CP 3
4129            JR Z,BRK3
412B            LD (CHR),A
412E            CP 25
```

```
4130            JR Z,RIGHT
4132            CP 8
4134            JP Z,LEFT
4137            CP 10
4139            JP Z,DOWN
413C            CP 11
413E            JP Z,UP
4141            CP 9
4143            JP Z,TAB
4146            CP 129
4148            JP Z,CTRLW
414B            CP 21
414D            JP Z,INSERT
4150            CP 127
4152            JP Z,DELETE
4155            CP 5
4157            JP Z,EOL
415A            CP 12
415C            JP Z,CLEAR
415F            CP 26
4161            JP Z,RESET
4164            CP 13
4166            JP Z,RETURN
4169            CP 130
416B            JP Z,RET7
416E            CP 27
4170            JP Z,ESCAPE
4173 OUTCHR:    LD A,(INS)
4176            BIT 0,A
4178            JP Z,INS2
417B OUTCHR1:   LD A,(CHR)
417E            CP 25
4180            JR Z,RIGHT
4182            LD (DE),A
4183 RIGHT:     LD A,(X)
4186            CP 40
4188            JR Z,RIGHT1
418A            INC A
418B            LD (X),A
418E            INC HL
418F            JR RIGHT2
4191 RIGHT1:    LD A,(COL)
4194            SUB 39
4196            LD B,A
4197            LD A,(XX)
419A            CP B
419B            JP Z,GETCHR
419E            INC A
419F            LD (XX),A
41A2            LD BC,(TEMP1)
41A6            INC BC
41A7            LD (TEMP1),BC
41AB RIGHT2:    INC DE
41AC            CALL SCROLL
41AF            JP UPDATE
41B2 LEFT:      LD A,(X)
41B5            CP 1
41B7            JR Z,LEFT1
41B9            DEC A
41BA            LD (X),A
41BD            DEC HL
41BE            JR LEFT2
41C0 LEFT1:     LD A,(XX)
41C3            CP 1
41C5            JP Z,GETCHR
41C8            DEC A
41C9            LD (XX),A
41CC            LD BC,(TEMP1)
41D0            DEC BC
41D1            LD (TEMP1),BC
41D5 LEFT2:     DEC DE
41D6            CALL SCROLL
41D9            JP UPDATE
41DC DOWN:      LD A,(COL)
41DF            LD C,A
41E0            LD B,0
41E2            PUSH BC
41E3            LD A,(Y)
41E6            CP 21
41E8            JR Z,DOWN1
41EA            INC A
41EB            LD (Y),A
41EE            LD BC,£0028
41F1            ADD HL,BC
41F2            JR DOWN2
41F4 DOWN1:     LD A,(ROW)
41F7            SUB 20
41F9            LD B,A
41FA            LD A,(YY)
41FD            CP B
41FE            JP Z,STACK
4201            INC A
4202            LD (YY),A
4205            PUSH HL
4206            LD HL,(TEMP1)
4209            LD B,0
420B            ADD HL,BC
420C            LD (TEMP1),HL
420F            LD HL,(TEMP4)
4212            ADD HL,BC
4213            LD (TEMP4),HL
4216            POP HL
4217 DOWN2:     EX DE,HL
4218            POP BC
4219            ADD HL,BC
421A            EX DE,HL
421B            CALL SCROLL
421E            JP UPDATE
4221 UP:        LD A,(COL)
4224            LD C,A
4225            LD B,0
4227            PUSH BC
4228            LD A,(Y)
422B            CP 1
422D            JR Z,UP1
422F            DEC A
4230            LD (Y),A
4233            LD BC,£0028
4236            XOR A
4237            SBC HL,BC
```

```
4239            JR UP2
423B UP1:       LD A,(YY)
423E            CP 1
4240            JP Z,STACK
4243            DEC A
4244            LD (YY),A
4247            PUSH HL
4248            LD HL,(TEMP1)
424B            XOR A
424C            SBC HL,BC
424E            LD (TEMP1),HL
4251            LD HL,(TEMP4)
4254            XOR A
4255            SBC HL,BC
4257            LD (TEMP4),HL
425A            POP HL
425B UP2:       EX DE,HL
425C            POP BC
425D            XOR A
425E            SBC HL,BC
4260            EX DE,HL
4261            CALL SCROLL
4264            JP UPDATE
4267 SCROLL:    LD (TEMP2),HL
426A            LD (TEMP3),DE
426E            DI
426F            LD HL,£1C00
4272            CALL LRAM
4275            LD HL,(TEMP1)
4278            LD A,(COL)
427B            SUB 40
427D            LD E,A
427E            LD D,0
4280            LD A,21
4282            LD C,1
4284 SCROLL1:   LD B,40
4286            OTIR
4288            DEC A
4289            JR Z,SCROLL2
428B            ADD HL,DE
428C            JR SCROLL1
428E SCROLL2:   EI
428F            LD DE,(TEMP3)
4293            LD HL,(TEMP2)
4296            CALL LRAM
4299            CALL CSR
429C            CALL LRAM
429F            RET
42A0 UPDATE:    LD HL,MES1
42A3            LD BC,£05
42A6            ADD HL,BC
42A7            CALL YPOS
42AA            CALL UPD
42AD            LD BC,£0A
42B0            ADD HL,BC
42B1            CALL XPOS
42B4            CALL UPD
42B7            CALL SCREEN
42BA            LD DE,(TEMP3)
42BE            LD HL,(TEMP2)
42C1            CALL LRAM
42C4            CALL CSR
42C7            CALL LRAM
42CA            JP GETCHR
42CD UPD:       LD A,100
42CF            LD E,A
42D0            CALL UPD1
42D3            LD A,10
42D5            LD E,A
42D6            CALL UPD1
42D9            LD A,C
42DA            ADD A,48
42DC            LD (HL),A
42DD            RET
42DE UPD1:      LD B,0
42E0 UPD2:      CP C
42E1            JR Z,UPD3
42E3            JR NC,UPD4
42E5 UPD3:      INC B
42E6            CP 200
42E8            JR Z,UPD5
42EA            ADD A,E
42EB            JR UPD2
42ED UPD4:      SUB E
42EE UPD5:      LD E,A
42EF            LD A,C
42F0            SUB E
42F1            LD C,A
42F2            LD A,B
42F3            ADD A,48
42F5            LD (HL),A
42F6            INC HL
42F7            RET
42F8 LRAM:      PUSH AF
42F9            LD A,L
42FA            OUT (2),A
42FC            LD A,H
42FD            OR £40
42FF            OUT (2),A
4301            POP AF
4302            RET
4303 CLS:       LD BC,840
4306            LD HL,£1C00
4309            CALL LRAM
430C CLS1:      LD A,£20
430E            OUT (1),A
4310            DEC BC
4311            LD A,C
4312            OR B
4313            JR NZ,CLS1
4315            CALL LRAM
4318 CSR:       LD A,127
431A            OUT (1),A
431C            XOR A
431D            RET
431E MULT:      PUSH AF
431F            PUSH BC
4320            PUSH DE
4321            PUSH HL
4322            LD A,(MPR)
```

Volume Three **MEMOPAD** Number 8

CONTENTS

What's In It For You!

NOTICE TO ALL MEMBERS

A RENEWAL SLIP WILL BE ENCLOSED WITH THE MAGAZINE OF ANYONE WHO IS DUE TO RENEW THEIR SUBSCRIPTION. PLEASE FILL THIS IN AND RETURN IT TO US AS SOON AS POSSIBLE SO THAT YOU DO NOT MISS AN ISSUE OF THE MAGAZINE.

Number 8

MEMOPAD

Volume Three

HIGH SCORES

ASTRO PAC	213,430	Michael Hunt
KNUCKLES	1,147,360	T. Erikson
CHAMBERIODS	Comp. 4 mins	T. Erikson
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
OBLOIDS	62,400	M. Hurley
PHAID	26,000	Sally Street
KILOPEDE	82,253	Richard Nash
3D TACHYON FIGHTER	12,500	C. Walker
CONTINENTAL RAIDERS	106,240	Sean Haverty
BLOBBO	148,283	E. Mahon
QUANTUM	14	Ian Cartwright
QOGO 2	205,000	R. Siddall
MINEFIELD	2,100	C. Walker
FLUMMOX	251,510	C. Walker
TURBO	18,610	Michael Hunt
FATHOMS DEEP	3,450	Matthew Moss
AGROVATOR	675,000	P. Howard
FIREHOUSE FREDDIE	29,620	T. Erikson
QOGO	43,960	T. Erikson
ARCADIANS	25,900	Adrian Johnson
MISSILE COMMAND	27,580	Adrian Johnson
LITTLE DEVILS	34,320	Leslie Banks
FELIX IN THE FACTORY	14,740	Peter Crighton
HUNCHY	8,457	John Quin
SON OF PETE	17,233	T. Erikson
HAWKWARS	25,800	Gordon Hurd
ESCAPE FROM ZARCOS	76 Items	G. Hill
SALTY SAM	40,642	Andrew Johnson
MISSION OMEGA	10,850	A. Knott & S. Paine
ICEBURG	17,431	Alan Dobson
SNOWBALL	1,000	Victor Stepney
EMERALD ISLE	768/1000	Victor Stepney
SUPERBIKE	23.9kms	A. 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	A. Miller
SURFACE SCANNER	72,060	T. Erikson
CAVES OF ORB	496/500	V. Stepney
SEPULCRI SCALERATI	8,000	Andrew Miller
SMG	105,400	Clare Townsend
RETURN TO EDEN	1,000	Andy Crick
QUAZZIA	26,660	Andrew Miller
OBLITERATION ZONE	32,670	Alan Dobson
ASTORMILLION	142,342	D.J. Chamberlain
CRYSTAL	32,425	Gordon Hurd (COMP)
DRIVE THE CEE?5	12,907	V. Stepney
HIGHWAY ENCOUNTER	123,120	
KARATE KING	4,570	G. Hill
DOWNSTREAM DANGER	8,976	G. Hill
DOODLEBUGS	4,340	A. Miller
THE WALL	53,500	A. Miller
COMBAT	47,690	A. Miller

Pascal Indenter

INTRODUCTION

This program is a utility to indent Pascal source programs to a 'standard' format to improve their presentation and readability. This utility was written so that programs written on the Memotech in Hisoft Pascal could be quickly made acceptably indented to be presented at university, without laboriously padding the program with spaces. The utility does not indent everything - RECORD statements and complex IF statements must still be done manually. The program was written in machine code for speed and also because the compiler does not allow a program to alter it's own source text. The machine code program is given so that improvments can be made by anyone who wishes to, eg. to make it indent RECORD's, IF..THEN..ELSE statements. Only the assembled code Pascal version is needed to use the utility.

USER INSTRUCTIONS

The program is simple since the user intervention is not required while the utility program is running. The utility should be loaded after the program under development. This is done by setting the marker (press M) at the end of the program to be indented and loading the utility as normal. To run the utility simply set the marker to the beginning of the utility and compile (press C) then enter 'Y' to run. The program will return control to Basic when the indenting is complete. Reset the computer using the two keys by the space bar. Enter the Pascal compiler and press ESC until SE.B error occurs when asked for a table. Enter the compiler again and the indented program should be intact but indented. If some lines appear to have been removed, perhaps due to fewer END's than BEGIN's, then check that they are not just indented so far that they are of the screen! Use the FIND and REPLACE functions to speed up entering the program. Create one line of INLINE (# ,# ,#); as a replacement for Z for instance then simply insert the program heading and 20 'Z's on separate lines and use the global replace to create 20 INLINE statements into which the assembled code can be typed.

HOW THE PROGRAM WORKS

The line structure in Pascal consists of a byte containing the spaces for that line and then the token for the first reserved word in the line or the ASCII code for the first character. The program searches for the carriage returns (OD hex) and then the second byte of the next line. It tests the contents for being a reserved word and indents appropriately, keeping a count of the number of spaces to indent by. The program returns control to Basic so that the source code remains indented and is not restored by the compiler. The machine has to be reset because, although the program is present without resetting, it will not compile because the compiler has been upset by the jump to Basic. The machine code program is commented to increase transparency and is totally relocatable code so the position of the assembled code does not matter, any length of source program can therefore be indented. A modified version of an already published panel dump routine was used to obtain a hard copy of the assembled code (PCW June 1985 page 213). The Pascal INLINE statement was used to interface the object code with the Pascal program.

The machine code program uses the registers as follows: DE as two separate registers - D holds the number of spaces indented in the last line, E holds the number of BEGINs currently not ENDed and bit 7 is used as a flag to indicate variable definitions. HL points to the memory locations being examined BC is used to create an indefinate loop until an 'END.' is reached. B is also used as a flag to indicate that a reserved word has been found and the spaces count must be increased for the next line. AF is used for all comparisons. The program initialised the flags and the start address (8000hex) and the count is set to zero. The PROGRAM and heading is ignored as are comments in '()' but '(**)' comments are indented. The program searches for an occurance of the end of line character, EOLN (ODhex). HL is automatically incremented to point to the second byte of the line following the EOLN and the contents are loaded into A. They are then compared with the tokens for the reserved words. If a defining word is found such as TYPE then it is indented by a fixed amount (=2 spaces). If a PROCEDURE or FUNCTION definition is found then flag bit 7 of E is set so that the variable definitions and the BEGINs and ENDs of the PROCEDUREs are indented. The first BEGIN of the main program is not indented. This is achieved by resetting the flag after a function or procedure begin has been encountered. Note that the reserved word is only indented if it is the first in the line excepting spaces. FOUND increments the count of spaces after indenting the current line so that the following line is indented three spaces more. DECNT decrements the count of spaces before indenting the current line so that it is indented three spaces less than the previous line and level with the one following. TEST tests for the end of a program by testing for 'END.'. DEFN sets the spaces for the global and local variable definitions. Note: This does not support nested procedures in that they are all indented as if they were global. POKE alters the contents of the byte for the number of spaces and tests for a reserved word. The number of spaces is increased if a reserved word was found, otherwise the search is continued for the next line to be indented. TEMPJR is a temporary jump because the largest possible relative jump is exceeded because of the length of the program eg. 416C could be JR NZ, TEMPJR. The next line is then no longer neccessary. The program works satisfactorily so tidying up the code is probably not worth the effort required. Care must be taken when typing in the program - it must agree with the listing given. The program may self destruct if typed in incorrectly!

PASCAL TOKENS

These are the tokens for the reserved Pascal words.

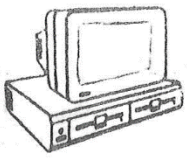
Hex value	Reserved word	Hex value	Reserved word
OD	END OF LINE (= RET)	92	ELSE
81	PROGRAM	93	REPEAT
82	DIV	94	CASE
83	CONST	95	WHILE
84	PROCEDURE	97	IF
85	FUNCTION	98	BEGIN
86	NOT	99	WITH
87	OR	9A	GOTO
88	AND	9B	SET
89	MOD	9C	ARRAY
8A	VAR	9D	FORWARD
8B	OF	9E	RECORD

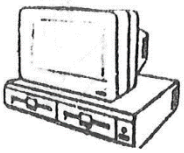
8C	TO	9F	TYPE
8D	DOWNTO	A0	IN
8E	THEN	A1	LABEL
8F	UNTIL	A2	NIL
90	END	A3	PACKED
91	DO		

```
0 REM SOURCE CODE FOR PASCAL INDENTER
1 REM
2 REM For use with MTX Hisoft Pascal
3 REM
4 REM Written by J.R.Stopforth 06/08/86
5 REM
6 REM The assembled code is to indent
7 REM
8 REM the Pascal.source code
9 REM
10 CODE

40DB           LD E,0      ;initialise BEGIN count/PROCEDURE-FUNCTION flag
40DD           LD D,0      ;initialise spaces counter
40DF           LD HL,#8000         ;pascal start
40E2 SEARCH:   LD BC,#FFFF         ;largest HEX number !
40E5           LD A,#0D    ;code for EOLN (RET)
40E7           CPIR        ;find EOLN
40E9           INC HL      ;point to byte after EOLN
40EA           LD A,(HL)   ;prepare to check it's contents for Pascal reserved word
s
40EB           CP #7B      ;ASCII for  '{'
40ED           JR Z,SEARCH         ;comments not indented
40EF           CP #A1      ;token for  LABEL
40F1           JR Z,DEFN   ;found reserved defining word
40F3           CP #83      ;token for CONST
40F5           JR Z,DEFN   ;so indent as a defining word
40F7           JR TYPE     ;skip next instruction
40F9 TEMPJR:   INC HL      ;pointer set beyond spaces
40FA           JR SEARCH   ;jr too large otherwise!
40FC TYPE:     CP #9F      ;token for TYPE
40FE           JR Z,DEFN
4100           CP #8A      ;token for VAR
4102           JR Z,DEFN
4104           CP #84      ;token for PROCEDURE
4106           JR NZ,FNTN  ;not PROCEDURE
4108           SET 7,E     ;set flag for PROCEDURE
410A           LD D,2      ;no. of spaces to indent PROCEDURE
410C           JR POKE     ;poke no. of spaces
410E FNTN:     CP #85      ;token for FUNCTION
4110           JR NZ,BEGIN         ;not a FUNCTION
4112           SET 7,E     ;set flag for FUNCTION
4114           LD D,2      ;no. of spaces to indent FUNCTION
4116           JR POKE     ;insert spaces
4118 BEGIN:    CP #98      ;token for BEGIN
411A           JR NZ,REPEAT        ;not a BEGIN
411C           LD A,E      ;access flag/BEGIN counter
411D           INC E       ;increment BEGIN count
411E           CP 0        ;is previous count/flag = 0 ?
4120           JR Z,COUNT  ;if flags/count =0 then set spaces counter for first BEG
IN
4122           BIT 7,E     ;test flag condition ( counter <> 0 )
4124           JR Z,FOUND  ;main program BEGIN
4126           RES 7,E     ;reset flag
4128           LD D,2      ;no. of spaces for PROCEDURE/FUNCTION BEGIN
412A           JR FOUND    ;found reserved word
412C COUNT:    LD D,0      ;set spaces count for first main program BEGIN
412E           JR FOUND
```

```
4130 REPEAT: CP #93      ;token for REPEAT
4132         JR Z,FOUND
4134         CP #94      ;token for CASE
4136         JR Z,FOUND
4138         CP #92      ;token for ELSE
413A         JR Z,DECNT  ;decrement no. spaces counter
413C         CP #8F      ;token for UNTIL
413E         JR Z,DECNT  ;dec count + indent
4140         CP #90      ;token for END
4142         JR NZ,POKE  ;not a resword so indent to previous no. of spaces
4144         LD A,0      ;test BEGIN count
4146         CP E        ;for count=0 and
4147         JR Z,TEST   ;if count <> 0
4149         DEC E       ;then decrement the count due to finding an END
414A TEST:   INC HL      ;move pointer on one byte
414B         LD A,#2E    ;ASCII for '.'
414D         CP (HL)     ;test for '.'
414E         JR Z,STOP   ;PASCAL EOF (END.)
4150         DEC HL      ;restore pointer
4151 DECNT:  DEC D       ;dec count
4152         DEC D       ;dec count
4153         DEC D       ;dec count
4154         JR POKE     ;indent line
4156 FOUND:  LD B,0      ;set found flag
4158         JR POKE
415A DEFN:   LD B,0      ;set found flag
415C         LD A,0      ;set test value
415E         CP E        ;test=0 if E=0 and
415F         JR Z,GLOBALS        ;variables are global
4161         LD D,4      ;spaces for local variables
4163         JR POKE
4165 GLOBALS:LD D,2      ;spaces for global variables
4167 POKE:   DEC HL      ;point to byte containing the no. of spaces
4168         LD (HL),D   ;poke spaces
4169         LD A,0      ;test found flag
416B         CP B        ;flag reset for a reserved word
416C         JR Z,RESWORD        ;B=0 for resword
416E         JR TEMPJR   ;to jump to search:next line
4170 RESWORD:INC D       ;resword:inc count
4171         INC D       ;inc count
4172         INC D       ;inc count
4173         JR TEMPJR   ;tempory jump to get to search & next line
4175 STOP:   RET         ;reached end of PASCAL
4176         RET

Symbols:
POKE     4167    SEARCH   40E2
DEFN     415A    FOUND    4156
STOP     4175    DECNT    4151
RESWORD  4170    REPEAT   4130
FNTN     410E    BEGIN    4118
COUNT    412C    TEST     414A
GLOBALS  4165    TEMPJR   40F9
TYPE     40FC

11 STOP
12 REM   PANEL EXPANSION UTILITY
13 REM
14 REM   From PCW JUNE 1985
15 REM
16 REM   Anonymous * 23/10/85
17 REM
18 REM   Modifiied by J.R.Stopforth
19 REM
20 REM   Prints lower panel screen
21 CODE

4C1A START:  LD A,#C3   ;jump code for panel extension.
4C1C         LD (#FA9E),A
4C1F         LD HL,PANEL        ;Address for panel extension.
4C22 FINISH: LD (#FA9F),HL      ;Set fexpand.
```

```
4C25 PANEL:  CP #50      ;Is key P.
4C27         RET NZ      ;If so then continue.
4C28         LD A,6      ;Screen lines.
4C2A         PUSH AF
4C2B         LD HL,#1C00         ;Start of name table in VRAM.
4C2E         LD DE,40    ;Skip to next screen line.
4C31         LD B,17     ;Number of lines to skip.
4C33 SKIP:   ADD HL,DE   ;Skip the line.
4C34         DJNZ SKIP
4C36 STRT:   CALL LADDR  ;Set up address for VRAM read.
4C39         CALL DUMP   ;Print out one line.
4C3C         POP AF
4C3D         DEC A       ;Keep line countdown.
4C3E         CP 0        ;Has it finished.
4C40         RET Z       ;finish routine after     lines.
4C41         PUSH AF
4C42         ADD HL,DE   ;Address of next screen line.
4C43         JR STRT     ;Do until finished.
4C45 LADDR:  LD A,L      ;Set up VRAM address.
4C46         OUT (2),A   ;LS byte first.
4C48         LD A,H      ;Now set MS byte.
4C49         OUT (2),A
4C4B         CALL WAIT   ;Timing pause for VDP.
4C4E         RET
4C4F DUMP:   LD B,40     ;Set screen width for dump.
4C51 LOOP1:  IN A,(1)    ;Read screen character.
4C53         CALL WAIT   ;Pause.
4C56         CALL BUFFER         ;Load printer buffer.
4C59         DJNZ LOOP1 ;Get next character.
4C5B         CALL LPRINT         ;LPRINT one line of the screen.
4C5E         RET
4C5F BUFFER: OUT (4),A   ;Latch char into printer buffer.
4C61 STATUS: IN A,(4)    ;Is PTR ready.
4C63         AND #1      ;Check ready bit.
4C65         JR NZ,STATUS
4C67         CALL WAIT   ;Pause for printer.
4C6A         IN A,(0)    ;Strobe data into printer buffer.
4C6C         CALL WAIT   ;Pause
4C6F         IN A,(4)    ;Reset strobe signal.
4C71         RET
4C72 LPRINT: LD A,10     ;Line feed.
4C74         CALL BUFFER         ;Send to printer.
4C77         LD A,13     ;Carriage return.
4C79         CALL BUFFER         ;Send it.
4C7C         RET
4C7D WAIT:   PUSH BC     ;Pause for a period
4C7E         LD B,50     ;of time to allow
4C80 LOOP2:  DJNZ LOOP2 ;the printer and the
4C82         POP BC      ;VRAM to respond.
4C83         RET

Symbols:
START   4C1A     PANEL   4C25
FINISH  4C22     LADDR   4C45
STRT    4C36     DUMP    4C4F
WAIT    4C7D     LOOP1   4C51
BUFFER  4C5F     LPRINT  4C72
STATUS  4C61     LOOP2   4C80
SKIP    4C33

22 STOP
```





MEMOTEXT

(PART TWO)

```
4325            LD H,A
4326            LD L,0
4328            LD DE,(MPD)
432C            LD D,0
432E            LD B,8
4330            XOR A
4331 MULT1:     ADD HL,HL
4332            JR NC,NOADD
4334            ADD HL,DE
4335 NOADD:     DJNZ MULT1
4337            LD (RES),HL
433A            POP HL
433B            POP DE
433C            POP BC
433D            POP AF
433E            RET
433F MPR:       DS 1
4340 MPD:       DS 1
4341 RES:       DS 2
4343 RES1:      DS 1
4344 ACC1:      DS 1
4345 ACC2:      DS 2
4347 EMPT:      DB 0,0,0,0,0
434C BUFF:      DS 5
4351 VAL1:      DB 6,5,5,3,6
4356 DIV:       PUSH AF
4357            PUSH BC
4358            PUSH DE
4359            PUSH HL
435A            LD A,(MPR)
435D            LD C,A
435E            LD A,(MPD)
4361            LD E,A
4362            XOR A
4363            LD B,8
4365 DIV1:      RL E
4367            RLA
4368            SUB C
4369            JR NC,DIV2
436B            ADD A,C
436C DIV2:      DJNZ DIV1
436E            LD B,A
436F            LD A,E
4370            RLA
4371            CPL
4372            LD (ACC1),A
4375            LD A,B
4376            LD (RES1),A
4379            POP HL
437A            POP DE
437B            POP BC
437C            POP AF
437D            RET
437E DEC:       PUSH AF
437F            PUSH BC
4380            PUSH DE
4381            PUSH HL
4382            LD (TEMP2),HL
4385 DEC1:      CALL LRAM
4388            LD DE,BUFF
438B            LD B,5
438D DEC2:      CALL £79
4390            JR Z,DEC2
4392            CP 3
4394            JP Z,BRK2
4397            CP 13
4399            JR Z,DEC4
439B            CP 48
439D            JR C,DEC2
439F            CP 58
43A1            JR C,DEC3
43A3            JR DEC2
43A5 DEC3:      OUT (1),A
43A7            SUB £30
43A9            LD (DE),A
43AA            INC DE
43AB            CALL CSR
43AE            INC HL
43AF            CALL LRAM
43B2            DJNZ DEC2
43B4 DEC4:      LD A,32
43B6            OUT (1),A
43B8            LD BC,£0005
43BB            EX DE,HL
43BC            DEC HL
43BD            LD DE,VAL1
43C0            DEC DE
43C1            LDDR
43C3 DEC5:      LD DE,BUFF
43C6            LD HL,VAL1
43C9            LD B,5
43CB DEC6:      LD A,(DE)
43CC            CP (HL)
43CD            JR Z,DEC7
43CF            JP M,OKN
43D2            JP P,HIGH
43D5 DEC7:      INC DE
43D6            INC HL
43D7            DJNZ DEC6
```

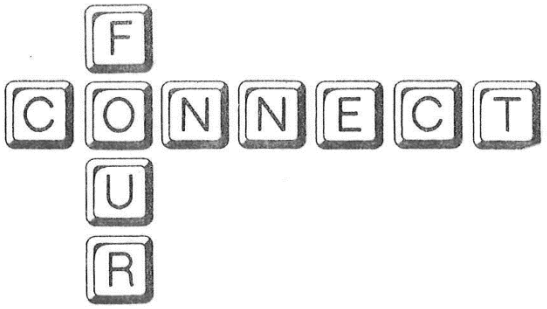
```
43D9 HIGH:    LD HL,(TEMP2)
43DC          CALL LRAM
43DF          LD B,5
43E1          LD A,32
43E3 HIGH2:   OUT (1),A
43E5          DJNZ HIGH2
43E7          LD HL,(TEMP2)
43EA          CALL LRAM
43ED          CALL CSR
43F0          JP DEC1
43F3 OKN:     LD A,10
43F5          LD C,A
43F6          LD B,5
43F8          LD HL,BUFF
43FB          LD DE,£00
43FE DEC8:    PUSH BC
43FF          PUSH HL
4400          PUSH HL
4401          LD HL,£00
4404          LD B,8
4406 MULT2:   SRL C
4408          JR NC,MULT3
440A          ADD HL,DE
440B MULT3:   SLA E
440D          RL D
440F          DJNZ MULT2
4411          EX DE,HL
4412          POP HL
4413          POP BC
4414          LD L,(HL)
4415          LD H,0
4417          ADD HL,DE
4418          EX DE,HL
4419          INC BC
441A          PUSH BC
441B          POP HL
441C          POP BC
441D          DJNZ DEC8
441F          LD (RES),DE
4423          POP HL
4424          POP DE
4425          POP BC
4426          POP AF
4427          RET
4428 LINE:    LD (TEMP2),HL
442B          LD (TEMP3),DE
442F          LD A,(Y)
4432          DEC A
4433          LD C,A
4434          LD A,(YY)
4437          DEC A
4438          ADD A,C
4439          LD (MPR),A
443C          LD A,(COL)
443F          LD (MPD),A
4442          CALL MULT
4445          LD DE,(RES)
4449          LD HL,STORE
444C          ADD HL,DE
444D          RET
444E TAB:     CALL OKTAB
4451          LD A,(X)
4454          CALL TAB7
4457 TAB1:    LD B,A
4458          LD A,C
4459          ADD A,8
445B          LD (X),A
445E          SUB B
445F          LD C,A
4460          LD B,0
4462          EX DE,HL
4463          ADD HL,BC
4464          EX DE,HL
4465          ADD HL,BC
4466          LD A,(X)
4469          CP 41
446B          JR Z,TAB2
446D          CALL SCROLL
4470          JP UPDATE
4473 TAB2:    LD A,40
4475          LD (X),A
4478          DEC HL
4479          LD (TEMP2),HL
447C          LD A,(XX)
447F          CP 1
4481          JR Z,TAB5
4483          CALL TAB7
4486          LD A,C
4487          ADD A,8
4489          PUSH AF
448A          LD A,(COL)
448D          SUB 39
448F          LD B,A
4490          POP AF
4491          CP B
4492          JR NZ,TAB3
4494          DEC A
4495 TAB3:    LD (XX),A
4498          LD HL,(TEMP4)
449B          LD C,A
449C          LD B,0
449E          ADD HL,BC
449F          LD (TEMP1),HL
44A2          LD BC,39
44A5 TAB4:    LD A,(Y)
44A8          DEC A
44A9          LD (MPR),A
44AC          LD A,(COL)
44AF          LD (MPD),A
44B2          CALL MULT
44B5          LD DE,(RES)
44B9          ADD HL,DE
44BA          ADD HL,BC
44BB          EX DE,HL
44BC          LD HL,(TEMP2)
44BF          LD A,(XX)
44C2          JR TAB6
44C4 TAB5:    LD BC,(TEMP1)
```

```
44C8            INC BC
44C9            LD (TEMP1),BC
44CD TAB6:      INC 'A
44CE            LD (XX),A
44D1            CALL SCROLL
44D4            JP UPDATE
44D7 TAB7:      PUSH AF
44D8            LD C,33
44DA            LD A,(COL)
44DD            CP 80
44DF            JR C,TAB8
44E1            SUB 47
44E3            LD C,A
44E4 TAB8:      POP AF
44E5 TAB9:      CP C
44E6            RET NC
44E7            LD B,8
44E9 TAB10:     DEC C
44EA            DJNZ TAB10
44EC            JR TAB9
44EE OKTAB:     LD A,(COL)
44F1            LD (MPD),A
44F4            LD A,8

44F6            LD (MPR),A
44F9            CALL DIV
44FC            LD A,(RES1)
44FF            CP 0
4501            JR NZ,NOTAB
4503            LD A,(COL)
4506            CP 80
4508            JR C,NOTAB
450A            RET
450B NOTAB:     POP BC
450C            PUSH HL
450D            LD BC,MES12
4510            CALL SCR3
4513            POP HL
4514            CALL LRAM
4517 STACK:     LD SP,£FD48
451A            JP GETCHR
451D CTRLW:     CALL OKTAB
4520            LD A,(X)
4523            CALL TAB7
4526 CW1:       LD B,A
4527            LD A,C
4528            CP B
4529            JR NZ,CW2
452B            SUB 8
452D CW2:       LD (X),A
4530            CP 1
4532            JP P,CW3
4535            LD A,1
4537            LD (X),A
453A            JR CW4
453C CW3:       LD C,A
453D            LD A,B
453E            SUB C
453F            LD C,A

4540            LD B,0
4542            EX DE,HL
4543            XOR A
4544            SBC HL,BC
4546            EX DE,HL
4547            XOR A
4548            SBC HL,BC
454A            CALL SCROLL
454D            JP UPDATE
4550 CW4:       LD (TEMP2),HL
4553            LD A,(XX)
4556            CP 1
4558            JR Z,CW7
455A            CALL TAB7
455D            LD B,A
455E            DEC B
455F CW5:       DEC C
4560            LD A,C
4561            CP B
4562            JR NZ,CW6
4564            SUB 8
4566            LD C,A
4567 CW6:       LD (XX),A
456A            LD B,0
456C            LD HL,(TEMP4)

456F            ADD HL,BC
4570            LD (TEMP1),HL
4573            LD BC,£0000
4576            JP TAB4
4579 CW7:       CALL SCROLL
457C            JP UPDATE
457F RETURN:    LD A,(X)
4582            LD C,A
4583            DEC C
4584            LD B,0
4586            XOR A
4587            SBC HL,BC
4589            LD BC,£0028
458C            ADD HL,BC
458D            LD (TEMP2),HL
4590 RET2:      LD A,(Y)
4593            CP 21
4595            JR Z,RET4
4597            INC A
4598            LD (Y),A
459B RET3:      LD HL,(TEMP4)
459E            LD (TEMP1),HL
45A1            DEC A
45A2            LD (MPR),A
45A5            LD A,(COL)
45A8            LD (MPD),A
45AB            CALL MULT
45AE            LD DE,(RES)
45B2            ADD HL,DE
45B3            EX DE,HL
45B4            LD A,1
45B6            LD (X),A
45B9            LD (XX),A
```

```
45BC            LD HL,(TEMP2)
45BF            CALL SCROLL
45C2            JP UPDATE
45C5 RET4:      LD HL,(TEMP2)
45C8            LD BC,£0028
45CB            XOR A
45CC            SBC HL,BC
45CE            LD (TEMP2),HL
45D1            LD A,(ROW)
45D4            SUB 20
45D6            LD B,A
45D7            LD A,(YY)
45DA            CP B
45DB            JR Z,RET5
45DD            INC A
45DE            LD (YY),A
45E1            LD HL,(TEMP4)
45E4            LD A,(COL)
45E7            LD C,A
45E8            LD B,0
45EA            ADD HL,BC
45EB            LD (TEMP4),HL
45EE RET5:      LD A,(Y)
45F1            JR RET3
45F3 RET7:      LD A,(X)
45F6            LD C,A
45F7            LD A,40
45F9            SUB C
45FA            LD C,A
45FB            LD B,0
45FD            ADD HL,BC
45FE            LD BC,£0028
4601            XOR A
4602            SBC HL,BC
4604            LD (TEMP2),HL
4607 RET8:      LD A,(Y)
460A            CP 1
460C            JR Z,RET10
460E            DEC A
460F            LD (Y),A
4612 RET9:      LD HL,(TEMP4)
4615            PUSH AF
4616            LD A,(COL)
4619            SUB 40
461B            LD C,A
461C            LD B,0
461E            ADD HL,BC
461F            LD (TEMP1),HL
4622            LD HL,(TEMP4)
4625            POP AF
4626            LD (MPR),A
4629            LD A,(COL)
462C            LD (MPD),A
462F            CALL MULT
4632            LD DE,(RES)
4636            ADD HL,DE
4637            DEC HL
4638            EX DE,HL
4639            LD A,40
463B            LD (X),A
463E            LD A,(COL)
4641            SUB 39
4643            LD (XX),A
4646            LD HL,(TEMP2)
4649            CALL SCROLL
464C            JP UPDATE
464F RET10:     LD HL,(TEMP2)
4652            LD BC,£0028
4655            ADD HL,BC
4656            LD (TEMP2),HL
4659            LD A,(YY)
465C            CP 1
465E            JR Z,RET9
4660            DEC A
4661            LD (YY),A
4664            LD HL,(TEMP4)
4667            LD A,(COL)
466A            LD C,A
466B            LD B,0
466D            XOR A
466E            SBC HL,BC
4670            LD (TEMP4),HL
4673 RET11:     LD A,(Y)
4676            JR RET9
4678 INSERT:    LD A,(INS)
467B            BIT 0,A
467D            JR Z,INS1

467F            RES 0,A
4681            LD (INS),A
4684            PUSH HL
4685            LD BC,MES22
4688            CALL SCR3
468B            POP HL
468C            CALL LRAM
468F            JP GETCHR
4692 INS1:      SET 0,A
4694            LD (INS),A
4697            PUSH HL
4698            LD BC,MES7
469B            CALL SCR3
469E            POP HL
469F            CALL LRAM
46A2            JP GETCHR
46A5 INS2:      CALL EOL2
46A8            LD A,(COL)
46AB            SUB C
46AC            CP 0
46AE            JP Z,OUTCHR1
46B1            LD C,A
46B2            LD HL,(TEMP3)
46B5            INC HL
46B6            LD A,(DE)
46B7 INS3:      LD B,(HL)
46B8            LD (HL),A
46B9            LD A,B
46BA            INC HL
```

```
46BB            DEC C
46BC            JR NZ,INS3
46BE            LD HL,(TEMP2)
46C1            JP OUTCHR1
46C4 DELETE:    CALL EOL2
46C7            LD A,(COL)
46CA            SUB C
46CB            CP 0
46CD            JP Z,DEL2
46D0            LD C,A
46D1            LD B,0
46D3            LD HL,(TEMP3)
46D6            INC HL
46D7 DEL1:      LDIR
46D9 DEL2:      LD A,32
46DB            LD (DE),A
46DC            LD HL,(TEMP2)
46DF            LD DE,(TEMP3)
46E3            CALL SCROLL
46E6            JP GETCHR
46E9 EOL:       CALL EOL2
46EC            LD A,(COL)
46EF            SUB C
46F0            LD B,A
46F1            INC B
46F2            EX DE,HL
46F3            LD A,32
46F5 EOL1:      LD (HL),A
46F6            INC HL
46F7            DJNZ EOL1
46F9            LD HL,(TEMP2)
46FC            LD DE,(TEMP3)
4700            CALL SCROLL
4703            JP GETCHR
4706 EOL2:      LD (TEMP2),HL
4709            LD (TEMP3),DE
470D XPOS:      LD A,(X)
4710            LD C,A
4711            LD A,(XX)
4714            DEC A
4715            ADD A,C
4716            LD C,A
4717            RET
4718 YPOS:      LD A,(Y)
471B            LD C,A
471C            LD A,(YY)
471F            DEC A
4720            ADD A,C
4721            LD C,A
4722            RET
4723 JOYS:      XOR A
4724            PUSH HL
4725            LD HL,KEY
4728            LD (HL),A
4729 FIRE:      LD A,£DF
472B            CALL STROBE
472E            JR NZ,LL
4730            SET 4,(HL)
4732 LL:        LD A,£F7
4734            CALL STROBE
4737            JR NZ,RR
4739            SET 0,(HL)
473B RR:        LD A,£EF
473D            CALL STROBE
4740            JR NZ,UU
4742            SET 1,(HL)
4744 UU:        LD A,£FB
4746            CALL STROBE
4749            JR NZ,DD
474B            SET 2,(HL)
474D DD:        LD A,£BF
474F            CALL STROBE
4752            JR NZ,DONE
4754            SET 3,(HL)
4756            JR DONE
4758 KEY:       DB 0
4759 STROBE:    OUT (5),A
475B            IN A,(5)
475D            CP 127
475F            RET
4760 DONE:      LD A,25
4762            LD (CHR),A
4765            LD A,(HL)
4766            POP HL
4767            CP 1
4769            JP Z,LEFT
476C            CP 2
476E            JP Z,RIGHT
4771            CP 4
4773            JP Z,UP
4776            CP 8
4778            JP Z,DOWN
477B            CP 16
477D            JP Z,SLOW
4780            JR JOYS
4782 SLOW:      LD A,(INS)
4785            SET 1,A
4787            LD (INS),A
478A            LD BC,MES7
478D            PUSH HL
478E            CALL SCR3
4791            POP HL
4792            CALL LRAM
4795            CALL DELAY
4798            JP GETCHR
479B DELAY:     LD B,40
479D DLY:       HALT
479E            DJNZ DLY
47A0            RET
47A1 FAST:      LD A,(INS)
47A4            RES 1,A
47A6            LD (INS),A
47A9            LD BC,MES9
47AC            PUSH HL
47AD            CALL SCR3
47B0            POP HL
```

TO BE CONTINUED




```
4299  00            CHAR:   DB    00H
429A  00            INKCOL: DB    00H
429B  01            PAPCOL: DB    01H

429C                ADDOUT:
429C  F5                    PUSH  AF
429D  7D                    LD    A,L
429E  D3 02                 OUT   (02),A
42A0  7C                    LD    A,H
42A1  F6 40                 OR    40H
42A3  D3 02                 OUT   (02),A      ;These are the only value in the VDP routines
42A5  F1                    POP   AF          ;that would need changing for another machine
42A6  C9                    RET

42A7                ADDIN:
42A7  F5                    PUSH  AF
42A8  7D                    LD    A,L
42A9  D3 02                 OUT   (02),A
42AB  7C                    LD    A,H
42AC  E6 3F                 AND   3FH
42AE  D3 02                 OUT   (02),A
42B0  F1                    POP   AF
42B1  C9                    RET

                    ; Because the stack builds down in memory reserve some space ... this is more than ample.

42B2                STKEND: DS    100H
43B2  0000          STACK:  DW    0000H

                    ; The following are the register values to set up the vdp where we want it !

43B4  02 C0 0F FF   REGSET: DB    02H,0C0H,0FH,0FFH,03H,07EH,07H,11H
43BB  03 7E 07 11

                    ; This is the character set that we shall use instead of the internal MIX

43BC  00 00 00 00   CHARX:  DB    00,00,00,00,00,00,00,00
43C0  00 00 00 00
43C4  04 04 04 04           DB    04,04,04,04,00,00,04,00     ;!
43C8  00 00 04 00
43CC  00 0A 0A 00           DB    00,10,10,00,00,00,00,00     ;"
43D0  00 00 00 00
43D4  11 0A 1F 0A           DB    17,10,31,10,31,10,10,00     ;£
43D8  1F 0A 0A 00
43DC  04 0F 14 0E           DB    04,15,20,14,05,30,04,00     ;$
43E0  05 1E 04 00
43E4  18 19 02 04           DB    24,25,02,04,10,19,03,00     ;%
```

```
43E8  0A 13 03 00
43EC  00 00 00 00          DB    00,00,00,00,00,00,00,00    ;&..
43F0  00 00 00 00
43F4  02 04 0A 00          DB    02,04,10,00,00,00,00,00    ;'
43F8  00 00 00 00
43FC  02 04 0A 0A          DB    02,04,10,10,10,04,02,00    ;(
4400  0A 04 02 00
4404  0A 04 02 02          DB    10,04,02,02,02,04,10,00    ;)
4408  02 04 0A 00
440C  04 15 0E 04          DB    04,21,14,04,14,21,04,00    ;*
4410  0E 15 04 00
4414  00 04 04 1F          DB    00,04,04,31,04,04,00,00    ;+
4418  04 04 00 00
441C  00 00 00 00          DB    00,00,00,00,00,04,04,10    ;,
4420  00 04 04 0A
4424  00 00 00 0F          DB    00,00,00,15,00,00,00,00    ;-
4428  00 00 00 00
442C  00 00 00 00          DB    00,00,00,00,00,12,12,00
4430  00 0C 0C 00
4434  00 00 01 02          DB    00,00,01,02,04,10,16,00    ;/
4438  04 0A 10 00
443C  0E 11 13 15          DB    14,17,19,21,25,17,14,00    ;0
4440  19 11 0E 00
4444  04 0C 14 04          DB    04,12,20,04,04,04,31,00    ;1
4448  04 04 1F 00
444C  0E 11 01 02          DB    14,17,01,02,12,16,31,00    ;2
4450  0C 10 1F 00
4454  0E 11 01 06          DB    14,17,01,06,01,17,14,00    ;3
4458  01 11 0E 00
445C  02 06 0A 12          DB    02,06,10,18,31,02,02,00    ;4
4460  1F 02 02 00
4464  1F 10 1C 02          DB    31,16,28,02,01,02,28,00    ;5
4468  01 02 1C 00
446C  06 0A 10 1E          DB    06,10,16,30,17,17,14,00    ;6
4470  11 11 0E 00
4474  1F 11 02 04          DB    31,17,02,04,04,04,04,00    ;7
4478  04 04 04 00
447C  0E 11 11 0E          DB    14,17,17,14,17,17,14,00    ;8
4480  11 11 0E 00
4484  0E 11 11 0F          DB    14,17,17,15,01,02,12,00    ;9
4488  01 02 0C 00
448C  00 00 04 00          DB    00,00,04,00,00,04,00,00    ;:
4490  00 04 00 00
4494  00 00 04 00          DB    00,00,04,00,00,04,12,10    ;;
4498  00 04 0C 0A
449C  03 06 0C 18          DB    03,06,12,24,12,06,03,00    ;`
44A0  0C 06 03 00
44A4  00 00 1F 00          DB    00,00,31,00,31,00,00,00    ;=
44A8  1F 00 00 00
44AC  18 0C 06 03          DB    24,12,06,03,06,12,24,00    ;"
44B0  06 0C 18 00
44B4  0E 11 01 02          DB    14,17,01,02,04,00,04,00    ;?
44B8  04 00 04 00
44BC  0E 11 01 0D          DB    14,17,01,13,21,21,14,00    ;@
44C0  15 15 0E 00
44C4  04 0A 11 11          DB    04,10,17,17,31,17,17,00    ;A
44C8  1F 11 11 00
44CC  1E 09 09 0E          DB    30,9,9,14,9,9,30,00        ;B
44D0  09 09 1E 00
44D4  06 09 10 10          DB    06,9,16,16,16,9,06,00      ;C
```

```
44D8    10 09 06 00
44DC    1C 0A 09 09     DB      28,10,9,9,9,10,28,00        ;D
44E0    09 0A 1C 00
44E4    1F 10 10 1E     DB      31,16,16,30,16,16,31,00     ;E
44E8    10 10 1F 00
44EC    1F 10 10 1E     DB      31,16,16,30,16,16,16,00     ;F
44F0    10 10 10 00
44F4    0E 11 10 17     DB      14,17,16,23,17,17,14,00     ;G
44F8    11 11 0E 00
44FC    11 11 11 1F     DB      17,17,17,31,17,17,17,00     ;H
4500    11 11 11 00
4504    0E 04 04 04     DB      14,04,04,04,04,04,14,00     ;I
4508    04 04 0E 00
450C    07 02 02 02     DB      07,02,02,02,18,18,12,00     ;J
4510    12 12 0C 00
4514    11 12 14 18     DB      17,18,20,24,20,18,17,00     ;K
4518    14 12.11 00
451C    10 10 10 10     DB      16,16,16,16,16,16,31,00     ;L
4520    10 10 1F 00
4524    11 1B 15 15     DB      17,27,21,21,17,17,17,00     ;M
4528    11 11 11 00
452C    11 19 19 15     DB      17,25,25,21,19,19,17,00     ;N
4530    13 13 11 00
4534    0E 11 11 11     DB      14,17,17,17,17,17,14,00     ;O
4538    11 11 0E 00
453C    1E 11 11 1E     DB      30,17,17,30,16,16,16,00     ;P
4540    10 10 10 00
4544    0E 11 11 11     DB      14,17,17,17,21,18,13,00     ;Q
4548    15 12 0D 00
454C    1E 11 11 1E     DB      30,17,17,30,20,18,17,00     ;R
4550    14 12 11 00
4554    0E 11 10 0E     DB      14,17,16,14,01,17,14,00     ;S
4558    01 11 0E 00
455C    1F 04 04 04     DB      31,04,04,04,04,04,04,00     ;T
4560    04 04 04 00
4564    11 11 11 11     DB      17,17,17,17,17,17,14,00     ;U
4568    11 11 0E 00
456C    11 11 11 11     DB      17,17,17,17,10,10,04,00     ;V
4570    0A 0A 04 00
4574    11 11 11 15     DB      17,17,17,21,21,27,17,00     ;W
4578    15 1B 11 00
457C    11 11 0A 04     DB      17,17,10,04,10,17,17,00     ;X
4580    0A 11 11 00
4584    11 11 11 0E     DB      17,17,17,14,04,04,04,00     ;Y
4588    04 04 04 00
458C    1F 01 02 04     DB      31,01,02,04,10,16,31,00     ;Z
4590    0A 10 1F 00
4594    63 00 00 00     DB      99,00,00,00,00,00,00,00
4598    00 00 00 00
459C    00 00 00 00     DB      00,00,00,00,00,00,00,00
45A0    00 00 00 00
45A4    00 00 00 00     DB      00,00,00,00,00,00,00,00
45A8    00 00 00 00
45AC    00 00 00 00     DB      00,00,00,00,00,00,00,00
45B0    00 00 00 00
45B4    00 00 00 00     DB      00,00,00,00,00,00,00,00
```

```
45B8    00 00 00 00
45BC    0F 11 10 46          DB      15,17,16,70,16,16,31,00      ;°  CHR$(60H) (96)dec
45C0    10 10 1F 00
45C4    00 00 0E 01          DB      00,00,14,01,15,17,15,00      ;a
45C8    0F 11 0F 00
45CC    10 10 16 19          DB      16,16,22,25,17,25,22,00      ;b
45D0    11 19 16 00
45D4    00 00 0E 11          DB      00,00,14,17,16,17,14,00      ;c
45D8    10 11 0E 00
45DC    01 01 0D 13          DB      01,01,13,19,17,19,13,00      ;d
45E0    11 13 0D 00
45E4    00 00 0E 11          DB      00,00,14,17,31,16,14,00      ;e
45E8    1F 10 0E 00
45EC    02 05 04 1F          DB      02,05,04,31,04,04,04,00      ;f
45F0    04 04 04 00
45F4    00 00 0D 13          DB      00,00,13,19,19,13,01,16      ;g
45F8    13 0D 01 10
45FC    10 10 1E 11          DB      16,16,30,17,17,17,17,00      ;h
4600    11 11 11 00
4604    04 00 0C 04          DB      04,00,12,04,04,04,14,00      ;i
4608    04 04 0E 00
460C    02 00 06 02          DB      02,00,06,02,02,02,18,14      ;j
4610    02 02 12 0E
4614    0A 0A 09 0A          DB      10,10,9,10,12,10,9,00        ;k
4618    0C 0A 09 00
461C    0C 04 04 04          DB      12,04,04,04,04,04,14,00      ;l
4620    04 04 0E 00
4624    00 00 1A 15          DB      00,00,26,21,21,21,21,00      ;m
4628    15 15 15 00
462C    00 00 16 19          DB      00,00,22,25,17,17,17,00      ;n
4630    11 11 11 00
4634    00 00 0E 11          DB      00,00,14,17,17,17,14,00      ;o
4638    11 11 0E 00
463C    00 00 16 19          DB      00,00,22,25,25,22,16,20      ;p
4640    19 16 10 14
4644    00 00 0D 13          DB      00,00,13,19,19,13,01,01      ;q
4648    13 0D 01 01
464C    00 00 16 19          DB      00,00,22,25,16,16,16,00      ;r
4650    10 10 10 00
4654    00 00 0F 10          DB      00,00,15,16,30,01,30,00      ;s
4658    1E 01 1E 00
465C    0A 0A 1E 0A          DB      10,10,30,10,10,9,06,00       ;t
4660    0A 09 06 00
4664    00 00 12 12          DB      00,00,18,18,18,18,13,00      ;u
4668    12 12 0D 00
466C    00 00 11 11          DB      00,00,17,17,17,10,04,00      ;v
4670    11 0A 04 00
4674    00 00 11 15          DB      00,00,17,21,21,21,10,00      ;w
4678    15 15 0A 00
467C    00 00 11 0A          DB      00,00,17,10,04,10,17,00      ;x
4680    04 0A 11 00
4684    00 00 11 11          DB      00,00,17,17,19,13,01,16      ;y
4688    13 0D 01 10
468C    00 00 1F 02          DB      00,00,31,02,04,10,31,00      ;z
4690    04 0A 1F 00
4694    03 04 04 0A          DB      03,04,04,10,04,04,03,00
4698    04 04 03 00

                    ;End of normal Character set....

469C                START:          ;CONTINUED NEXT MONTH HERE
```

That's Life!

The Following program has been sent to us from Mr R. Nunn, this is a variation on the "Game of Life" as published in issue five volume three. The program has the addtional features of horizontal planes and a wrap around, both vertical and horizontal.

This is the Basic listing and sound can be added to give a further dimention. Frequency dropping as cells die out and increasing as regeneration takes place. Each generation takes about 0.5 of a second and the longest 'movement' of cells takes around ten minutes, this constitutes over 1000 generations.

```
10 CODE

           LD A,40          ;VS5 TO 40 WIDE
           LD (#FFAD),A
           RST 10
           DB 4D
           XOR A
           OUT (2),A
           LD A,#5C
           OUT (2),A
           LD BC,960
NEXT:      LD HL,(3FD5C)
           LD D,A           ;PSUEDO-RANDOM
           LD E,L           ;NUMBER
           ADD HL,HL        ;GENERATOR
           ADD HL,HL
           ADD HL,DE
           ADD HL,HL
           ADD HL,HL
           ADD HL,HL
           ADD HL,DE
           LD (#FD5C),HL
           LD A,H
           CP #C4
           JR C,NONC        ;PRINT A
           LD A,"*"         ;CHARACTER
           JR CHAR
NONC:      LD A," "         ;OR NOT
CHAR:      OUT (1),A
           DEC BC
           LD A,B
           OR C
           JR NZ,NEX1
NEXGEN:    LD HL,(A)        ;PLACE SCREEN
           LD BC,960        ;TO DUMP
           RST 10
           DB #1A
           XOR A
           OUT (2),A
           LD A,#1C
           OUT (2),A
PR1:       IN A,(1)
           LD (HL),A
           INC HL
```

```
            DEC BC
            LD A,B
            OR C
            JR NZ,PR1
            LD HL,(A)       ;CHECK EACH
            EXX             ;POSITION
            LD BC,960       ;NEIGHBOURING
            LD DE,#A400     ;STARS
            EXX
NEXCELL:    LD BC,#800      ;C=STAR COUNT
            PUSH HL         ;B= NO OF NEIGHBOURS
            PUSH HL
            LD HL,TAB       ;TABLE OF
            JR N1           ;DISPLACEMENTS
NEXDIS:     EX (SP),HL
            RST 8
            EX (SP),HL
            ADD HL,DE
Z:          LD A,(HL)
            CP 32
            JR C,CHANGE
            CP "*"
            JR NZ,N2
            INC C
N2:         DJNZ NEXDIS
            POP HL
            POP HL
            LD A,C
            CP 2            ;IF LESS THAN
            JR C,NODE       ;2, THEN SPACE
            CP 4            ;IF 4 OR MORE
            JR NC,NOCE      ;THEN SPACE
            CP 3            ;IF 3 THEN
            JR 2,CELL       ;PRINT A STAR
            LD A,(HL)
            JR PUT
CELL:       LD A,"*"
            JR PUT
NOCE:       LD A," "
PUT:        INC HL
            EXX
            LD (DE),A
            INC DE
            DEC BC          ;REPEAT 960
            LD A,B          ;TIMES
            OR C
            EXX
            JR NZ,NEXCELL
            LD HL,#A400
            LD BC,960
            XOR A
            OUT (2),A
            LD A,#5C
            OUT (2),A
            LD A,(HL)
            OUT (1),A
            INC HL
            DEC BC
            LD A,B
            OR C
            JR NZ,K
KSC:        CALL #/9        ;END OF SCREEN
            JR Z,NEXGEN
            RET             ;RET ON INPUT
CHANGE:     LD A,H          ;ADJUSTS FOR
            LD DE,960       ;TOP OR BOTTOM
            CP #A0          ;OF SCREEN
            JR NC,CH1
            ADD HL,DE
            JR Z
CH1:        AND A
            SBC HL,DE
            JR Z
TAB:        DB #D/,255,1,0,1,0,40,0
            DB 40,0,255,255,255,255,#D8,255
            DW #A000
            RET
```

SYMBOLS:

NEX1	4018	NUNC	4030
CHAR	4032	PR1	4048
NEXGEN	4039	NEXCELL	405C
NEXDIS	4066	CHANGE	40B4
CELL	4088	POT	408E
CH1	40BF	Z	406A
KSC	40AE	N1	4067
N2	4074	TAB	40C4
NODE	408C	A	40D4
K	40A5		

Volume Three **MEMOPAD** Number 8

FOOTBALL POOLS PREDICTOR

The original program was by Professor Frank George as a result of his keen interest, as a mathematician, in the Football Pools. The first program was written for a mainframe computer, and later adapted for the Commodore 64 home computer. This version for the Memotech MTX was adapted and modified by Dave Wemyss.

The program uses blocks of information that are stored on disc. The Team Files (.TMS) hold details of each team, while the Fixtures Files (.FIX) hold the names of each pair of teams on the pools coupon. The program allows for the setting up of these files, and also gives the facility for updating the information. For this you will need:

a. results of previous Saturday's matches
b. results of matches played on any other day
c. list of matches on next pools coupon

For accuracy in forecast, it is essential that all league results are entered. Using a series of comparisons, the program will produce a forecast for the next set of matches.

Press <RET> to continue

* PLEASE NOTE:- This is a series of 6 programmes (some to be published in future editions) which when linked together, make a complete pools predictor.

```
110 REM ********************************************************************
20 REM ********************* FOOTBALL POOLS FORECASTING *********************
30 REM *********************    PROFESSOR FRANK GEORGE   *********************
40 REM *********************       COMMODORE 64          *********************
50 REM *********************     ADAPTED FOR MEMOTECH    *********************
60 REM *********************       BY DAVE WEMYSS        *********************
70 REM *********************         JAN 1987            *********************
80 REM *********************         POOLS.BAS           *********************
90 REM ********************************************************************
100 DISC SAVE "POOLS.BAS"
110 V6 5: CLS : CLEAR
120 DIM F$(6,14),D$(6,23),CONF$(3)
130 LET F$(1)="F1.............": LET F$(2)="F2.............": LET F$(3)="F3.............": LET F$(4
)="F4.............": LET F$(5)="F5.............": LET F$(6)="F6............."
140 LET D$(1)="Start New File": LET D$(2)="View/Update Team Files": LET D$(3)="View/Update Fixtu
res": LET D$(4)="Forecast Results": LET D$(5)="Sort league positions": LET D$(6)="End Session"
150 GOSUB 400: CLS : CSR 25,0: PRINT "Football Pools Forecasting": CSR 24,1: PRINT "==========
==================="
160 LET N=1
170 FOR I=3 TO 18 STEP 3
180 CSR 23,I: PRINT F$(N);D$(N)
190 LET N=N+1: NEXT I
200 LET A$=INKEY$: LET A=ASC(A$)
210 IF A<128 OR A>133 THEN GOTO 200
220 CLS : CSR 25,10
230 LET A=ABS(A-128): ON A GOTO 300,320,340,360,500,270
270 INPUT "Type <YES> to quit ";CONF$
280 IF CONF$<>"YES" AND CONF$<>"yes" THEN CLS : GOTO 150
290 NEW
300 PRINT "New File Module loading": GOSUB 380
310 DISC LOAD "INPUTTMS.BAS"
```

```
320 PRINT "View/Update Teams Module loading": GOSUB 380
330 DISC LOAD "VIEWAMND.TMS"
340 PRINT "View/Update Fixtures Module loading": GOSUB 380
350 DISC LOAD "VIEWAMND.FIX"
360 PRINT "Forecast Results Module loading": GOSUB 380
370 DISC LOAD "FORECAST.BAS"
380 CSR 25,12: PRINT "Please wait....."
390 RETURN
400 CSR 20,10: PRINT "Do you want the introduction? (Y/N)": GOSUB 450
410 IF A$="N" OR A$="n" THEN RETURN ELSE CLS : PLOD "PROG1"
420 RETURN
450 LET A$=INKEY$: IF A$<>"" THEN GOTO 450
460 LET A$=INKEY$: IF A$="" THEN GOTO 460
470 IF A$<>"N" AND A$<>"n" AND A$<>"Y" AND A$<>"y" THEN GOTO 450
480 RETURN
500 PRINT "Sort Positions Module loading": GOSUB 380
510 DISC LOAD "POSITION.SOR"



10 REM ********************************************************************************
20 REM ********************* FOOTBALL POOLS FORECASTING ***********************
30 REM ********************    PROFESSOR FRANK GEORGE    ***********************
40 REM ********************       COMMODORE 64           ***********************
50 REM ********************    ADAPTED FOR MEMOTECH      ***********************
60 REM ********************       BY DAVE WEMYSS         ***********************
70 REM ********************         JAN 1987             ***********************
80 REM ********************        VIEWAMND.TMS          ***********************
90 REM ********************************************************************************
100 DISC SAVE "VIEWAMND.TMS"
110 VS 5: CLS : CLEAR
120 DIM RECORD$(25,11,16),HEAD$(11,11),O$(16),DIV$(25),DATE$(9),LOA$(12),EMP$(16
)
130 LET W=0: LET EMP$="                    ": LET DATE=0
180 DISC OPEN £1,"HEADINGS.PLS","I"
190 CLS : GOSUB 1200: PRINT "Loading heading number ";W: CSR 20,0: PRINT "View/A
mend Teams"
200 FOR X=1 TO 12
210 DISC EOF £1,250
220 LET W=W+1: CSR 43,7: PRINT W: PAUSE 100
230 DISC INPUT £1,HEAD$(X)
240 NEXT X
250 DISC CLOSE £1
260 CLS : PLOD "PROG1"
270 GOSUB 1050: IF A<128 OR A>134 THEN GOTO 270
280 IF A=128 THEN LET LOA$="ENGDIVS1.TMS": LET DIV$="English First Division"
290 IF A=129 THEN LET LOA$="ENGDIVS2.TMS": LET DIV$="English Second Division"
300 IF A=130 THEN LET LOA$="ENGDIVS3.TMS": LET DIV$="English Third Division"
310 IF A=131 THEN LET LOA$="ENGDIVS4.TMS": LET DIV$="English Fourth Division"
320 IF A=132 THEN LET LOA$="SCOTPREM.TMS": LET DIV$="Scottish Premier Division"
330 IF A=133 THEN LET LOA$="SCOTDIV1.TMS": LET DIV$="Scottish First Division"
340 IF A=134 THEN LET LOA$="SCOTDIV2.TMS": LET DIV$="Scottish Second Division"
360 DISC OPEN £1,LOA$,"I"
370 LET Z=0: GOSUB 1200: PRINT "Loading team number ";Z
375 DISC INPUT £1,DATE$
378 CSR 50,0: PRINT DATE$
380 FOR X=1 TO 25
390 DISC EOF £1,440
```

```
400 LET Z=Z+1: CSR 40,7: PRINT Z
410 FOR I=1 TO W
420 DISC INPUT £1,RECORD$(X,I)
430 NEXT I: NEXT X
440 DISC CLOSE £1
460 CLS : CSR 10,10: PRINT "F1......View                                    F2......Change":
CSR 10,12: PRINT "F3......View another Division      F4......Return to Main Menu
"
470 GOSUB 1050: IF A<128 OR A>131 THEN GOTO 470
480 IF A=129 THEN GOTO 580
485 IF A=130 THEN GOTO 260
490 IF A=131 THEN GOTO 1000
500 FOR X=1 TO Z
510 GOSUB 1250
520 CSR 10,20: PRINT "F1......Next Team    F2......Last Team    F3......Finished
"
530 GOSUB 1050: IF A<128 OR A>130 THEN GOTO 530
540 IF A=130 THEN CLS : GOTO 460
550 IF A=129 THEN LET X=X-2: IF X<0 THEN LET X=0
560 NEXT X
570 GOTO 460
580 REM Change Teams
590 LET CHECK=0: CLS : IF DATE=0 THEN CSR 20,10: INPUT "Enter date of changes >
";NEWDATE$
600 IF CHECK=0 THEN GOTO 620 ELSE CLS : CSR 20,10: PRINT "Any more changes? (Y/N
)": GOSUB 1150
610 IF A$="N" OR A$="n" THEN GOTO 850
620 LET CHECK=1: LET DATE=1: LET DATE$=NEWDATE$
630 CLS : CSR 20,10: INPUT "Team to change > ";CHANGE$: IF LEN(CHANGE$)>15 THEN
GOTO 630 ELSE LET SER=LEN(CHANGE$)
640 FOR X=1 TO Z
650 IF LEFT$(RECORD$(X,1),SER)=CHANGE$ THEN GOTO 680
660 NEXT X
670 CLS : CSR 20,10: PRINT "Team not found": PAUSE 2000: GOTO 600
680 GOSUB 1250
690 FOR I=1 TO W: LET O$=EMP$
700 CSR 0,20: PRINT CHR$(5): CSR 20,20
710 PRINT RECORD$(X,I): CSR 40,20: PRINT "Any changes? (Y/N)": GOSUB 1150
720 IF A$="N" OR A$="n" THEN GOTO 750
730 CSR 0,20: PRINT CHR$(5): CSR 20,20: INPUT "Enter new record (Max 15 chars) >
 ";O$: IF LEN(O$)>15 THEN GOTO 730 ELSE LET RECORD$(X,I)=O$
740 CSR 36,I+4: PRINT RECORD$(X,I)
750 NEXT I
760 CLS : FOR I=1 TO W
770 CSR 20,I+4: PRINT HEAD$(I): CSR 34,I+4: PRINT ": ";RECORD$(X,I): NEXT I
780 CSR 20,20: PRINT "Record O.K.? (Y/N)": GOSUB 1150
790 IF A$="Y" OR A$="y" THEN GOTO 600
800 CSR 0,20: PRINT CHR$(5): CSR 20,20: INPUT "Which line to change? > ";F
810 LET O$=EMP$: CSR 0,20: PRINT CHR$(5): CSR 20,20: INPUT "Enter new record (Ma
x 15 chars) > ";O$: IF LEN(O$)>15 THEN GOTO 810 ELSE LET RECORD$(X,F)=O$
820 CSR 36,F+4: PRINT RECORD$(X,F)
830 CSR 0,20: PRINT CHR$(5)
840 GOTO 780
850 CLS : DISC ERA LOA$
860 DISC OPEN £1,LOA$,"O"
880 DISC PRINT £1,DATE$
890 GOSUB 1200: PRINT "Saving team number "
900 FOR X=1 TO Z
910 CSR 39,7: PRINT X
```

```
920 FOR I=1 TO W
930 DISC PRINT £1,RECORD$(X,I)
940 NEXT I: NEXT X
950 DISC CLOSE £1
960 FOR N=5 TO 7 STEP 2: CSR 0,N: PRINT CHR$(5): NEXT N: CSR 20,10: PRINT "Chang
ed data now saved"
970 CSR 5,20: PRINT "F1.....View/Update another division    F2.....No further
changes"
980 GOSUB 1050: IF A<128 OR A>129 THEN GOTO 980
990 IF A=128 THEN GOTO 260
1000 CLS : GOSUB 1200: PRINT "Returning to Main Menu"
1010 DISC LOAD "POOLS.BAS"
1020 PRINT RECORD$(X,I): CSR 40,20: PRINT "Any changes? (Y/N)": GOSUB 900
1030 IF A$="N" OR A$="n" THEN GOTO 1050
1040 CSR 0,20: PRINT CHR$(5): CSR 20,20: INPUT "Enter new record (Max 15 chars)
> ";O$: IF LEN(O$)>15 THEN GOTO 1040 ELSE LET RECORD$(X,I)=O$
1050 LET A$=INKEY$: IF A$<>"" THEN GOTO 1050
1060 LET A$=INKEY$: IF A$="" THEN GOTO 1060
1070 LET A=ASC(A$)
1080 RETURN
1100 FOR J=0 TO 79: PRINT "-";: NEXT J
1110 RETURN
1150 LET A$=INKEY$: IF A$<>"" THEN GOTO 1150
1160 LET A$=INKEY$: IF A$="" THEN GOTO 1160
1170 IF A$<>"N" AND A$<>"n" AND A$<>"Y" AND A$<>"y" THEN GOTO 1150
1180 RETURN
1200 CLS : CSR 20,0: PRINT DIV$: CSR 50,0: PRINT DATE$: GOSUB 1100
1210 CSR 20,5: PRINT "Please wait.....": CSR 20,7
1220 RETURN
1230 GOTO 1170
1240 RETURN
1250 CLS : CSR 0,0: PRINT DATE$: CSR 20,0: PRINT DIV$: GOSUB 1100
1260 FOR I=1 TO W: CSR 60,0: PRINT X

1270 CSR 20,I+4: PRINT HEAD$(I): CSR 34,I+4: PRINT ": ";RECORD$(X,I)
1280 NEXT I
1290 RETURN
1300 NEXT X          PROG1
1310 GOTO 1250

                          * D MENU1.

                          * R

                    CHOOSE DIVISION
                    ===============

 English Division One.....F1

 English Division Two.....F2

 English Division Three.....F3

 English Division Four.....F4

 Scottish Premier Division.....F5

 Scottish Division One.....F6

 Scottish Division Two.....F7
```

TO BE CONTINUED

Volume Three **MEMOPAD** Number 8

The Book List

REF	TITLE	PUBLISHER	AUTHOR	PRICE
ARTIFICIAL INTELLIGENCE				
1626	Artificial Intelligence	PRENTICE-HALL	Bonnet	9.95
2989	Artificial Intelligence	ADDISON-WESLEY	Gale	37.95
1128	Beginners guide to LISP	ADDISON-WESLEY	Hasemer	10.95
0949	Build your own expert syst	SIGMA	Naylor	8.95
2892	Building Expert - System	ADDISON WESLEY	Hayes Roth Waterman	29.95
2795	Building your first exp system	ASHTON-TATE	Nagy & Gault	21.70
1602	Expert systema and Micro's	N.C.C.	Simons	10.50
1958	Expert Systems/Practical Intro	MACMILLAN	Sell	5.95
1972	Guide to Expert Systems	ADDISON-WESLEY	Waterman	16.95
0977	Implementation of PROLOG	WILEY	Campbell	15.70
1377	Intro Artificial Intelligence	N.C.C.	Simons	12.75
2888	Intro Artificial Intelligence	ADDISON-WESLEY	Charnaik & McDermott	16.95
1969	Introduction to Expert Systems	ADDISON-WESLEY	Jackson	13.95
1129	LISP 2nd Edition	ADDISON-WESLEY	Winston & Horn	14.95
1372	LISP Language of Artificial Intel	GRANADA	Berk	9.95
1457	Learning Micro PROLOG	ADDISON-WESLEY	Conlan	8.95
1516	Micro-Prolog & Art Intell.	COLLINS	Berk	9.95
1717	Prog in Micro Prolog (paper back)	ELLIS HORWOOD	Saram	8.50
1998	Prolog prog for Artificial Intell.	ADDISON-WESLEY	Bratko	15.95
2772	Prolog Programming & Applications	MACMILLAN	Burnham & Hall	7.50
3032	Turbo Prolog Primer	SAMS	Shafer	17.95
1852	Understanding Artificial Intell.	SAMS	Mishoff	13.50
BASIC				
6006	BASIC computer games	CREATIVE COMP	AH1 DH	6.25
1051	BASIC Program Conversions	FOUNTAIN		9.95
0343	Inside BASIC Games	SYBEX	Mateosian	5.95
6007	More BASIC Computer Games	CREATIVE COMP	AH1 D H	6.25
0708	Your first BASIC Program	SYBEX	Zaks	4.95
1941	BASIC Handbook	COMPUSOFT	Lien	22.95
0579	BASIC Programs for Business V2	HAYDEN	Sternberg	8.95
0459	BASIC Programs for Scien/Engin	SYBEX	Miller	16.95
"C"				
2773	A Book on C	MACMILLAN	Berry & Meekings	19.95
2978	Advanced 'C' Primer ++	SAMS	Prata	19.95
2736	Advanced 'C'	McGRAW-HILL	Schildt	19.95
1605	Advanced 'C' Techniques & Applic.	QUE	Sobelman & Krekelber	19.95
1826	Applied Prog Tech in 'C'	SCOTT FORESMAN	Ward	18.20
1137	Big Red Book of 'C'	SIGMA	Sullivan	7.50
2887	C ++ Programming Language	ADDISON-WESLEY	Stroustrup	16.95
2857	'C' A Reference Manual	PRENTICE-HALL	Harbison & Steele	18.95
3017	'C' At A Glance	CHAPMAN & HALL	Denning	7.95
1829	'C' Language for Programmers	SCOTT FORESMAN	Pugh	16.45

2808	'C' Library	McGRAW-HILL	Jamsa	18.95
1043	'C' Programmers Library	QUE	Purdum Leslie & Steg	19.95
1608	'C' Programming Guide 2nd Edition	QUE	Purdum	18.45
8005	'C' Programming Language	PRENTICE-HALL	Kernighan & Ritchie	22.95
1201	'C' Programming Tutor	PRENTICE-HALL	Wortman & Sidebottom	13.95
1722	'C' Self Study Guide	QUE	Purdum	15.45
1534	Common 'C' Functions	QUE	Brand	16.45
1920	Data Handling Utilities In 'C'	SYBEX	Radcliffe & Raab	19.95
2860	Dr Dobbs Toolbook of 'C'	PRENTICE-HALL	Dr Dobbs Journal	21.70
1734	Introducing 'C'	COLLINS	Allan	9.95
1848	Mastering 'C'	SYBEX	Bolon	19,95
1785	Practical 'C'	SIGMA	Harrison	7.95
2992	Solutions In 'C'	ADDISON-WESLEY	Jaeschke	16.40
1853	The 'C' Compendium	SUNSHINE	Lawrence & England	12.95
2798	The 'C' Primer 2nd Edition	McGRAW-HILL	Hancock & Krieger	17.95
1674	The 'C' Toolbox	ADDISON-WESLEY	Hunt	19.20
1020	Understanding 'C'	SYBEX	Hunter	17.95
1760	Variations In 'C'	MICROSOFT	Schustacw	16.95

CHIP - Z80

0200	Programming The Z80	SYBEX	Zaks	19.95
1449	Z-80 Reference Guide	MELBOURNE HOUSE	Tully	12.95
0709	Z80 Applications	SYBEX	Coffron	15.95
1470	Z80 Assy Language Subroutines	OSBORNE/McGRAW	Leventhal & Saville	19.95
0159	Z80 Assembly Language Programming	OSBORNE/MCGRAW	Leventhal	19.95

COMMS AND INTERFC

1981	Data Communications	GLENTOP PUBLISH	Heijer & Tolsma	14.95
2891	Data Communications Programmer	ADDISON-WESLEY	Purser	13.95
2890	Local Area Network Design	ADDISON-WESLEY	Hopper-Temple-Willia	13.95
2836	Master Serial Communications	SYBEX	Gofton	19.95
2006	Microprocessor Sourcebook	PITMANS	Loveday	9.95
1873	Modem Connections Bible	SAMS	Curtis & Majhor	15.50
1769	New Hackers Handbook	CENTURY	Cornwall	6.95
1745	RS-232 Made Easy	PRENTICE-HALL	Sayer	19.95
1023	RS232 Solution	SYBEX	Campbell	17.95
1779	Using Network	QUE	Durr	22.95

CP/M

0805	C/PM Bible	SAMS	Waite & Angermeyer	16.50
0279	CP/M Handbook	SYBEX	Zaks	14.95
1214	CP/M Programmer's Encyclopedia	QUE	Brigham	18.45
2819	CP/M Solutions	PRENTICE-HALL	Barbier	13.00
0829	CP/M The Software Bus	SIGMA	Clarke,Eaton & Powys	8.95
1111	CP/M - Plus Handbook	SYBEX	Miller	13.95
1744	Inside Concurrent CP/M	HOLT RHINEHART	Curtesi	14.50
0601	Mastering CP/M	SYBEX	Miller	17.95
0806	Soul of CP/M	SAMS	Waite & Laford	16.50
2776	Using CP/M	MACMILLAN	Gosling	5.95

FORTH

0626	Complete Forth	SIGMA	Winfield	6.95
1290	Introduction to Forth	WILEY	Kail	8.95
8003	Starting Forth	PRENTICE-HALL	Brodie	17.35
2984	Forth - The Next Step	ADDISON-WESLEY	Geere	8.95

GENERAL

0939	BASIC Computer Simulations	TABS	McNitt	10.95
1052	Hypergrowth	IDTHEKKETHAN	Osbourne & Dvorak	8.95
1256	Big Computer Games	CREATIVE COMPUTER	Ahl	4.95
1063	Guide to Playing The Hobbit	MELBOURNE HOUSE	Elkan	3.95

GRAPHICS

2715	Algorithmic Image	MICROSOFT	Riulin	21.95
1487	Device - Independant Graphics	McGRAW-HILL	Sproull & Sutherland	35.95
0581	Fundementals Interact Graph	ADDISON-WESLEY	Foley & Van-Dam	23.95
2858	G.S.X. Handbook	GLENTOP PUBLISH	Digital Research	19.95
1447	Icon nad Images	COMPUTE!	Larsen	13.95
1906	Geometric & Artistic Graphics	MACMILLAN	De La Haye	9.95

PROGRAMMING TECHNIQUES

2722	Sorting Routines/Microcomputer	MACMILLAN	McLuckie & Barber	6.50
1651	The Professional Touch	SIGMA	Rowley	7.95

PROGRAMMER'S REFERENCE

1468	16 Bit Handbook	OSBOURNE/MCGRAW	Osbourne & Kane	27.50
1467	4 & 8 Bit Handbook	OSBOURNE/MCGRAW	Osbourne & Kane	27.50
1691	Knuth V 1 Fund Algor 2nd Edition	ADDISON-WESLEY	Knuth	14.95
1698	Knuth V 2 Seminum Algor 2nd Edit.	ADDISON-WESLEY	Knuth	24.95
1127	Knuth V3 Sorting & Searching	ADDISON-WESLEY	Knuth	24.95
3007	Management of Inform Systems	McGRAW-HILL	Dickson & Wetterley	11.50
1907	Microcomp Business Software	SIGMA	Rapkins	10.95
1876	Prog Gd Video Display Terminal	ATLANTIS PUBLISH	Stephens	24.95
2711	Programmers at Work	MICROSOFT	Lammers & Belel	12.95
2889	Systems Analysis & Design	ADDISON-WESLEY	Davis	12.95
1882	Industrial Robotics	GLENTOP PUBLISH	Polet	11.50
1879	Intro to Robotics	ADDISON-WESLEY	Craig	17.95
3010	Introduction to Robotics	GLENTOP PUBLISH	Loper & Numa Foult	17.50
1930	Introduction to Robotics	MACMILLAN	Critchlow	17.95
1643	The Robot Book	WINDWARD	Pawson	7.95

WORDSTAR

1965	Getting Started with Wordstar	CAMBRIDGE UNIV.	Mabbett	9.95
0503	Introduction to Wordstar	SYBEX	Naiman	17.95
1715	Introduction to Wordstar 2000	SYBEX	Koludney & Blackadar	17.95
1893	Practical Tech - Wordstar 2000	SYBEX	Donovan	18.95
0790	Practical Wordstar Uses	SYBEX	Arca	18.95
2906	Sci & Tech Text Proc Wordstar	McGRAW-HILL	McKeague	16.95

1481	Star Power	TABS	Garrison	14.90
1363	The Wordstar Handbook	HUTCHINSON	Curtin	11.95
1533	Using Wordstar 2000	QUE	Sorensen	16.45
1563	Using Wordstar 2000	PRENTICE-HALL	Barry & Krumm	16.50
0735	Wordstar & CP/M Made Easy	WILEY	Lee	8.00
0442	Wordstar Made Easy	OSBORNE/MCGRAW	Ettlin	16.95
1075	Wordstar User's Ref. Manual	DUCKWORTH	Hancorn	12.50
1371	Wordstar In Action	GRANADA	McMullan	10.95
1644	Wordstar Prompt	GRANADA	McMullan	5.95

PLEASE NOTE - IT IS NOW POSSIBLE TO BUY ANY OF THE BOOKS LISTED ABOVE THROUGH MEMOPAD, SIMPLY FILL IN THEFORM BELOW AND RETURN IT TO US.

REF	TITLE	PUBLISHER	AUTHOR	PRICE

Please find enclosed my Cheque/Postal Order for £..... in payment for the above Books.

Access..... Barclaycard .....

ALLOW 14 DAYS FOR DELIVERY, AS THE BOOKS ARE ALL SUBJECT TO AVAILABILITY.

MEMBERS ADVERTISMENT'S

* WANTED * - DBASEII or DBASEIII for the FDX on CP/M 2.2, any offer not exceeding £200 considered. Ring 0423 526267 (after 5.30 pm) Mr J.G. Metcalfe

For Sale - Obloids, Alice in Wonderland, Cobra, Time Bandits, Phaid, Tapeworm Astromillion, Blobbo and Chess, all at £4.50 (or less) - Will swop any of above games for Downstream Danger & Highway Encounter. Please contact Jamie Sneddon on (0869) 252760

Volume Three **MEMOPAD** Number 8

It's Here - The Classic Strategy Game Of 1986

FOOTBALL MANAGER

RE-WRITTEN VERSION 100% MACHINE CODE FOR FASTER INTER-ACTION

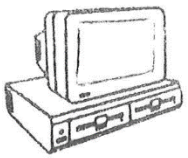
TAPE - £6.95

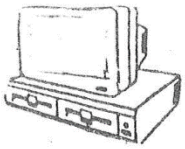
CP/M & ALL 3.5" FORMATS - £9.95

CP/M & ALL 5¼" FORMATS - £8.25

* PLEASE STATE DISC SIZE AND TYPE WHEN ORDERING *

E.G. PLEASE FORWARD FOOTBALL MANAGER FOR 5¼"
NON CP/M TYPE 03

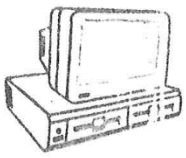


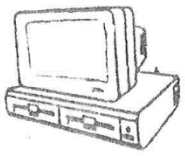


The Complete Price List

Hardware

DESCRIPTION	MEMBERS PRICE	NON MEMBERS PRICE	CARRIAGE
COMPLETE CP/M PACKAGE			
1 X 1 MBYTE 3.5" INDUSTRY STANDARD DISC DRIVE, 500K FAST ACCESS RAM DISC CP/M 2.2 OPERATING SYSTEM. 256K RAM. 12" GREEN SCREEN MONITOR CENTRONICS STANDARD PRINTER I/F POSITIVE ACTION KEYBOARD. COLOUR MONITOR OUTPUT. TWO JOYSTICK I/F.	359.95	399.95	12.96
PURCHASES INDIVIDUALLY (basic system)			
256K COMPUTER PLUS TAPE OPERATING SYSTEM	89.95	99.95	6.48
CP/M SYSTEM			
1 X 1 MBYTE 3.5" DRIVE + 512 SILICON DISC + 80 COL + CP/M + N.W.	237.59	264.00	6.48
HX 12" GREEN SCREEN MONITOR	85.49	95.00	6.48
TWIN RS232 INTERFACE	26.96	29.95	2.00
FDX 2 X 1 MBYTE CP/M + 2 MBYTE SILICON DISC.	877.50	975.00	6.48
32K MEMORY EXPANSION	37.95	39.95	2.00
64K MEMORY EXPANSION	47.45	49.95	2.00
128K MEMORY EXPANSION	75.95	79.95	2.00
NEWWORD ON ROM	37.95	39.95	2.00
PASCAL ON ROM	37.95	39.95	2.00
RS232 INTERFACE (FULL BOARD)	37.95	39.95	2.00





SIDISC PRICES

1 X 1 MBYTE	161.10	179.00	6.48
1 X 2 MBYTE	304.20	338.00	6.48
1 X 3 MBYTE	542.40	636.00	6.48
PRINTER CABLE	11.65	12.95	0.50

SPECIAL NOTES

Silicon Discs can be factory fitted for an extra #30.00 (U.K. Only).

FDX Twin Systems require RS232 Comms Board.

Carriage is applicable to U.K. orders only.

Always quote the type of computer owned Eg: MTX 500, 512 or series 2 when ordering add-on's.

** PLEASE NOTE ALL PRODUCTS WHICH ARE NOT MENTIONED ON THIS LIST ARE NO LONGER AVAILABLE **

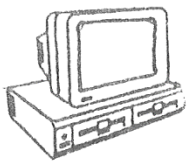
THE ISSUE OF THIS PRICE LIST CANCELS ALL PREVIOUS OFFERS

Software

TYPE CODES

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

STOCK NO.	MANUFACT. REFERENCE	TITLE	TYPE	RETAIL PRICE INCL.VAT	MEMBERS PRICE INCL.VAT
		SOFTWARE (MEMOTECH)			
		==========			
7001	00057	3D TACHYON FIGHTER	G+J	7.72	6.95
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
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
7059	00085	CAVES OF ORB	G	6.61	5.95
7061	00094	CHAMBEROIDS	G+J	7.72	6.95
7062	00059	CHESS	G	9.94	8.95
7065	00098	COMBAT	G	4.39	3.95
7069	00110	CRYSTAL	G	7.72	6.95
7092		DISASM	U	8.83	7.95
7093	00068	DOODLEBUG	G+J	6.61	5.95



TYPE CODES

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

STOCK NO.	MANUFACT. REFERENCE	TITLE	TYPE	RETAIL PRICE INCL.VAT	MEMBERS PRICE INCL.VAT
7094	00108	DOWNSTREAM DANGER	G+J	7.72	6.95
7095	00096	DR. FRANKIE	G+J	6.61	5.95
7097	00111	DRIVE THE CEE 5	G+J	7.72	6.95
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 (TIME BANDITS)	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
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
7235	00097	JUMPING JACK FLASH	G+J	6.61	5.95
7255	00115	KARATE KING	G+J	7.72	6.95
7257	00042	KILOPEDE	G+J	7.72	6.95
7258	00019	KNUCKLES	G+J	8.83	7.95
7279	00032	LITTLE DEVILS	G+J	6.61	5.95
7300	00035	MISSILE COMMAND & ARCADIANS	G+J	6.61	5.95
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
7311	00044	MISSION ALPHATRON	G+J	6.61	5.95
7312	00030	MISSION OMEGA	G+J	6.61	5.95
7333	00003	NEMO	G+J	7.72	6.95
7354	00112	OBLITERATION ZONE	G+J	7.72	6.95
7355	00045	OBLOIDS	G+J	7.72	6.95
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
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
7406		QUEST 1	G	5.78	5.20
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
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
7458	00036	SON OF PETE	G+J	7.72	6.95
7459		SOUL OF A ROBOT	G+J	5.50	4.95
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
*	7494	**	00128	*	THE WALL	*	G+J	**	6.61	**	5.95
*	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
*	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

MANUALS

				NON MEMBER		MEMBER
6501	**	CRIB CARD	**	2.99	**	2.12
6502	**	MTX ROM LISTING	**	45.00	**	32.40
6503	**	SDX CONTROLLER LISTING	**	25.00	**	18.75
6504	**	ROM CALLS INFORMATION	**	0.50	**	0.50
6505	**	RST10 CALLS INFO SHEET	**	0.50	**	0.50
6506	**	INTERRUPTS INFO SHEET	**	0.50	**	0.50
6507	**	MTX SERVICE MANUAL	**	9.95	**	9.95
6508	**	MTX NEW USER MANUAL	**	8.95	**	8.95
6509	**	VDP MANUAL	**	7.95	**	7.95
6510	**	DDT MANUAL	**	2.50	**	2.50

****

ADVERTISING IN THE MEMOPAD

SIZE OF ADVERT	NON MEMBER	MEMBER
SMALL	5.00	5.00
1/8 PAGE	27.50	27.50
1/4 PAGE	45.00	45.00
1/2 PAGE	80.00	80.00
FULL PAGE	125.00	125.00

MEMBER'S ADVERTISMENTS

MTX 512 WITH SINGLE FDX DISC DRIVE AND INTERFACE, NEWWORD, PASCAL ROM AND MANUALS - £200 Telephone: Gillian Porter on 0772 613257 (Evenings)

FOR SALE

MTX 512 + PASCAL ROM + SDX DISC, RS232, NEWWORD, FORTH MEMOSKETCH AND LOTS OF DISCS, GAMES, BOOKS etc.

£175.00 ono Telephone: Phil (0227) 263454