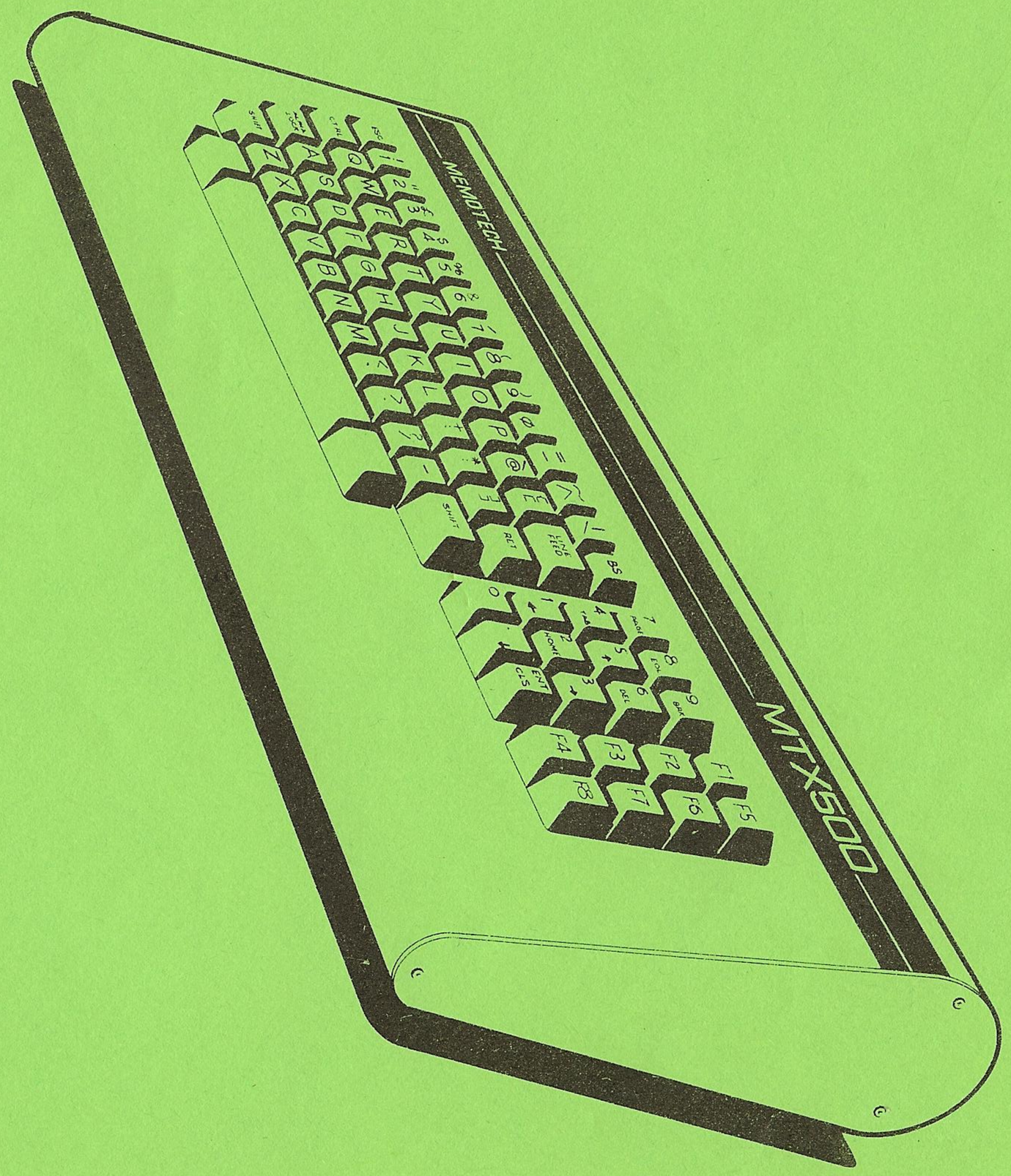
memopad

Memotech Computer User Club Magazine



MEMOTECH
MTX
SERIES

LISTING

I'm sure you have all played Milton Bradley's Connect Four - if not, the rules are simple to learn. The listing below provides you with a computer simulation of the game. The program is fully documented, and should be easy to follow.

The computer plays quite a good game, but you can alter the way it plays by changing the values in line 1020. After playing the game, you will notice that in the early stages, the computer takes about 50 secs to make a move . This is because the game is written in Basic,which brings me to the next point. **If it is your intention to learn Assembly Language programming, get to know this program inside-out !!**

Over the next issues, MEMOPAD will be showing you how to transpose the Basic version of Connect Four into a machine code program. If you study the Assembly language tutorial at the same time, you should, at the end of the series, have enough 'know-how' to start you on your way to a new, and exciting way of programming.

```
0 REM MTX 500/512    "CONNECT FOUR" COMPUTER AND ONE PLAYER.  (C) K. HOOK 1984
5 PLOD "PROG1"
10 VS 4: COLOUR 2,1: CLS : COLOUR 4,1: COLOUR 0,1: RAND -8
20 DIM TG$(8,8,2),A(4),R(8),K(4),J(4),G(16),GC(8),DN(8),X$(2),H$(2),O$(2)
25 DIM TH$(8),IN$(1): LET TH$=CHR$(134)+CHR$(134): LET TH$=TH$+TH$+TH$+TH$
30 GOSUB 1000: GOSUB 500: GOSUB 600
40 GOTO 1500
59 REM^^^^^^^^^^^^^Main Calculating^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
60 SOUND 1,0,0
100 LET O$=H$: IF X$=H$ THEN LET O$=C$
110 LET Y=1: LET YY=0: LET S=0: GOSUB 200
120 LET Y=1: LET YY=1: GOSUB 200
130 LET Y=0: LET YY=1: GOSUB 200
140 LET Y=-1: LET YY=1: GOSUB 200
150 RETURN
200 LET XX=1: LET A=1: LET Q=0: LET S=S+1
210 LET M=0
220 FOR I=1 TO 3: LET H=X+I*YY: LET N=R+I*Y
230 IF H<1 OR N<1 OR H>8 OR N>8 THEN GOTO 280
240 LET G$=TG$(N,H): IF M=0 THEN GOTO 260
245 IF G$=O$ THEN LET I=3: GOTO 290
250 LET Q=Q+1: GOTO 280
260 IF G$=X$ THEN LET A=A+1: GOTO 280
270 LET M=1: GOTO 245
280 NEXT I
290 IF XX=0 THEN LET A(S)=A: LET K(S)=Q: RETURN
300 LET XX=0: LET YY=-YY: LET Y=-Y: GOTO 210
500 REM ^^^^^^^^^^Draw Board^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
505 PAPER 1: INK 10
510 FOR I=36 TO 164 STEP 16: LINE 36,I,228,I: NEXT I
520 FOR I=36 TO 228 STEP 24: LINE I,22,I,164: NEXT I: INK 4: LINE 36,22,228,22
530 LET J=1: FOR I=5 TO 26 STEP 3: CSR I,20: PRINT J;: LET J=J+1: NEXT I
540 COLOUR 1,15: COLOUR 0,6: CSR 9,1: PRINT "CONNECT    FOUR";: RETURN
```

```
549 REM ~~~~~~~~~~Main Input Routine~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
560 CSR 5,22: INK 0: PRINT CHR$(5);
570 COLOUR 0,11: INK 1: CSR 5,22: PRINT MES$;
573 LET IN$=INKEY$
575 IF IN$="" THEN GOTO 573
580 SOUND 1,56,12: RETURN
599 REM~~~~~~~~~~~~Define Characters~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
600 GENPAT 1,130,07,31,51,127,126,59,28,07
610 GENPAT 1,131,224,248,156,254,126,220,56,224
615 GENPAT 1,134,255,255,255,255,255,255,255,255
620 GENPAT 1,132,07,15,153,185,255,191,156,07
630 GENPAT 1,133,224,240,153,157,255,253,25,224
640 LET H$=CHR$(130)+CHR$(131)
650 LET C$=CHR$(132)+CHR$(133)
660 RETURN
999 REM~~~~~~~~~~~~Initialise Variables~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1000 DATA  5,18,8,16,11,14,14,12,17,10,20,8,23,6,26,4
1010 FOR I=1 TO 8: READ GC(I),DN(I): NEXT I
1018 REM ~~~~~~Values below control computer evaluatioions ~~~~~~~~~~~~~~~~~~
1019 REM ~~~~~~Try changing them and see how it affects the play.~~~~~~~~~~~~~~
1020 DATA   1,120,505,1E22,1,880,3000,1E30,1,80,1000,1E16,1,475,3050,1E14
1030 FOR I=1 TO 16: READ G(I): NEXT
1040 RETURN
1500 SOUND 1,0,0: SOUND 0,0,0: LET MES$="Do You Want To Go First?": GOSUB 560
1510 IF IN$="Y" THEN GOTO 5060
1520 IF IN$<>"N" THEN GOTO 1500 ELSE LET X=INT(RND*8+1): GOTO 6210
5000 REM ~~~~~~~~~Human Rountine~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
5060 SOUND 1,0,0: FOR I=1 TO 500: NEXT : LET MES$="Pick a number (1 to 8)"
5061 GOSUB 560
5070 LET X=VAL(IN$): LET X=INT(X)
5080 IF X>=1 AND X<=8 THEN GOTO 5130
5090 CSR 5,22: PRINT CHR$(5);: CSR 5,22: COLOUR 0,9: INK 15
5095 PRINT "ILLEGAL INPUT (1 TO 8)": SOUND 1,600,15: PAUSE 200: SOUND 1,900,12
5100 SOUND 1,0,0: SOUND 1,900,15: PAUSE 200: SOUND 1,0,0: GOTO 5060
5130 LET R=R(X): IF R>7 THEN GOTO 5090
5140 LET R(X)=R+1: LET R=R+1: LET E=GC(X): LET F=DN(R): LET TG$(R,X)=H$
5150 CSR E,F: PAPER 1: INK 3: PRINT H$;
5160 LET X$=H$: SOUND 1,100,15: GOSUB 60
5170 FOR I=1 TO 4: IF A(I)<4 THEN GOTO 6000
5180 LET I=4
5190 FOR I=1 TO 6: CSR 5,22: PRINT "<<<  O.K YOU WIN !!!  >>>";: PAUSE 200
5195 CSR 5,22: PRINT "                              ";: PAUSE 100: NEXT I: GOTO 7000
6000 NEXT I
6001 REM~~~~~~~~~Computer Routine~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
6010 LET P6=0: LET MES$="Thinking ": CSR 5,22: PRINT CHR$(5);
6015 COLOUR 0,1: INK 15: CSR 5,22: PRINT MES$;
6020 LET Z1=13: LET Z2=22: CSR Z1,Z2: INK 2: PRINT TH$;
6025 LET U=0: LET J=1: FOR P=1 TO 8: LET R=R(P)+1
6030 IF R>8 THEN GOTO 6181
6040 LET E=1: LET X$=C$: LET F=0: LET X=P
6045 GOSUB 60
```

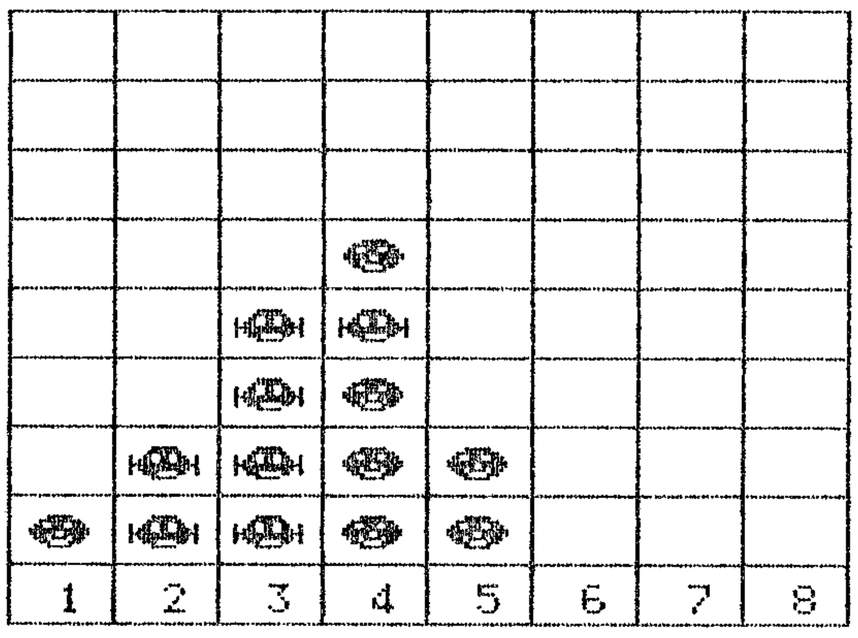
```
6050 FOR L=1 TO 4: LET J(L)=0: NEXT L
6060 FOR I=1 TO 4: LET A=A(I): IF A-F>3 THEN LET I=4: GOTO 6210
6070 LET Q=A+K(I): IF Q<4 THEN GOTO 6090
6080 LET E=E+4: LET J(A)=J(A)+1
6090 NEXT I
6100 FOR I=1 TO 4: LET W=J(I)-1: IF W=-1 THEN GOTO 6130
6110 LET Z=8*F+4*SGN(W)+I
6120 LET E=E+G(Z)+W*G(8*F+I)
6130 NEXT I
6140 IF F=1 THEN GOTO 6155
6150 LET F=1: LET X$=H$: GOTO 6045
6155 LET R=R+1: IF R>8 THEN GOTO 6170
6160 GOSUB 60
6165 FOR I=1 TO 4: IF A(I)>3 THEN LET E=2 ELSE NEXT I
6170 IF E<U THEN GOTO 6181
6171 IF E>U THEN LET O=1: GOTO 6180
6175 LET O=O+1: IF RND>1/O THEN GOTO 6181
6180 LET U=E: LET P6=P
6181 CSR Z1,Z2: INK 6: PRINT CHR$(134);: LET Z1=Z1+1: SOUND 1,500,14: NEXT P
6185 IF P6<>0 THEN GOTO 6200 ELSE CSR 10,22: INK 0: PRINT CHR$(5);
6190 CSR 5,22: INK 5: PRINT "~~~ IT'S A DRAW ~~~";: PAUSE 1000
6195 GOTO 7000
6200 LET X=P6
6210 CSR 5,22: INK 0: PRINT CHR$(5);: CSR 5,22: INK 15: COLOUR 0,1
6215 PRINT "I'm Going in Column";X;
6230 LET R=R(X)+1: LET R(X)=R(X)+1
6240 LET TG$(R,X)=C$
6250 LET X$=C$
6260 LET E=GC(X): LET F=DN(R): SOUND 1,0,0
6270 FOR I=1 TO 3: CSR E,F: PAPER 1: INK 8: PRINT "  ";: PAUSE 100: CSR E,F
6271 PRINT C$;: PAUSE 200: NEXT : SOUND 1,170,15: PAUSE 50
6280 GOSUB 60
6290 FOR I=1 TO 4: IF A(I)<4 THEN NEXT I: GOTO 5060
6295 LET I=4
6300 CSR 5,22: PRINT CHR$(5);
6310 FOR I=1 TO 8: CSR 5,22: INK 7: PRINT "^^^^ SORRY I WIN ^^^^";: PAUSE 300
6315 CSR 5,22: INK 13: PRINT "\\\ Ha! Ha! Ha! ///": PAUSE 300: NEXT
7000 PAUSE 1000
7010 LET MES$="Do You Want Another Game?": GOSUB 560
7020 IF IN$="Y" THEN CLEAR : GOTO 10
7030 IF IN$<>"N" THEN GOTO 7010
7040 CLS : STOP
7500 REM TO ENTER NODDY PROGRAM :-
7510 REM ENTER LISTING AS ABOVE AND WHEN FINISHED ENTER "NODDY"
7520 REM Type: NODDY / RET
7530 REM IN RESPONSE TO: Noddy> type as follows
7540 REM Noddy>INSTRUCTIONS                                    RET
```

```
50 REM       CONNECT FOUR
50 REM THE GAME CONSISTS OF PLACING YOUR
70 REM MARKERS ON A BOARD WITH THE INTENTION
30 REM OF TRYING TO GET FOUR OF YOUR OWN
90 REM MARKERS IN A ROW
00 REM                DIAGONALLY
10 REM                VERTICALLY
20 REM                HORIZONTALLY
30 REM
40 REM THE COMPUTER WILL TRY TO OUT - WIT YOU
50 REM SO BE ON YOUR GUARD.
50 REM
70 REM PRESS 'RETURN' TO CONTINUE
30 REM Now press RET
30 REM Noddy> PROG1
00 REM         RET
10 REM      *DISPLAY INSTRUCTIONS.
20 REM      *ENTER
30 REM      *RETURN
40 REM now type:- RET                                    DIR
50 REM             RET                                   CLS
50 REM             RET
70 REM The program is now installed in the computer.
30 REM it can be saved   with the main program by typing:
90 REM      SAVE "CONNECT FOUR"                          RET
```

USING THE GR$ INSTRUCTION.

Have you had difficulty getting the GR$ instruction to work ? I know the Basic Manual is not very clear on the subject. You MUST assign the GR$ to a string variable. E.g LET A$ = GR$.

CONNECT FOUR



Do You Want Another Game?

Screen_Dump

```
REM SET UP PRINTER TO GRAPHICS MODE
10 VS4:LPRINT CHR$(27);"3";CHR$(24);
20 FOR Y=191 TO 0 STEP -8
30 LPRINT CHR$(27);"K";CHR$(255);CHR$(0);
40 FOR X=0 TO 255:LET A$ = GR$(X,Y,8)
50 LPRINT A$;
60 NEXT X:LPRINT:NEXT Y
```

PREMIER EDITION

MEMOPAD the official magazine of GENPAT - Memotech MTX User Club.

NEW SOFTWARE RELEASES FROM CONTINENTAL.

FDX BASIC. Enhanced MTX basic on disc for use with the FDX Floppy Disc Systems.
[Full review of this program in later edition]

PHAID. Now released. A cross between Space Invaders & Galaxian. Presents a challenge to the most ardent 'shoot-em-up' fan. Really fast ! ....From Continental Software or your local dealers.

FIRST LETTERS & WORD AND PICTURE Two nice programs for the young children. Continental Software.

GRAFICS A utility program from Continental. A must for any serious programmer.

HI-SOFT PASCAL This implementation on ROM board from Memotech.

SNOWBALL The ultimate in adventure games from Level 9 Computing.

See last page of this issue for full listing of available software.

VRAM LOCATIONS USED BY THE MTX ROM

TEXT MODE

```
TEXT SCREEN [ PAT NAME TABLE ] ..... 7168  ..... £1C00
GRAPHICS    [ PAT GEN  TABLE ] ..... 6144  ..... £1800
```

GRAPHICS MODE

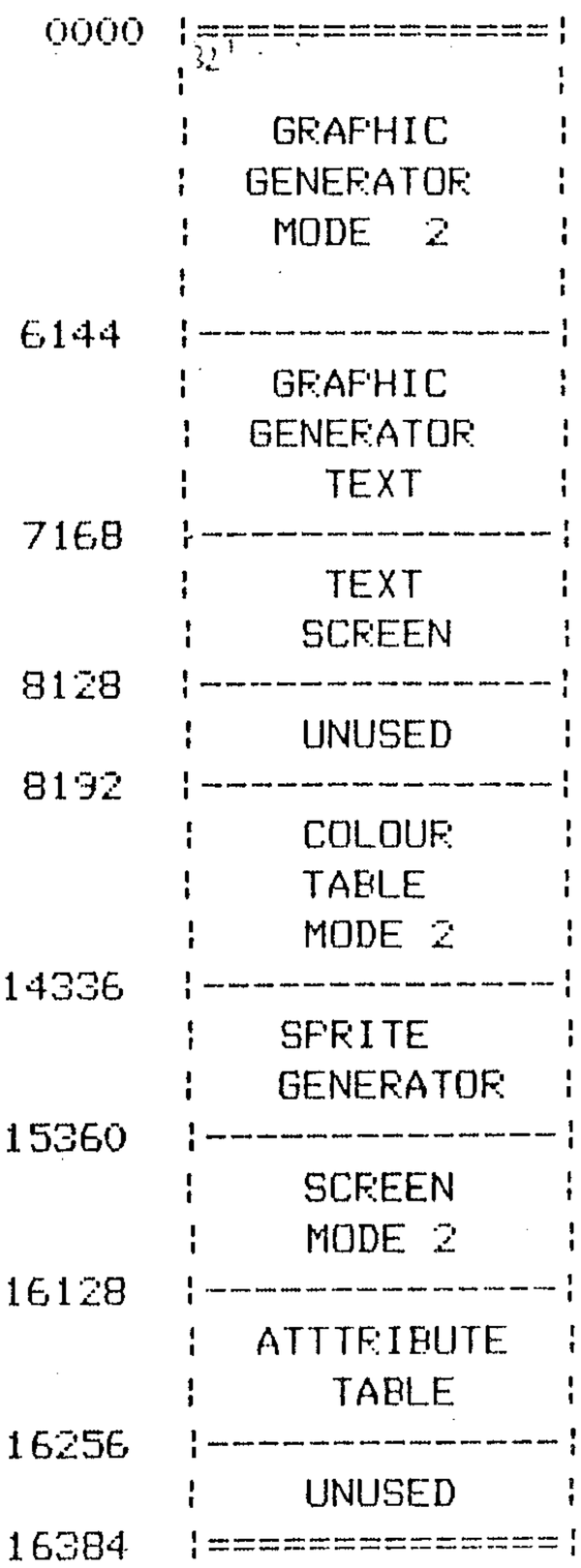
```
SCREEN      [ PAT NAME TABLE ] ..... 15360 ..... £3C00
GRAPHICS    [ PAT GEN  TABLE ] .... 0000 - 6143 ..... £0000 - £17FF
COLOUR                         .... 8192 -14335 ..... £2000 - £37FF
SPRITE ATTRIBUTE TABLE         ..... 16218 .....£3F00
SPRITE GENERATOR TABLE         ..... 14336 .....£3800
```

VALUES REQUIRED TO SEND TO VDP REGISTERS .....

DB £02,£C2,£0F,£FF,£03,£7E,£07,Colour

MEMOPAD

VRAM MEMORY MAP AS USED BY MTX BASIC



VALUES USED TO SET UP THE ABOVE ADDRESSES

```
REGSET:     DB  £02,£C2,£0F,£FF,£03,£7E,£07,£04
```

  Vers 3.4.84

GRASS ROOTS BASIC FOR BEGINNERS

Judging from the letters I have received from people new to computing, and even from owners who have been using Microsoft Basic until they purchased the MTX, there is some confusion on how to dimension arrays under MTX Basic. I hope the following explanation will help to clarify the points raised in your letters.

One of the more powerful commands available to Basic is the DIMension statement.

The DIM statement is used to dimension a type of variable called a **subscripted variable.** A subscripted variable is held in memory as an ordered list called an **array.** There is nothing mysterious about an array - in fact, all of you will have written an array at sometime or another, when you made out a shopping list of the items you wanted from your local shop, or a christmas present list.

Computer arrays can be one dimensional, two dimensional, or **dimensioned** to the limits of your computer memory.

The array variable [subscripted variable] is treated in exactly the same manner as any other type of variable. E.g LET A(2) = 57

If we **dimension** an array with the statement - DIM A(8), we have told the MTX to reserve enough memory for 8 **elements** for the array named A. We can now fill these empty elements with any values we want to store there :-

```
LET A(1) = 5
LET A(2) = 127
```

or:- `FOR I = 1 TO 8 : LET A(I) = 6 : NEXT I`

Each element of array A , A(1) through to A(8), now holds the value 6.

On the MTX, the DIM statement is also used to dimension the length of **$tring variables.** If a $tring is not dimensioned, the computer will allocate no less than 64 bytes for each $tring. If you are using a $tring that will never hold more than three characters, you can see that this would be a complete waste of memory. It is better, therefore, to get into the habit of dimensioning your $trings at the start of your program. E.g DIM A$(4). This will tell the computer to reserve enough memory for a $tring named A that will never be more than four characters long. From now on A$ can be assigned the word 'GUN' or 'READ', but not 'READS' or 'GUNNER'.

We have already said that an array can be multidimensional, and an example of this type of array is:- DIM A(8,8). To help you grasp how data is stored in this type of array, take a look at the following diagram.

Columns =>	0	1	2	3	4	5	6	7	8
Rows=> 0	~~0,0~~	~~0,1~~	~~0,2~~	~~0,3~~	~~0,4~~	~~0,5~~	~~0,6~~	~~0,7~~	~~0,8~~
1	1,0	1,1	1,2	1,3	1,4	1,5	1,6	1,7	1,8
2	2,0	2,1	2,2	2,3	2,4	2,5	2,6	2,7	2,8
3	3,0	3,2	3,2	3,3	3,4	3,5	3,6	3,7	3,8
4	4,0	4,1	4,2	4,3	4,4	4,5	4,6	4,7	4,8
5	5,0	5,1	5,2	5,3	5,4	5,5	5,6	5,7	5,8
6	6,0	6,1	etc.....						
7	7,0	7,1	etc.....						
8	8,0	etc.....							

~~Not Used~~

It should be obvious from the above, that an array of the dimension (8,8) is capable of holding 64 different values. To gain access to any one of these values use the following formula :-

```
LET ROW = 5 : LET COL = 7
LET X = (ROW,COL)
PRINT X
```

This type of array can be filled with values by using **For Next Loops.**

```
10  DIM A(8,8)
20  DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30
30  DATA 31,32,33, etc ..... up to 72
40  FOR I = 0 TO 8: FOR J = 0 TO 8: READ A(I,J):NEXT J:NEXT I
```

Each location will now be filled with the integers from the Data statement - one to each location.

This type of **Two Dimensional** array is very useful when programming board games. Take a look at the listing for Connect 4 - you will see a couple of this type of array used within the program. Each piece is assigned a place within the array as it is moved. It is a simple matter to make the computer look up each location to check on the progress of the game.

$tring Arrays can also be multi-dimensional. However, they vary slightly in the way they are set up. We have already set up an array for 72 locations using the numerical variable A(8,8). From the opening paragraphs you should also be aware that DIM A$(8,8) initialises a **one dimensional** $string array with each location capable of holding a $tring up to 8 characters long. To set up an array of 72 locations which will allow up to 3 characters, you would use:- DIM A$(8,8,3). **When you write to, or read from the array you treat it just as you would a numerical array.:-**

```
10 LET A$ = A$(8,4)        OR        10 LET A$(8,8) = "TWO"        OR        10 LET A$(8,1) = Z$
```

The usefulness of arrays reaches infinity, and it will repay you enormously later, if you take time to experiment with this type of variable.....ISN'T THE MTX A MARVELLOUS MACHINE !! ?

PREMIER EDITION

MEMOPAD the official magazine of GENPAT - Memotech MTX User Club.

REVIEW **MAXIMA** **PSS SOFTWARE.**

At last, an original game for the MTX - PSS Software assure me this has not yet been released for any other machine.

Maxima is a 'shoot em up' type of game in the space battle genus. You are faced with **sixteen** different screens to battle your way through before you accomplish your mission. No, don't yawn yet, the graphics are superb in this game, and the action gets really furious. These are not your common space invaders, their actions and shapes are different from any you have yet encountered.

The game starts peaceful enough, and the first wave of attackers is easily overcome after a few practice rounds. From now on, the pace hots up and the speed and difficulty increases with each new wave [screen].

The opening sequence is one of the best - be it not original - starts to a game I have seen on the MTX. The four parts of your fighter start in each corner of the screen, and then merge together with a loud explosion.

At the foot of the screen is a temperature gauge that changes whenever you fire your laser cannon. Trigger happy commanders will find no joy in this innovation because any wasted shots will only serve to raise the temperature, and you will find, at a crucial moment, your cannon will not fire - not to be recommended on later screens!

The aliens drop all kinds of bombs, and one..... well, I'll leave you to find out about that one.

After loading the program you have a choice of using the keyboard or joysticks. If you choose the keyboard, you can define which key does what .... a nice touch.

HIGHLY RECOMMENDED AVAILABLE FROM PSS SOFTWARE, 452,Stoney Stanton Road, Coventry. 81346.

COMPETITION * COMPETITION ***

WIN M A X I M A BY PSS SOFTWARE & N E M O BY CONTINENTAL SOFTWARE

The loser gets a copy of ALICE - but we won't mention that !

The writer of the best program [in the opinion of the judges] submitted for publication in the next edition of MEMOPAD will win the two excellent games mentioned above. The program can be on anything you wish. Age will be taken into consideration, so please mention this when you send in your program. The program must be on cassette or floppy disc, which cannot be returned unless you include a SAE. If the program is really excellent we will pass it on to Continental Software who will pay you royalties if they decide to offer it to the public.

SEND YOUR ENTRIES CARE OF THE EDITOR HERE AT MEMOPAD.

MEMOPAD

SYSTEM VARIABLES CHECK SPRITE COLLISIONS

The MTX stores its System Variables in high memory from £FA52 through to £FFFF, and this block is common to both models.Basic recognises address £FE55 [65109] as the start of a Sprite Control Block, and uses this area to temporarily store sprite data. Each active sprite takes up eight bytes so that sprite two would take up addresses 65117 through to 65124. It is an easy matter to extract the X,Y values from these tables, and compare them for coincidence. The SPRTBL is set out as follows:-

1st active Sprite

```
65109 ..............  Y speed.
65110 ..............  Y position [ Lsb ]
65111 ..............  Y position [ Msb ]
65112 ..............  X speed.
65113 ..............  X position [ Lsb ]
65114 ..............  X position [ Msb ]
65115 ..............  Pattern Number.
65116 ..............  Colour.
65117==========================> Start of active sprite 2.
```

SYSTEM VARIABLE USER ERROR £18AF

It is possible to add new commands to MTX Basic by using this System Variable. Whenever Basic comes across an error in a command line, or a direct statement, it jumps to this address before printing the relevant error message on the screen. By re-vectoring this basic routine from £18AF to your own program, you can fool the system into believing that your new command is part of normal Basic.

```
First step:-    LD HL,£18AF
                LD DE,Your error trap.
                LD BC,£03          ;3 BYTES FROM £18AF
                LDIR

2nd step :-     LD HL,£18AF
                LD A,£C3    ;JP INSTRUCTION
                LD (HL),A
                LD A, Lsb of your error trap routine.
                INC HL
                LD (HL),A
                LD A, Msb "     "     "     "     "  "
                INC HL
                LD (HL),A
```

Your routine must have some kind of an error trap to check that Basic is in fact, rejecting your new command and not a true error. If it is an error other than your new

command point the flow of the program to the place you LDIR'd the bytes from £18AF and Basic will take over by jumping back into the error routine. If it is your command,jp to your routine and then back to Basic with a RET instruction. This is one way a renumber routine can be made to operate. Experiment and let us all share in your new super basic commands !!!

NEXT EDITION ::: NEXT EDITION ::: NEXT EDITION ::: NEXT EDITION :::

TEACH YOURSELF ASSEMBLY PART 2....we start turning Connect 4 into a Machine code program.
GRASS ROOTS .....Learn how to use Control Codes for better basic programming.
THE SECRETS OF THE RST10 CALLS .....Take the heartache out of those machine code graphic routines.
LISTINGS GALORE .....Lots of your programs for you to type in.
HOW TO USE INTERRUPT MODE2.....We show you how to move your graphics under control of the VDP.
SYSTEM VARIABLES .....more update on the system.
HOW TO PROGRAM USING THE HI-RES SCREEN .....From setting up the registers to filling ps.
ALL THE LATEST ON THE SOFTWARE SCENE ..... What's new from the keybashers.
AN IN DEPTH LOOK AT DISC BASIC .....FOR THE FDX AND SINGLE DISC VERSIONS.
PLUS ++++++++++ MUCH MORE !!!

REVIEW MTX GRAFIC

A new release from Continental this month is GRAFIC - a graphic utility package to take the chore out designing sprites and graphics. I won't comment on how good it is because I wrote it. All I will say is that I use it constantly in my programming.

You can design Sprites,User Definable Graphics, and alter the standard Ascii codes. When you have finished, you can save all your designs to tape and use them with your basic programs without having to do all those Genpat statements - they are done automatically for you. Machine code programmers can also use this utility. There is a comprehensive instruction pamphlet with it, and the program has been left open so that you can list it and see how it was programmed - this allows the assembly programmer to use the files thin his/her own programs. There is even an address left vacant so that you can insert a routine of your own that can then be called from the program menu.

NEW MEMBERS if you have begged this from another member join now !

Please enrol me as a member of GENPAT the official Memotech User Club. I enclose a cheque/PO for £16.00 [Overseas members £21.00] made payable to GENPAT. Please make sure that I receive MEMOPAD. I realise that this payment is for 12 months from the date of receiving my first issue of MEMOPAD.

NAME ..... {Please complete and send to GENPAT
ADDRESS ..... 3,BULCOCK STREET,BURNLEY, BB10 1UH,}
..... AGE _____ : MACHINE _____
POST CODE ..... EXTRAS _____

MEMOPAD

DISK RELEASE **CONTACT**

Here's a program for all you Rs232 buffs. If you have the FDX system and running CP/m then this is a program you must have. Available from Memotech at Witney. I'll leave the opening page of the manual to do the talking.....

INTRODUCTION

The CONTACT program (and its CP/M support package: H, I, R & T) provides a comprehensive and powerful communications facility for use with MEMOTECH computer systems. The software makes extensive use of hardware facilities available on these machines and consequently will not run on other machines. The CONTACT support programs however are designed to run on any CP/M machine, their function is to provide a suite of utilities, running on a remote machine, to facilitate the transfer of text and binary files to and from the CONTACT host computer.

When CONTACT is running, the computer system is transformed into a terminal device working from one of the two serial ports on the machine. Anything typed at the keyboard is sent down the line and anything received from the line is displayed on the screen. Thus the CONTACT host machine may be used as the console (or a terminal) to the remote computer. Once operating in this way local files may be transmitted down the line (in straight ASCII or as a HEX image) with due regard for one or both of the 2 available flow control protocols. Also any ASCII or HEX image data received from the line may be deposited in a local CP/M file - note that characters are received under interrupt control, and disc accesses during data transfers will not cause any loss of data.

The CONTACT program may be used to communicate with a variety of remote computer systems including micros, minis and mainframes; ASCII file transfers from a remote machine to the CONTACT machine are performed without the need for any flow control protocol as data can be displayed and dumped to disc faster than the highest serial transmission rate. File transfers to a remote computer are a little more complex since some kind of flow control will almost certainly be required to allow the remote computer time to dump its data into file store. Two common protocols are provide-^B-=-Bhieve this: XON/XOFF which is always active and an optional protocol which involves the transmission of a line at a time in response to a prompt character sent from the remote machine.

These options and others are invoked by means of the CONTACT control menu which is rapidly swapped onto the screen when a control-G character is typed, after interaction with the menu the user can return to the CONTACT screen by typing CR when the original display is swapped back in and updated with any characters received during the interaction.

When the CONTACT program is terminated, a file (LAST.LNK), is saved on disc which contains its status information. On re-entering CONTACT this status information is retrieved and the previous operating mode restored. A further refinement is provided whereby if the LAST.LNK file is renamed (to say UNIX.LNK) and CONTACT is invoked quoting the new name (CONTACT UNIX) on the command line then CONTACT will use this instead of the "LAST.LNK" file to obtain its default status information.

TEACH YOURSELF MTX ASSEMBLY LANGUAGE Part 1.

A lot of rubbish is talked about Assembly programming. The people who can use the language seem reluctant to share their knowledge with the novice. **Let me assure you, there is no mystery in writing machine code routines, 90% is just common sense.** Of course, you have to learn the rules of syntax, but you had to do this with Basic. So don't panic, if, at first, things seem complicated, persevere - the fog will soon clear !

We are not going to concern ourselves with the technicalities, such as how long an instruction takes, or why it is better to use one form of instruction instead of another. You can learn the finer points as you become proficient with the fundamentals of the language.

There is no substitute for writing your programs in machine code. Programs will run up to 250 per cent faster, and many techniques not available in Basic will be available to you. **Arcade action in the real sense is only obtainable using this method of programming.**

Machine Code or Assemby ?

When you type a program into the computer using Basic, the computer contains a program, held in ROM, called an **interpreter**. This does exactly that - it interprets your Basic statements into machine code so that your computer can carry out the instructions you have typed into the program. No matter which language you use, Pascal,Basic,Fortran etc,**the only language the computer executes is machine code.**

Assembly language is written using a program called an assembler. Because the MTX is such a sophisticated machine, this program is already in memory, and you do not have to load it from tape or disc. The assembler allows you to use instructions called **mnemonics** which are logical names assigned to the various machine code instructions. E.g the 'Load' group of instructions use the mnemonic 'LD'. The assembler then translates [assembles] these mnemonics into machine code instructions that the computer can understand.

Assembly language SOURCE program:-

```
LD A,32
LD (£401A),A
LD DE,£03FF
```

Assembly language OBJECT program:-[after assembly]

```
£577B  3E20    LD A,32
£577D  321A40  LD (£401A),A
£5780  11FF03  LD DE,£03FF
          :
```

machine code.

Each machine code instruction is actually a set of binary bits arranged in a certain order which represents a state of ON [1] or OFF [0]. The computer recognises the ON/OFF status and acts accordingly.

The same code in binary would look like this:-

```
                00111110 00100000    LD A,32
00110010 00011010 01000000    LD (£401A),A
```

After studying the above example, it should be apparent that it is far easier to recognise errors in hexidecimal than it is with the equivalent binary code. The hexidecimal numbers in the example are actual machine code instructions called the **object code** which have been translated from the mnemonics in the source program. Two hex digits represent 1 byte in the instruction, and instructions can be one,two,three, or four bytes long.

In machine code programming such terms as **One's Complement** and **Two's Complement** soon appear on there scene, and one of the first stumbling blocks the novice comes across is how an 8 bit binary number can sometimes represent 255 decimal sometimes -127.
To help us overcome these problems,an elementary knowledge of binary and hexidecimal arithmetic is disirable. Don't skip this - it will help you over many of the problems I will
experience in your later programming.

When counting in decimal, we us the digits 0 - 9. To carry on counting after 9 we must go back to 0 and carry 1 into the 'Tens' column and so on. If we write 152 we can also mean,

```
    2     1     0
[1*100]+[5*10]+[2*1]  or  [1*10 ]+[5*10 ]+[2*10 ]  - remember from your school days:
```

any number raised to the power of zero = 1. From this it can be seen that 152 actuall represents:-

```
   2   1   0
  10  10  10
   1   5   2
```

The binary system only uses 0 and 1. When you count in binary the same rules apply as in decimal, but after counting to 1 we go back to 0 and carry 1 into the next column left. the binary number 0111 can be written as:

```
  2  1  0
2 +2 +2    or  [2*2]+[2*1]+1 = 7
```

In binary each position represents a power of 2 not 10.

Given that 1 byte represents 8 digits [bits], then by examining the following notation of one byte,you can see how easy it is to calculate the equivalent decimal number using

```
  2  1
positional notation [2 ,2 , etc].
```

```
                 7  6  5  4  3  2  1  0
               2  2  2  2  2  2  2  2
               1  1  0  0  0  1  1  1
Bit number     7  6  5  4  3  2  1  0
```

Which is [2*2] 7 times + [2*2] 6 times + [2*2] 2 times etc.= 128 + 64 + 4 + 2 + 1 = 199.
It should be obvious that as you move to the next position left, the previous value doubles.

```
E.g  128 - 64 - 32 - 16 - 8 - 4 - 2 - 1.
Bit number   7    6    5    4    3    2    1    0
```

Binary Addition is quite a simple matter. There are only two digits :-

0+0 = 0 : 0+1 = 1 : 1+1 = 10

so that : 15 = 1111
6 = 0110
21 =10101

Binary Multiplication creates no serious problem. Whatever number base you use, multiplication really consists of adding a number to itself a set number of times. E.g 4 * 3 = 4+4+4 or 3+3+3+3 = 12. If we examine a binary multiplication a number of interesting facts emerge.

Multiplicand .....1010 = 10
multiplier .....0110 = 6
0000 .....partial answer
1010 ..... " " "
1010 ..... " " "
0000 ..... " " "
Product ..... 0111100 = 60

**1. Whenever a 1 appears in the multiplier, the multiplicand is copied into the partial answer column. If an 0 appears in the multiplier, the multiplicand is not copied.

**2. On each step, the partial answer is shifted one place left, even if the multiplier contains an 0. This provides an easy way of multiplying in steps of 2 :-

00000010 = 2 : 2 * 2 = 4 : Shift binary left once = 00000100 = 4

Binary Division is also easy because you can see at a glance if one number will divide into another. Division can also be performed by successive subtraction until a negative remainder is encountered. As division is the inverse of multiplication an easy way of dividing in multiples of 2 is by shifting one place to the right :-

8/4 = 2 : 00001000 = 8 : 00000100 = 4 : 00000010 = 2

You can see that shifting right twice is the same as dividing by four because $4=2^2$. It follows that since $8 = 2^3$, to divide by eight, shift right three times. Although you may not realise the significance of these observations now, rest assured they play an important part in future programming.

Binary Subtraction by normal decimal means is a long winded affair, but we can deal with this problem in another way. However, we must first understand the way computers deal with numbers.

As there are only eight bits in a byte and 00000000 = 0, while 11111111 = 255 in normal binary representation, there is no room for negative or signed numbers. E.g -3,-4, or +9. To get round this problem the computer uses the seventh bit as the sign bit. If the seventh bit is 0 then the number is positive. If the last bit [7th] is 1 then the

number is negative. This is where machine language programmers must know which they are dealing with, that is, 0 - 255 or signed numbers. From the above example we could mistakenly assume that if the positive value of a number is 16 decimal then by changing the seventh bit to a 1 would make it -16; this would be wrong ! When a computer deals with signed numbers, it uses the two's complement of the positive number.

There is nothing mysterious about two's complement, and it's easy to calculate:-

To obtain the two's complement of a number.....

First obtain the one's complement and then add 1

You obtain the one's complement by changing all the 0s to 1s and all the 1s to 0s. E.g :-

```
+16 = 00010000
      11101111 ..........One's complement.
      _______1..........add one
-16 = 11110000 ..........Two's complement.
[which is in fact 240]
```

This may seem a little strange at first, but using two's complement simplifies subtraction and addition with signed numbers. Don't be too worried about the complements of numbers because the Z80A CPU has the NEG instruction, which automatically changes positive integer to its equivalent two's complement negative number.

One final point on signed numbers. The maximum range of numbers obtainable using the seventh bit as the sign is +0 to +127 to -127. This is because, in two's complement, -0 does not exist. Instead -128 takes its place. Why bother with this subject ? Well, when you start single-stepping through your programs to find that elusive 'bug', you must know whether you are dealing with signed or unsigned numbers, and how they are represented.

Once you have learned the rudiments of binary, hexidecimal is easy. In hexidecimal the digits start as in decimal,0 through to 9, and as we are now working to base 16 the numbers 10 to 15 are represented by the letters A,B,C,D,E,F. The decimal number 13 now becomes £0D - the £ signifies that we are working in hexidecimal. 255 becomes £FF.

```
   5   4   3   2   1   0
  16  16  16  16  16  16
```

FF = [15*16*1] + [15*1] = 255

FFF = [15*16*16] + [15*16*1] + [15*1] = 4095

Two hexidecimal digits make up one byte, and one hex digit makes up four bits [a nybble]. Bearing this in mind, you can easily translate from binary to hex or the reverse by treating every four bits as a separate binary number:-

255 = 1111 1111

1111 = 15 : 15 = £0F and the 2nd nybble = 1111 = 15 = £0F : so 255 => £FF.

All of this may seem a little strange at first, but you will soon get the hang of it, and once mastered you'll find working in binary or hex is a lot easier than working on the computer with decimal numbers. You will have also overcome one of the major obstacles in assembly language programming.

REVIEW * OBLOIDS

If you like maze games you'll love this one ! The first thing that struck me about this game was the colourful screen. Full instructions are displayed before entering the program, and I found it a bit irritating that the program reverted to these instructions every time a new game was started. However, this is a minor point.

To play the game you have to search the nine playing [screens] areas looking for power boxes. If you are successful, and you find one of these boxes, you must then take it to an 'outer cell' near the barbed wire perimeter of the maze. Easy, isn't it ?

Oh! I forgot to mention the **Obloid Monsters** who constantly chase your man around the maze. These monsters are not the friendliest of little fellows either, if they catch you, they scratch you. Now a body can only stand so much scratching, and after that it's instant death! But all is not lost. Your man is equipped with a 'power ball' that he can throw at the scratching monsters to dispense instant death on their nasty little heads.

The maze is different too. This is not your common 'packy' maze, but a maze with sealed ends that you can sometimes open up by using the correct combination of keys. So for those of you who just like to whiz around gobbling up monsters, and piling on your score, forget it - this is definitely a maze game with a difference.

Just one little niggle. I wish Continental would have some person proof check the title screens before mastering, as the often found spelling mistake really does spoil the professional finish.

HIGHLY RECOMMENDED.

Available from CONTINENTAL SOFTWARE or most Dealers

^^^ If you find that your local dealer isn't stocking these games, ask him to get in touch with Continental Software on 0993 2997. Also, don't always put the blame at your dealer's door - sometimes the company is very slow on notifying them of the new releases. Anyway, if you want to play the 'good samaritan', show him the last page of this issue with a complete listing of all the programs available at the time of going to press.

MEMOTECH MTX 500 / 512 MEMORY MAP

Address [Hex>	0000	2000	4000	8000	£C000
Page Zero	Monitor ROM available on all Pages 8K	Front Panel 8K	Start Of RAM for MTX 512	Start of 500 RAM. 2nd 16K for 512 RAM	Start 2nd 16K 500.Start 3rd 16K 512.
Page One		BASIC ROM 8K	###For#### ###RAM#### Expansion	3rd Block 16K RAM. Available to 512 only	Available to both pages

The 8K **Monitor Rom** is always available unless you switch it out with the ROM nn, instruction. At Early Morning Startup {switch on}, it disables interrupts, loads the HL register pair with £4000 and jumps to memory location £0194. At this address the monitor checks to see which model it is running on and proceeds to initialise all system variables etc by loading them out of Rom into high ram. All Page zero calls [RST 8, RST 10 etc.] are held in the monitor rom.

The **Front Panel ROM** contains all the routines used in the display and editing of memory. It also contains the routines used for handling the graphic and RST 10 Screen Re-start instructions which start at £2E71.

The **Basic ROM** holds all the necessary instructions to run your Basic programs, check syntax etc, and it also contains the token table which is at address £253B.

Addresses £FA52 through to £FFFF hold all the **System Variables** on both models. Virtual Screen controls are held in addresses starting at £FF5D. System Variable £FE55 [SPRTBL] uses memory from this address through to £FF54 and contains all the data relating to <u>active sprites</u> such as X pos, Y pos, colour, and pattern.

 Vers 4.1.84

THE SE.D COLUMN

THIS COLUMN IS RESERVED FOR YOUR COMPLAINTS OR MOANS & GROANS .... so if you have a chip on your shoulder, share it with other members.

MACHINE CODE TIP

Whenever you write a machine code program there comes a time when you must save all your registers before calling a certain sub-routine. Sometimes this can be every CALL. For instance, I have just finished a contract for Amstrad, the program I was writing made extensive use of ROM calls. Unfortunately, every return from these calls resulted in the corruption of all the major registers. To cut down programming length I required a sub-routine to push and pop all registers at once. This in itself creates a problem - you have to be very careful you do not loose the return address, in the stack, under the register values pushed there by the routine. This routine saves the return address in the HL register pair. This is all well and good providing you do not need to save HL. To overcome this the HL pair in the D and E registers.

```
        :::::::
        ::::::: .......Program flow.
        ::::::
        EX DE,HL        ;SAVE HL IN DE
        CALL SVEREG     ;Push all registers at once
        EX DE,HL        ;Get HL back
        ::::::::.......Program flow
        CALL GETREG     ;Pop all registers at once
        ::::::::
        END
SVEREG:
        EX (SP),HL      ;HL now on stack. Return address in HL.....
        PUSH AF                                                    .
        PUSH BC                                                    .
        PUSH DE                                                    .
        PUSH IX                                                    .
        PUSH IY                                                    .
        PUSH HL         ;Return  address now on top of stack..     .
        RET                                                      .  .
GETREG:                                                          .  .
        POP HL          ;Get Return address  <.................    .
        POP IY                                                     .
        POP IX                                                     .
        POP DE                                                     .
        POP BC                                                     .
        POP AF                                                     ....
        EX (SP),HL      ;Original value now in HL. Ret adr on stack<..:
        RET
```

PREMIER EDITION

MEMOPAD the official magazine of GENPAT - Memotech MTX User Club.

FANTASTIC NEWS ..... MTX 500 DOWN IN PRICE NOW ONLY £199.00

Have you seen the latest sales charts the Memotech is now the 6th best selling computer under £1000.00 - and this was before Memotech dropped the price. BLACK IS BEAUTIFUL !!!

WANT TO EARN SOME MONEY ???

Help keep the MTX computer in the news, and earn money at the same time ! Do you realise, **not one games program** has been published in any of the popular magazines. If you think you have anything interesting to say about the MTX, or you have written a quality program, you can earn money if it is published in the popular computing magazines. P.C. Weekly is always looking for programs. Get cracking at the keyboard - you never know - you might earn enough for that add-on you want.

PCN =>Nickie Robinson,62, Oxford Street,London W1A 2HG

Your Computer => Toby Wolpe,Quadrant House,The Quadrant,Sutton,Surrey SM2 5AS

What Micro => Geoff Bains, Programs,62,Oxford Street, London W1A 2HG

Computing Today => Peter Green, 1, Golden Square,London W1R 3AB

Computer & Video Games => Tim Metcalfe,Durant House,8,Herbal Hill, London EC1R 5EJ.

BOOKS BOOKS BOOKS

If you are stuck for ideas **A GAMUT OF GAMES by Sid Jackson**, and published by Hutchinson, is the book you need. The book contains the rules, and how to play 38 games. Some of them will be completely new to you as Sid Jackson is also a games designer. I bought my copy from W.H Smiths ..... £4.95.

Another excellent book is **COMPUTER GAMESMANSHIP by David Levy.** This book is the complete guide to designing computer games. Your programming will certainly benefit from reading this one. Published by Personal Computer/Century £7.95

A book to watch out for is **ADVANCED PROGRAMMING ON THE MTX by Spencer Bateson.** This should be available within the near future. As soon as I get a copy,I will tell you all about it.

MEMOPAD

MEMBERS SWOPS AND SALES

If you have anything you want to sell, exchange, or give, you can advertise it in this column completely free of charge. Members Only ! Send your copy to the Editor.

SAVING DATA TO TAPE Subroutine

The tape Input/Output Rom routine is located at £0AEE. It is a simply a matter of feedin data to this routine by using the correct registers. Address £FD68 is the system Variable 'TYPE'. If the value held in this location is 1 then Basic 'Loads' data into the computer, if the byte is zero, a 'Save' operation is performed.

```
LD HL,Start address of data
LD DE,Number of bytes to save/load
LD A,£00          ;00 = SAVE : 1 = LOAD
LD (£FD68),A      ;Store at System Variable 'TYPE'
CALL £0AAE        ;Go save or load it.
RET               ;Return to caller.
```

REVIEW STAR COMMAND Continental Software.

This game is more or less a graphic version of Star Trek. You are the commander of a Tachyon Space Craft which is wandering aimlessly through space. Suddenly, your 'Televid' flashes a message from Star Command - You are to intercept a rebel fleet that is winging its way toward the Star ship. Get the idea ??

You can fly your craft in all directions, and the ship is equipped with sheilds, lasers, galactic map, et al. I couldn't fathom out how to use the galactic map. The on-screen instructions didn't offer any advice, and I can only presume that because my copy was a pre-production sample, these instructions will be included on the cassette jacket.

I suppose I'll never make a space pilot - I didn't manage to destroy a solitary Klingon ! Star Command has the makings of a good game. The graphic representation of the fighter's control panel are excellent, and the sound effects when you jump to hyper-drive are just superb.

You can use joysticks, but you will still have to toggle the keyboard for the various functions when controlling your craft. This game should appeal to most space fans.

IF ANY OF YOU GAMES EXPERTS THINK YOU HAVE A HIGH SCORE TO BEAT ALL HIGH SCORES LET US KNOW, AND DON'T FORGET TO MENTION FOR WHICH GAME.

MEMOPAD

LISTING Decimal to Hex Hex to Decimal

```
10 DIM HX$(4),R$(1),X$(2): LET X=4096
20 PAPER 1: CLS
30 INK 2: CSR 14,4: PRINT "MENU"
40 INK 8: CSR 2,6: PRINT "TYPE 1 FOR   DECIMAL to HEX."
50 CSR 2,10: PRINT "TYPE 2  FOR  HEX to DECIMAL"
55 INK 7: CSR 16,8: PRINT "Awaiting Input  ";: INPUT R
70 IF R=1 THEN GOTO 100
80 IF R<>2 THEN GOTO 40
90 GOTO 210
100 LET J=20: INK 13: CLS : CSR 2,6: PRINT "Input Decimal Number ";: INPUT DEC
105 IF NOT (DEC>0) OR NOT (DEC<65536) THEN CSR 2,6: PRINT CHR$(30);: GOTO 100
110 LET DECA=DEC: LET DC1=DEC: LET DEC=DEC/X: GOSUB 160
120 LET CNV=H1: GOSUB 170: LET CNV=H2: GOSUB 170: LET CNV=H3: GOSUB 170
125 LET CNV=H4: GOSUB 170
130 CSR J-5,8: PRINT "£";: CSR 1,22: PRINT "AGAIN ? ['Y'to continue else 'N']";: INPUT R$
140 IF R$="Y" THEN GOTO 100 ELSE GOTO 20
160 LET H1=INT(DEC): LET DEC=H1*X: LET A=DC1-DEC: LET B=A: LET H2=INT(A/256)
165 LET H4=B-(H2*256): LET H3=INT(H4/16): LET H4=H4-(H3*16): RETURN
170 LET CNV=CNV+48: IF CNV>57 THEN LET CNV=CNV+7
180 IF CNV<48 THEN LET CNV=48
190 INK 8: CSR 2,8: PRINT "Decimal ";DECA;" = ";: INK 2: CSR J,8: PRINT CHR$(CNV);
195 INK 8: CSR 26,8: PRINT "Hexidecimal"
200 LET J=J+1: PAUSE 200: RETURN
210 CLS
220 INK 5: CSR 2,6: PRINT "Enter Hexidecimal Number ";: INPUT HX$
230 IF LEN(HX$)<>4 THEN CSR 2,6: PRINT CHR$(30);: GOTO 220
240 LET F1=ASC(LEFT$(HX$,1)): LET F2=ASC(RIGHT$(HX$,3)): LET F3=ASC(RIGHT$(HX$,2))
245 LET F4=ASC(RIGHT$(HX$,1))
250 IF F1>57 THEN LET F1=F1-7 ELSE IF F1<0 THEN LET F1=0
260 IF F2>57 THEN LET F2=F2-7 ELSE IF F2<0 THEN LET F2=0
270 IF F3>57 THEN LET F3=F3-7 ELSE IF F3<0 THEN LET F3=0
280 IF F4>57 THEN LET F4=F4-7 ELSE IF F4<0 THEN LET F4=0
290 LET F1=F1-48: IF F1<0 THEN LET F1=0
300 LET F2=F2-48: IF F2<0 THEN LET F2=0
310 LET F3=F3-48: IF F3<0 THEN LET F3=0
320 LET F4=F4-48: IF F4<0 THEN LET F4=0
330 LET F1=F1*4096: LET F2=F2*256: LET F3=F3*16
340 LET HX=F1+F2+F3+F4
350 INK 2: CSR 8,8: PRINT "Decimal Number = ";HX
360 LET MSB=INT(HX/256)
370 LET LSB=HX-(MSB*256)
380 INK 2: CSR 8,10: PRINT "MSB = ";MSB
390 INK 8: CSR 8,11: PRINT "LSB = ";LSB
410 CSR 1,22: PRINT "ANOTHER ? ['Y' for another go else'N']"
420 LET X$=INKEY$: IF X$="" THEN GOTO 420
430 IF X$<>"Y" THEN GOTO 20
440 GOTO 210
```

The listing on the previous page is in answer to those of you who are having difficulty in converting from one number base to another. For the buffs who like to Poke around in memory , the program also computes the MSB & LSB of decimal numbers.

A little explanation for the novice programmer .....

Each memory location in the MTX is given a number {address}, from 0 to 65535. Any value stored in these locations is converted to an eight bit number [1 byte]. One byte of memory can hold a number in the range 0 to 255 [£00 - £FF]. So, two bytes [16 bits] will hold any integer on a scale from 256 to 65535 [£0100 - £FFFF]. Problems can arise if you fail to realise that all Z80 based computers, including the MTX, store these numbers **back-to-front.** LSB first, MSB last.

If you wanted to poke the decimal integer 31867 into memory location £FFAA, you would first have to convert it into two values - the Least Significant Byte [123], and Most Significant Byte [124] - use the program ! You then poke £FFAA with 123 [£7B], and £FFAB with 124 [£7C].

Memory locations ::::::::> £FFAA ==>£7B
£FFAB ==>£7C

If you now reverse the two bytes into their correct order £7C7B = 31867. You should always bear this in mind when you are poking into memory or peeking at memory. This subject will be dealt with in greater detail in a later edition.

If you look at line 105, you will see another trick-of-the-trade. The boolean statement helps to cut down program space. The program will run faster, and there are many such boolean statements that can be compiled to insert in your programs. Consider the following:-

LET X = X+2: IF X> 3046 THEN LET X = 3046
LET X = X-(X+2<3046) will do exactly the same '' Nuff said. Experiment.

REVIEW DUNGEON ADVENTURE Level 9 Software.

For the past couple of years I have been of the opinion that text adventures are 'old hat' that was until I loaded Dungeon Adventure into the MTX. Participating in this adventure is like reading a good book - you don't want it to end, and you can't put it down. I could not stop playing this game. If I told the truth, the first time I tried it, I crept into bed around 5 am.

Gone are those boring situations, you know the type - "You are standing in a desert, you are surrounded by sand. Facing you is a pyramid with an opening at its base. What Now ? (Yawn yawn - switch off).

Well, listen to this: "You are on a mud-bank, **north of a wide river. A stone bridge spans the water, reaching from the granite cliffs above to the flat lands on the far bank, and a path climbs up to it. There is a piece of driftwood here. A huge packing case, open at one end, rests on the ground .....What Next ?**" And believe me, the answer is not obvious !

MEMOPAD

In Dungeon adventure, you encounter Orcs, queer yellow birds, giants, and many more strange, and weird characters as you go in search of the Demon Lord's treasure. At one point I was confronted by an evil set of fellows whose leader challenged me to a game. I actually experienced a shiver as they appeared before me. I tell no lie, this is the effect Dungeon Adventure has on you. It is an adventure in the true s ense of the word.

Dungeon Adventure has over 200 locations, 100 puzzles to sort through,and weirdos, too numerous to mention, wandering all over the scenario.

The packaging of the game is in keeping with the quality of the programming. Level 9 include a 'Fly Back a Clue' envelope - just in case you get into an impossible situation, and you will ! As a reviewer, I received 'clue sheets', and even when I cheated, and peeked at the prompts, I still found it challenging. I haven't yet completed the quest, but then, I really don't want it to end.

One of the best adventure games I have ever had the pleasure to get my hands on. I can recommend Dungeon Adventure without the slightest fear of being contradicted. This is massive sojourn into the unknown.

Level 9 Computing, 229,Hughenden Road, High Wycombe, Bucks. Tel: 0494 26871.

In the next edition of Memopad I'll let you know how I faired with Colossal Adventure - if I manage to return !

END STATEMENT

I am looking forward to your letters, be they moans, or pats on the back. Everyone concerned with the publication of this edition has worked really hard. Bear in mind, we have not had the benefit of your help with this, the first magazine.

Don't forget that Memotech have a marvellous printer, the DMX 80, which has just bee reduced in price. The company has also just released their new RS128 computer which must have enough memory for the most dedicated key basher.

I do hope you will play your part, and help Memotech spread the word. It's a great computer, and a very friendly company. Just remember, if you have not had the help you expected, Memotech do care. The directors have financed the set-up of this magazine, and User club, so that more assistance can be on hand should you need it.

Finally, can anyone tell me how the MTX got its name ? A FREE copy of **KNUCKLES** for the first correct answer.

MEMOPAD, 3,BULCOCK STREET, BURNLEY BB10 1UH. 0282 57427

MEMOPAD