MEMOTECH MEMORY

David Miles takes an objective look at the MTX-500.

Table 1.	0	2000	4000	8000	C000	FFFF
Page 0	8K Monitor ROM on all pages	Front panel ROM (8K)	512 RAM only	500/512 RAM	500/512 RAM on all pages	
Page 1		Basic ROM (8K)		512 RAM only		

Table 2.			
06 00	0A 00	80	FF
Length of line	Line number	Token for Rem	End marker

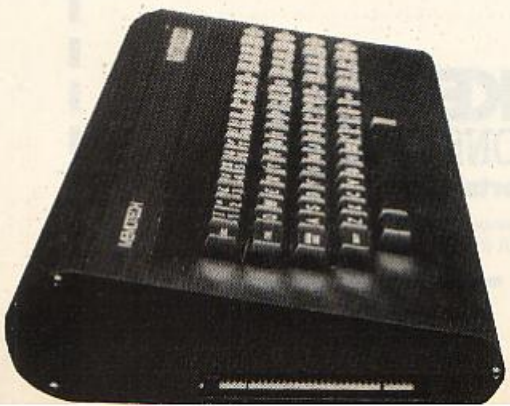
I HAVE HAD MY MTX-500 since late November and I have found it to be an exceptional machine, second only to the BBC.

The ROM is well-structured — unlike the messy layout of the Spectrum ROM — and this gives it the advantage of speed. However, there are bad features with every machine; the worst feature of the Memotech being the inadequate error messages. The cassette routines are also poor — in ROM between hex addresses 2AEE and 2B2F on page 1 — consisting of simple Load, Save and Verify routines with no provision made for Saving machine code, arrays, etc. The Circle command produces an ellipse, like the Oric-1.

This is not a programming error, but in the MTX-series the pixels are stretched laterally, so if a circle is plotted by Sin and Cos — which incidentally are faster than the BBC's mathematical functions — it will still end up as an oval.

The memory of a MTX-series machine is available on 16 64K pages, though only pages 0 and 1 are used in the unexpanded computers — see table 1.

The disadvantage of paged ROM is that it is difficult to keep jumping from one page to the next. For example, you cannot disassemble the Basic ROM on page 1 from the front panel because this is using page 0. The method to apply is to Poke the ROM routines into free RAM on page 0, so that it is possible to study that memory area from the front panel. The 8K monitor ROM contains the vital functions needed to set up the system ready for programming as well as all the graphics routines.



By disassembling the system-C 8K Basic ROM, I have discovered that the MTX-series use a token entry routine like the Sinclair and Commodore machines. This means that each Basic command, string or function has a number in a token table from 128 to 255. As the character codes of the function keys are between 128 and 144, they generate the first 16 tokens, eg., F1 is key 128 and Rem is token 128. By typing F1 and pressing Return, the word Rem is interpreted.

Print out Basic commands

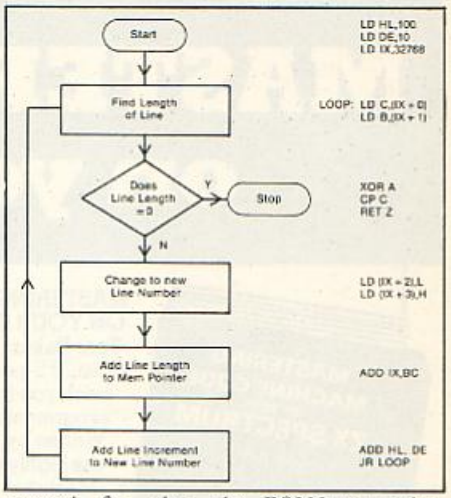
The program will print out the Basic commands, character codes and jump addresses by Peeking the token table:

```
 10 REM * TOKEN TABLE PEEKER *
 20 LET TJUMP = 9975:LET CHAR = 127
 30 FOR A = 9531 TO 9974
 40 LET Z$ = CHAR$(PEEK(A))
 50 IF ASC(Z$)>127 THEN LET
    Z$ = CHAR$(ASC(Z$) - 128) +
    CHR$(9)
 60 IF ASC(Z$)<32 THEN LET Z$ =
    CHR$(9)
 70 PRINT Z$;
 80 IF RIGHT$(Z$,1) = CHR$(9) THEN
    GOSUB 100
 90 NEXT A
100 LET M = PEEK(TJUMP):LET
    N = PEEK(TJUMP + 1)
110 LET ADDRESS = N*256 + M
120 LET CHAR = CHAR + 1:LET
    TJUMP = TJUMP + 2
130 IF CHAR, 193 THEN LET
    ADDRESS = 0
140 PRINT CHAR, ADDRESS
150 RETURN
```

The first thing you will probably notice is that there are six commands in the token table that are not mentioned in the manual. They are:

USER, NODE, FK, OFF, INP, FRE

Inp is the equivalent of the Spectrum's In command. It reads a byte from a port, for instance Print Inp(6) reads the keyboard matrix. Print Fre(X) returns free memory in the RML research machines but in the MTX-series it outputs e to the power of x. e is a constant (2.7128) used as the base for natural logs and in calculus. User jumps to a user routine at address 64137 — named User in the system variables. Node is associated with the network for when the RS232 expansion becomes available.



The screen VRAM cannot be directly assessed. It is addressed by passing bytes through I/O ports 1 and 2. The necessary technical information is at the back of the manual, but try this routine:

```
10 PRINT " ";
20 FOR A = 1 TO 13
30 READ B
40 OUT (1),B
50 NEXT A
60 DATA 72,105,32,77,84,88,32,85,
   115,101,114,115,33
```

A program is stored in the MTX-500 from hex address 8000 (decimal 32768). If you type the line:

10 REM

and then study the area 8000 hex through the Front Panel, you will see the bytes in table 2.

As line numbers are held in memory as two bytes, lines may be numbered as anything between 0 and 65535. This memory layout makes program renumbering a simple affair.

The procedure for entering an assembly language program is given on page 129 of the manual.

The two Reset keys, when pressed together, are supposed to erase the memory contents so that the computer can accept another program. In fact, the program remains in memory and is just over-written — as an examination of the memory at address 8000 hex will reveal. This means that previously unstoppable programs such as Toado or Kilopede can now be broken into and listed. Load in the program, press the two Reset keys and Poke 64167,1. This Poke makes the computer "remember" that it has a program in memory. The program is now listable.

This final routine will print the amount of memory left in bytes. It should be assembled into the first program line and called by

PRINT USR(32768)

The registers used are BC — top of Basic — and HL — start of system variables.

```
LD BC, (64167)
LD HL, 64082
SBC HL, BC
PUSH HL
POP BC
RET
```

■